

NAME

ffmpeg - ffmpeg video converter

SYNOPSIS

ffmpeg [*global_options*] {[*input_file_options*] -i *input_url*} ... {[*output_file_options*] *output_url*} ...

DESCRIPTION

ffmpeg is a very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high quality polyphase filter.

ffmpeg reads from an arbitrary number of input "files" (which can be regular files, pipes, network streams, grabbing devices, etc.), specified by the "-i" option, and writes to an arbitrary number of output "files", which are specified by a plain output url. Anything found on the command line which cannot be interpreted as an option is considered to be an output url.

Each input or output url can, in principle, contain any number of streams of different types (video/audio/subtitle/attachment/data). The allowed number and/or types of streams may be limited by the container format. Selecting which streams from which inputs will go into which output is either done automatically or with the "-map" option (see the Stream selection chapter).

To refer to input files in options, you must use their indices (0-based). E.g. the first input file is 0, the second is 1, etc. Similarly, streams within a file are referred to by their indices. E.g. "2:3" refers to the fourth stream in the third input file. Also see the Stream specifiers chapter.

As a general rule, options are applied to the next specified file. Therefore, order is important, and you can have the same option on the command line multiple times. Each occurrence is then applied to the next input or output file. Exceptions from this rule are the global options (e.g. verbosity level), which should be specified first.

Do not mix input and output files -- first specify all input files, then all output files. Also do not mix options which belong to different files. All options apply **ONLY** to the next input or output file and are reset between files.

- ⊕ To set the video bitrate of the output file to 64 kbit/s:

```
ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi
```

- ⊕ To force the frame rate of the output file to 24 fps:

```
ffmpeg -i input.avi -r 24 output.avi
```

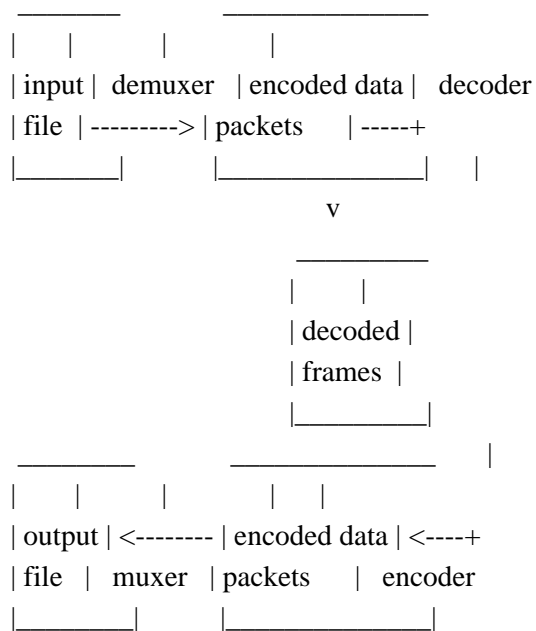
- ⊕ To force the frame rate of the input file (valid for raw formats only) to 1 fps and the frame rate of the output file to 24 fps:

```
ffmpeg -r 1 -i input.m2v -r 24 output.avi
```

The format option may be needed for raw input files.

DETAILED DESCRIPTION

The transcoding process in **ffmpeg** for each output can be described by the following diagram:



ffmpeg calls the libavformat library (containing demuxers) to read input files and get packets containing encoded data from them. When there are multiple input files, **ffmpeg** tries to keep them synchronized by tracking lowest timestamp on any active input stream.

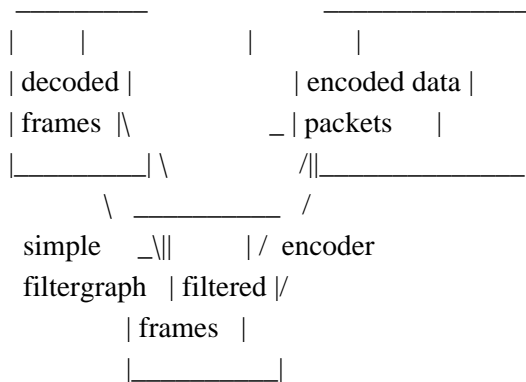
Encoded packets are then passed to the decoder (unless streamcopy is selected for the stream, see further for a description). The decoder produces uncompressed frames (raw video/PCM audio/...) which can be processed further by filtering (see next section). After filtering, the frames are passed to the encoder, which encodes them and outputs encoded packets. Finally those are passed to the muxer, which writes the encoded packets to the output file.

Filtering

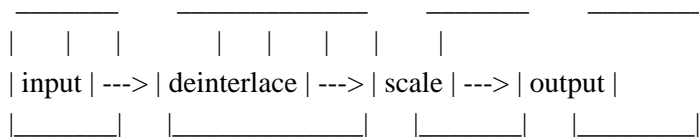
Before encoding, **ffmpeg** can process raw audio and video frames using filters from the libavfilter library. Several chained filters form a filter graph. **ffmpeg** distinguishes between two types of filtergraphs: simple and complex.

Simple filtergraphs

Simple filtergraphs are those that have exactly one input and output, both of the same type. In the above diagram they can be represented by simply inserting an additional step between decoding and encoding:



Simple filtergraphs are configured with the per-stream **-filter** option (with **-vf** and **-af** aliases for video and audio respectively). A simple filtergraph for video can look for example like this:

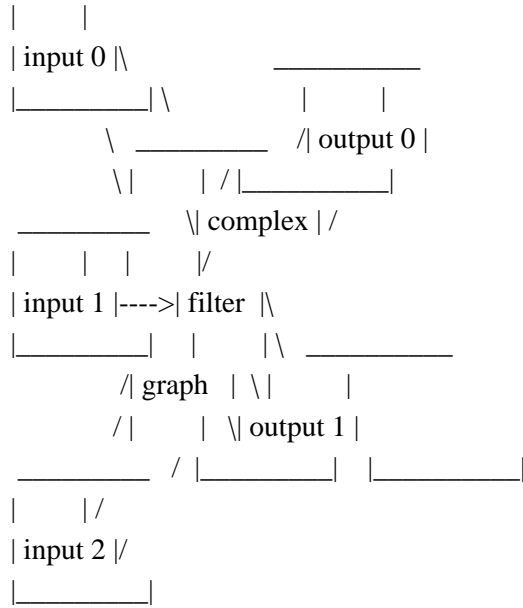


Note that some filters change frame properties but not frame contents. E.g. the "fps" filter in the example above changes number of frames, but does not touch the frame contents. Another example is the "setpts" filter, which only sets timestamps and otherwise passes the frames unchanged.

Complex filtergraphs

Complex filtergraphs are those which cannot be described as simply a linear processing chain applied to one stream. This is the case, for example, when the graph has more than one input and/or output, or when output stream type is different from input. They can be represented with the following diagram:





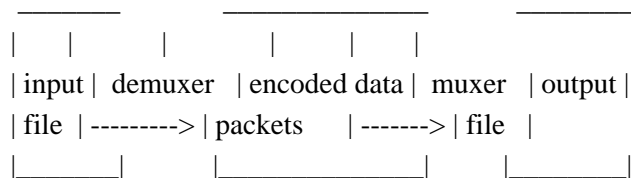
Complex filtergraphs are configured with the **-filter_complex** option. Note that this option is global, since a complex filtergraph, by its nature, cannot be unambiguously associated with a single stream or file.

The **-lavfi** option is equivalent to **-filter_complex**.

A trivial example of a complex filtergraph is the "overlay" filter, which has two video inputs and one video output, containing one video overlaid on top of the other. Its audio counterpart is the "amix" filter.

Stream copy

Stream copy is a mode selected by supplying the "copy" parameter to the **-codec** option. It makes **ffmpeg** omit the decoding and encoding step for the specified stream, so it does only demuxing and muxing. It is useful for changing the container format or modifying container-level metadata. The diagram above will, in this case, simplify to this:



Since there is no decoding or encoding, it is very fast and there is no quality loss. However, it might not work in some cases because of many factors. Applying filters is obviously also impossible, since filters

work on uncompressed data.

STREAM SELECTION

ffmpeg provides the "-map" option for manual control of stream selection in each output file. Users can skip "-map" and let ffmpeg perform automatic stream selection as described below. The "-vn / -an / -sn / -dn" options can be used to skip inclusion of video, audio, subtitle and data streams respectively, whether manually mapped or automatically selected, except for those streams which are outputs of complex filtergraphs.

Description

The sub-sections that follow describe the various rules that are involved in stream selection. The examples that follow next show how these rules are applied in practice.

While every effort is made to accurately reflect the behavior of the program, FFmpeg is under continuous development and the code may have changed since the time of this writing.

Automatic stream selection

In the absence of any map options for a particular output file, ffmpeg inspects the output format to check which type of streams can be included in it, viz. video, audio and/or subtitles. For each acceptable stream type, ffmpeg will pick one stream, when available, from among all the inputs.

It will select that stream based upon the following criteria:

- ⊕ for video, it is the stream with the highest resolution,
- ⊕ for audio, it is the stream with the most channels,
- ⊕ for subtitles, it is the first subtitle stream found but there's a caveat. The output format's default subtitle encoder can be either text-based or image-based, and only a subtitle stream of the same type will be chosen.

In the case where several streams of the same type rate equally, the stream with the lowest index is chosen.

Data or attachment streams are not automatically selected and can only be included using "-map".

Manual stream selection

When "-map" is used, only user-mapped streams are included in that output file, with one possible

exception for filtergraph outputs described below.

Complex filtergraphs

If there are any complex filtergraph output streams with unlabeled pads, they will be added to the first output file. This will lead to a fatal error if the stream type is not supported by the output format. In the absence of the map option, the inclusion of these streams leads to the automatic stream selection of their types being skipped. If map options are present, these filtergraph streams are included in addition to the mapped streams.

Complex filtergraph output streams with labeled pads must be mapped once and exactly once.

Stream handling

Stream handling is independent of stream selection, with an exception for subtitles described below. Stream handling is set via the "-codec" option addressed to streams within a specific *output* file. In particular, codec options are applied by ffmpeg after the stream selection process and thus do not influence the latter. If no "-codec" option is specified for a stream type, ffmpeg will select the default encoder registered by the output file muxer.

An exception exists for subtitles. If a subtitle encoder is specified for an output file, the first subtitle stream found of any type, text or image, will be included. ffmpeg does not validate if the specified encoder can convert the selected stream or if the converted stream is acceptable within the output format. This applies generally as well: when the user sets an encoder manually, the stream selection process cannot check if the encoded stream can be muxed into the output file. If it cannot, ffmpeg will abort and *all* output files will fail to be processed.

Examples

The following examples illustrate the behavior, quirks and limitations of ffmpeg's stream selection methods.

They assume the following three input files.

```
input file 'A.avi'  
  stream 0: video 640x360  
  stream 1: audio 2 channels
```

```
input file 'B.mp4'  
  stream 0: video 1920x1080  
  stream 1: audio 2 channels
```

```
stream 2: subtitles (text)
stream 3: audio 5.1 channels
stream 4: subtitles (text)

input file 'C.mkv'
stream 0: video 1280x720
stream 1: audio 2 channels
stream 2: subtitles (image)
```

Example: automatic stream selection

```
ffmpeg -i A.avi -i B.mp4 out1.mkv out2.wav -map 1:a -c:a copy out3.mov
```

There are three output files specified, and for the first two, no "-map" options are set, so ffmpeg will select streams for these two files automatically.

out1.mkv is a Matroska container file and accepts video, audio and subtitle streams, so ffmpeg will try to select one of each type. For video, it will select "stream 0" from *B.mp4*, which has the highest resolution among all the input video streams. For audio, it will select "stream 3" from *B.mp4*, since it has the greatest number of channels. For subtitles, it will select "stream 2" from *B.mp4*, which is the first subtitle stream from among *A.avi* and *B.mp4*.

out2.wav accepts only audio streams, so only "stream 3" from *B.mp4* is selected.

For *out3.mov*, since a "-map" option is set, no automatic stream selection will occur. The "-map 1:a" option will select all audio streams from the second input *B.mp4*. No other streams will be included in this output file.

For the first two outputs, all included streams will be transcoded. The encoders chosen will be the default ones registered by each output format, which may not match the codec of the selected input streams.

For the third output, codec option for audio streams has been set to "copy", so no decoding-filtering-encoding operations will occur, or *can* occur. Packets of selected streams shall be conveyed from the input file and muxed within the output file.

Example: automatic subtitles selection

```
ffmpeg -i C.mkv out1.mkv -c:s dvdsub -an out2.mkv
```

Although *out1.mkv* is a Matroska container file which accepts subtitle streams, only a video and audio stream shall be selected. The subtitle stream of *C.mkv* is image-based and the default subtitle encoder of the Matroska muxer is text-based, so a transcode operation for the subtitles is expected to fail and hence the stream isn't selected. However, in *out2.mkv*, a subtitle encoder is specified in the command and so, the subtitle stream is selected, in addition to the video stream. The presence of "-an" disables audio stream selection for *out2.mkv*.

Example: unlabeled filtergraph outputs

```
ffmpeg -i A.avi -i C.mkv -i B.mp4 -filter_complex "overlay" out1.mp4 out2.srt
```

A filtergraph is setup here using the "-filter_complex" option and consists of a single video filter. The "overlay" filter requires exactly two video inputs, but none are specified, so the first two available video streams are used, those of *A.avi* and *C.mkv*. The output pad of the filter has no label and so is sent to the first output file *out1.mp4*. Due to this, automatic selection of the video stream is skipped, which would have selected the stream in *B.mp4*. The audio stream with most channels viz. "stream 3" in *B.mp4*, is chosen automatically. No subtitle stream is chosen however, since the MP4 format has no default subtitle encoder registered, and the user hasn't specified a subtitle encoder.

The 2nd output file, *out2.srt*, only accepts text-based subtitle streams. So, even though the first subtitle stream available belongs to *C.mkv*, it is image-based and hence skipped. The selected stream, "stream 2" in *B.mp4*, is the first text-based subtitle stream.

Example: labeled filtergraph outputs

```
ffmpeg -i A.avi -i B.mp4 -i C.mkv -filter_complex "[1:v]hue=s=0[outv];overlay;aresample" \
    -map '[outv]' -an out1.mp4 \
    out2.mkv \
    -map '[outv]' -map 1:a:0 out3.mkv
```

The above command will fail, as the output pad labelled "[outv]" has been mapped twice. None of the output files shall be processed.

```
ffmpeg -i A.avi -i B.mp4 -i C.mkv -filter_complex "[1:v]hue=s=0[outv];overlay;aresample" \
    -an out1.mp4 \
    out2.mkv \
    -map 1:a:0 out3.mkv
```

This command above will also fail as the hue filter output has a label, "[outv]", and hasn't been mapped anywhere.

The command should be modified as follows,

```
ffmpeg -i A.avi -i B.mp4 -i C.mkv -filter_complex "[1:v]hue=s=0,split=2[outv1][outv2];overlay;aresample" \
    -map '[outv1]' -an out1.mp4 \
        out2.mkv \
    -map '[outv2]' -map 1:a:0 out3.mkv
```

The video stream from *B.mp4* is sent to the hue filter, whose output is cloned once using the split filter, and both outputs labelled. Then a copy each is mapped to the first and third output files.

The overlay filter, requiring two video inputs, uses the first two unused video streams. Those are the streams from *A.avi* and *C.mkv*. The overlay output isn't labelled, so it is sent to the first output file *out1.mp4*, regardless of the presence of the "-map" option.

The aresample filter is sent the first unused audio stream, that of *A.avi*. Since this filter output is also unlabelled, it too is mapped to the first output file. The presence of "-an" only suppresses automatic or manual stream selection of audio streams, not outputs sent from filtergraphs. Both these mapped streams shall be ordered before the mapped stream in *out1.mp4*.

The video, audio and subtitle streams mapped to "out2.mkv" are entirely determined by automatic stream selection.

out3.mkv consists of the cloned video output from the hue filter and the first audio stream from *B.mp4*.

OPTIONS

All the numerical options, if not specified otherwise, accept a string representing a number as input, which may be followed by one of the SI unit prefixes, for example: 'K', 'M', or 'G'.

If 'i' is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiples, which are based on powers of 1024 instead of powers of 1000. Appending 'B' to the SI unit prefix multiplies the value by 8. This allows using, for example: 'KB', 'MiB', 'G' and 'B' as number suffixes.

Options which do not take arguments are boolean options, and set the corresponding value to true. They can be set to false by prefixing the option name with "no". For example using "-nofoo" will set the boolean option with name "foo" to false.

Stream specifiers

Some options are applied per-stream, e.g. bitrate or codec. Stream specifiers are used to precisely specify which stream(s) a given option belongs to.

A stream specifier is a string generally appended to the option name and separated from it by a colon. E.g. "-codec:a:1 ac3" contains the "a:1" stream specifier, which matches the second audio stream. Therefore, it would select the ac3 codec for the second audio stream.

A stream specifier can match several streams, so that the option is applied to all of them. E.g. the stream specifier in "-b:a 128k" matches all audio streams.

An empty stream specifier matches all streams. For example, "-codec copy" or "-codec: copy" would copy all the streams without reencoding.

Possible forms of stream specifiers are:

stream_index

Matches the stream with this index. E.g. "-threads:1 4" would set the thread count for the second stream to 4. If *stream_index* is used as an additional stream specifier (see below), then it selects stream number *stream_index* from the matching streams. Stream numbering is based on the order of the streams as detected by libavformat except when a program ID is also specified. In this case it is based on the ordering of the streams in the program.

stream_type[:additional_stream_specifier]

stream_type is one of following: 'v' or 'V' for video, 'a' for audio, 's' for subtitle, 'd' for data, and 't' for attachments. 'v' matches all video streams, 'V' only matches video streams which are not attached pictures, video thumbnails or cover arts. If *additional_stream_specifier* is used, then it matches streams which both have this type and match the *additional_stream_specifier*. Otherwise, it matches all streams of the specified type.

p:*program_id[:additional_stream_specifier]*

Matches streams which are in the program with the id *program_id*. If *additional_stream_specifier* is used, then it matches streams which both are part of the program and match the *additional_stream_specifier*.

#stream_id or **i**:*stream_id*

Match the stream by stream id (e.g. PID in MPEG-TS container).

m:*key[:value]*

Matches streams with the metadata tag *key* having the specified value. If *value* is not given, matches streams that contain the given tag with any value.

u Matches streams with usable configuration, the codec must be defined and the essential information such as video dimension or audio sample rate must be present.

Note that in **ffmpeg**, matching by metadata will only work properly for input files.

Generic options

These options are shared amongst the ff* tools.

-L Show license.

-h, -?, -help, --help [*arg*]

Show help. An optional parameter may be specified to print help about a specific item. If no argument is specified, only basic (non advanced) tool options are shown.

Possible values of *arg* are:

long

Print advanced tool options in addition to the basic tool options.

full Print complete list of options, including shared and private options for encoders, decoders, demuxers, muxers, filters, etc.

decoder=*decoder_name*

Print detailed information about the decoder named *decoder_name*. Use the **-decoders** option to get a list of all decoders.

encoder=*encoder_name*

Print detailed information about the encoder named *encoder_name*. Use the **-encoders** option to get a list of all encoders.

demuxer=*demuxer_name*

Print detailed information about the demuxer named *demuxer_name*. Use the **-formats** option to get a list of all demuxers and muxers.

muxer=*muxer_name*

Print detailed information about the muxer named *muxer_name*. Use the **-formats** option to get a list of all muxers and demuxers.

filter=*filter_name*

Print detailed information about the filter named *filter_name*. Use the **-filters** option to get a list of all filters.

bsf=*bitstream_filter_name*

Print detailed information about the bitstream filter named *bitstream_filter_name*. Use the **-bsfs** option to get a list of all bitstream filters.

protocol=*protocol_name*

Print detailed information about the protocol named *protocol_name*. Use the **-protocols** option to get a list of all protocols.

-version

Show version.

-buildconf

Show the build configuration, one option per line.

-formats

Show available formats (including devices).

-demuxers

Show available demuxers.

-muxers

Show available muxers.

-devices

Show available devices.

-codecs

Show all codecs known to libavcodec.

Note that the term 'codec' is used throughout this documentation as a shortcut for what is more correctly called a media bitstream format.

-decoders

Show available decoders.

-encoders

Show all available encoders.

-bsfs

Show available bitstream filters.

-protocols

Show available protocols.

-filters

Show available libavfilter filters.

-pix_fmts

Show available pixel formats.

-sample_fmts

Show available sample formats.

-layouts

Show channel names and standard channel layouts.

-dispositions

Show stream dispositions.

-colors

Show recognized color names.

-sources *device*[,*opt1=val1*[,*opt2=val2*]...]

Show autodetected sources of the input device. Some devices may provide system-dependent source names that cannot be autodetected. The returned list cannot be assumed to be always complete.

```
ffmpeg -sources pulse,server=192.168.0.4
```

-sinks *device*[,*opt1=val1*[,*opt2=val2*]...]

Show autodetected sinks of the output device. Some devices may provide system-dependent sink names that cannot be autodetected. The returned list cannot be assumed to be always complete.

```
ffmpeg -sinks pulse,server=192.168.0.4
```

-loglevel [*flags+*]*loglevel* | **-v** [*flags+*]*loglevel*

Set logging level and flags used by the library.

The optional *flags* prefix can consist of the following values:

repeat

Indicates that repeated log output should not be compressed to the first line and the "Last message repeated n times" line will be omitted.

level

Indicates that log output should add a "[level]" prefix to each message line. This can be used as an alternative to log coloring, e.g. when dumping the log to file.

Flags can also be used alone by adding a '+'/'-' prefix to set/reset a single flag without affecting other *flags* or changing *loglevel*. When setting both *flags* and *loglevel*, a '+' separator is expected between the last *flags* value and before *loglevel*.

loglevel is a string or a number containing one of the following values:

quiet, -8

Show nothing at all; be silent.

panic, 0

Only show fatal errors which could lead the process to crash, such as an assertion failure. This is not currently used for anything.

fatal, 8

Only show fatal errors. These are errors after which the process absolutely cannot continue.

error, 16

Show all errors, including ones which can be recovered from.

warning, 24

Show all warnings and errors. Any message related to possibly incorrect or unexpected events will be shown.

info, 32

Show informative messages during processing. This is in addition to warnings and errors. This is the default value.

verbose, 40

Same as "info", except more verbose.

debug, 48

Show everything, including debugging information.

trace, 56

For example to enable repeated log output, add the "level" prefix, and set *loglevel* to "verbose":

```
ffmpeg -loglevel repeat+level+verbose -i input output
```

Another example that enables repeated log output without affecting current state of "level" prefix flag or *loglevel*:

```
ffmpeg [...] -loglevel +repeat
```

By default the program logs to stderr. If coloring is supported by the terminal, colors are used to mark errors and warnings. Log coloring can be disabled setting the environment variable **AV_LOG_FORCE_NOCOLOR**, or can be forced setting the environment variable **AV_LOG_FORCE_COLOR**.

-report

Dump full command line and log output to a file named "*program-YYYYMMDD-HHMMSS.log*" in the current directory. This file can be useful for bug reports. It also implies "-loglevel debug".

Setting the environment variable **FFREPORT** to any value has the same effect. If the value is a ':'-separated key=value sequence, these options will affect the report; option values must be escaped if they contain special characters or the options delimiter ':' (see the "Quoting and escaping" section in the ffmpeg-utils manual).

The following options are recognized:

file set the file name to use for the report; %p is expanded to the name of the program, %t is expanded to a timestamp, "%%" is expanded to a plain "%"

level

set the log verbosity level using a numerical value (see "-loglevel").

For example, to output a report to a file named *ffreport.log* using a log level of 32 (alias for log level "info"):

```
FFREPORT=file=ffreport.log:level=32 ffmpeg -i input output
```

Errors in parsing the environment variable are not fatal, and will not appear in the report.

-hide_banner

Suppress printing banner.

All Ffmpeg tools will normally show a copyright notice, build options and library versions. This option can be used to suppress printing this information.

-cpuflags flags (*global*)

Allows setting and clearing cpu flags. This option is intended for testing. Do not use it unless you know what you're doing.

```
ffmpeg -cpuflags -sse+mmx ...  
ffmpeg -cpuflags mmx ...  
ffmpeg -cpuflags 0 ...
```

Possible flags for this option are:

x86

- mmx**
- mmxext**
- sse**
- sse2**
- sse2slow**
- sse3**
- sse3slow**
- ssse3**
- atom**
- sse4.1**
- sse4.2**
- avx**
- avx2**
- xop**
- fma3**
- fma4**
- 3dnow**
- 3dnowext**
- bmi1**
- bmi2**
- cmov**

ARM

- armv5te**

armv6
armv6t2
vfp
vfpv3
neon
setend

AArch64

armv8
vfp
neon

PowerPC

altivec

Specific Processors

pentium2
pentium3
pentium4
k6
k62
athlon
athlonxp
k8

-cpucount *count* (*global*)

Override detection of CPU count. This option is intended for testing. Do not use it unless you know what you're doing.

```
ffmpeg -cpucount 2
```

-max_alloc *bytes*

Set the maximum size limit for allocating a block on the heap by ffmpeg's family of malloc functions. Exercise **extreme caution** when using this option. Don't use if you do not understand the full consequence of doing so. Default is INT_MAX.

AVOptions

These options are provided directly by the libavformat, libavdevice and libavcodec libraries. To see the list of available AVOptions, use the **-help** option. They are separated into two categories:

generic

These options can be set for any container, codec or device. Generic options are listed under AVFormatContext options for containers/devices and under AVCodecContext options for codecs.

private

These options are specific to the given container, device or codec. Private options are listed under their corresponding containers/devices/codecs.

For example to write an ID3v2.3 header instead of a default ID3v2.4 to an MP3 file, use the **id3v2_version** private option of the MP3 muxer:

```
ffmpeg -i input.flac -id3v2_version 3 out.mp3
```

All codec AVOptions are per-stream, and thus a stream specifier should be attached to them:

```
ffmpeg -i multichannel.mxf -map 0:v:0 -map 0:a:0 -map 0:a:0 -c:a:0 ac3 -b:a:0 640k -ac:a:1 2 -c:a:1 aac -b:2 128k
```

In the above example, a multichannel audio stream is mapped twice for output. The first instance is encoded with codec ac3 and bitrate 640k. The second instance is downmixed to 2 channels and encoded with codec aac. A bitrate of 128k is specified for it using absolute index of the output stream.

Note: the **-nooption** syntax cannot be used for boolean AVOptions, use **-option 0/-option 1**.

Note: the old undocumented way of specifying per-stream AVOptions by prepending v/a/s to the options name is now obsolete and will be removed soon.

Main options**-f *fmt*** (*input/output*)

Force input or output file format. The format is normally auto detected for input files and guessed from the file extension for output files, so this option is not needed in most cases.

-i *url* (*input*)

input file url

-y (*global*)

Overwrite output files without asking.

-n (*global*)

Do not overwrite output files, and exit immediately if a specified output file already exists.

-stream_loop *number* (*input*)

Set number of times input stream shall be looped. Loop 0 means no loop, loop -1 means infinite loop.

-recast_media (*global*)

Allow forcing a decoder of a different media type than the one detected or designated by the demuxer. Useful for decoding media data muxed as data streams.

-c[:stream_specifier] *codec* (*input/output,per-stream*)**-codec[:stream_specifier]** *codec* (*input/output,per-stream*)

Select an encoder (when used before an output file) or a decoder (when used before an input file) for one or more streams. *codec* is the name of a decoder/encoder or a special value "copy" (output only) to indicate that the stream is not to be re-encoded.

For example

```
ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT
```

encodes all video streams with libx264 and copies all audio streams.

For each stream, the last matching "c" option is applied, so

```
ffmpeg -i INPUT -map 0 -c copy -c:v:1 libx264 -c:a:137 libvorbis OUTPUT
```

will copy all the streams except the second video, which will be encoded with libx264, and the 138th audio, which will be encoded with libvorbis.

-t *duration* (*input/output*)

When used as an input option (before "-i"), limit the *duration* of data read from the input file.

When used as an output option (before an output url), stop writing the output after its duration reaches *duration*.

duration must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1) manual**.

-to and -t are mutually exclusive and -t has priority.

-to *position* (*input/output*)

Stop writing the output or reading the input at *position*. *position* must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1) manual**.

-to and -t are mutually exclusive and -t has priority.

-fs *limit_size (output)*

Set the file size limit, expressed in bytes. No further chunk of bytes is written after the limit is exceeded. The size of the output file is slightly more than the requested file size.

-ss *position (input/output)*

When used as an input option (before "-i"), seeks in this input file to *position*. Note that in most formats it is not possible to seek exactly, so **ffmpeg** will seek to the closest seek point before *position*. When transcoding and **-accurate_seek** is enabled (the default), this extra segment between the seek point and *position* will be decoded and discarded. When doing stream copy or when **-noaccurate_seek** is used, it will be preserved.

When used as an output option (before an output url), decodes but discards input until the timestamps reach *position*.

position must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1) manual**.

-sseof *position (input)*

Like the "-ss" option but relative to the "end of file". That is negative values are earlier in the file, 0 is at EOF.

-isync *input_index (input)*

Assign an input as a sync source.

This will take the difference between the start times of the target and reference inputs and offset the timestamps of the target file by that difference. The source timestamps of the two inputs should derive from the same clock source for expected results. If "copyts" is set then "start_at_zero" must also be set. If either of the inputs has no starting timestamp then no sync adjustment is made.

Acceptable values are those that refer to a valid ffmpeg input index. If the sync reference is the target index itself or *-1*, then no adjustment is made to target timestamps. A sync reference may not itself be synced to any other input.

Default value is *-1*.

-itsoffset *offset (input)*

Set the input time offset.

offset must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1)**

manual.

The offset is added to the timestamps of the input files. Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in *offset*.

-itsscale *scale* (*input,per-stream*)

Rescale input timestamps. *scale* should be a floating point number.

-timestamp *date* (*output*)

Set the recording timestamp in the container.

date must be a date specification, see **the Date section in the ffmpeg-utils(1) manual**.

-metadata[:metadata_specifier] *key=value* (*output,per-metadata*)

Set a metadata key/value pair.

An optional *metadata_specifier* may be given to set metadata on streams, chapters or programs. See "-map_metadata" documentation for details.

This option overrides metadata set with "-map_metadata". It is also possible to delete metadata by using an empty value.

For example, for setting the title in the output file:

```
ffmpeg -i in.avi -metadata title="my title" out.flv
```

To set the language of the first audio stream:

```
ffmpeg -i INPUT -metadata:s:a:0 language=eng OUTPUT
```

-disposition[:stream_specifier] *value* (*output,per-stream*)

Sets the disposition for a stream.

By default, the disposition is copied from the input stream, unless the output stream this option applies to is fed by a complex filtergraph - in that case the disposition is unset by default.

value is a sequence of items separated by '+' or '-'. The first item may also be prefixed with '+' or '-', in which case this option modifies the default value. Otherwise (the first item is not prefixed) this options overrides the default value. A '+' prefix adds the given disposition, '-' removes it. It is also possible to clear the disposition by setting it to 0.

If no "-disposition" options were specified for an output file, ffmpeg will automatically set the 'default' disposition on the first stream of each type, when there are multiple streams of this type in the output file and no stream of that type is already marked as default.

The "-dispositions" option lists the known dispositions.

For example, to make the second audio stream the default stream:

```
ffmpeg -i in.mkv -c copy -disposition:a:1 default out.mkv
```

To make the second subtitle stream the default stream and remove the default disposition from the first subtitle stream:

```
ffmpeg -i in.mkv -c copy -disposition:s:0 0 -disposition:s:1 default out.mkv
```

To add an embedded cover/thumbnail:

```
ffmpeg -i in.mp4 -i IMAGE -map 0 -map 1 -c copy -c:v:1 png -disposition:v:1 attached_pic out.mp4
```

Not all muxers support embedded thumbnails, and those who do, only support a few formats, like JPEG or PNG.

-program [title=title:][program_num=program_num:]st=stream[:st=stream...] (output)

Creates a program with the specified *title*, *program_num* and adds the specified *stream*(s) to it.

-target type (output)

Specify target file type ("vcd", "svcd", "dvd", "dv", "dv50"). *type* may be prefixed with "pal-", "ntsc-" or "film-" to use the corresponding standard. All the format options (bitrate, codecs, buffer sizes) are then set automatically. You can just type:

```
ffmpeg -i myfile.avi -target vcd /tmp/vcd.mpg
```

Nevertheless you can specify additional options as long as you know they do not conflict with the standard, as in:

```
ffmpeg -i myfile.avi -target vcd -bf 2 /tmp/vcd.mpg
```

The parameters set for each target are as follows.

VCD

<pal>:

```
-f vcd -muxrate 1411200 -muxpreload 0.44 -packetize 2324  
-s 352x288 -r 25  
-codec:v mpeg1video -g 15 -b:v 1150k -maxrate:v 1150k -minrate:v 1150k -bufsize:v 327680  
-ar 44100 -ac 2  
-codec:a mp2 -b:a 224k
```

<ntsc>:

```
-f vcd -muxrate 1411200 -muxpreload 0.44 -packetize 2324  
-s 352x240 -r 30000/1001  
-codec:v mpeg1video -g 18 -b:v 1150k -maxrate:v 1150k -minrate:v 1150k -bufsize:v 327680  
-ar 44100 -ac 2  
-codec:a mp2 -b:a 224k
```

<film>:

```
-f vcd -muxrate 1411200 -muxpreload 0.44 -packetize 2324  
-s 352x240 -r 24000/1001  
-codec:v mpeg1video -g 18 -b:v 1150k -maxrate:v 1150k -minrate:v 1150k -bufsize:v 327680  
-ar 44100 -ac 2  
-codec:a mp2 -b:a 224k
```

SVCD

<pal>:

```
-f svcd -packetize 2324  
-s 480x576 -pix_fmt yuv420p -r 25  
-codec:v mpeg2video -g 15 -b:v 2040k -maxrate:v 2516k -minrate:v 0 -bufsize:v 1835008 -scan_offset 1  
-ar 44100  
-codec:a mp2 -b:a 224k
```

<ntsc>:

```
-f svcd -packetize 2324  
-s 480x480 -pix_fmt yuv420p -r 30000/1001  
-codec:v mpeg2video -g 18 -b:v 2040k -maxrate:v 2516k -minrate:v 0 -bufsize:v 1835008 -scan_offset 1  
-ar 44100  
-codec:a mp2 -b:a 224k
```

<film>:

```
-f svcd -packetize 2324  
-s 480x480 -pix_fmt yuv420p -r 24000/1001
```

```
-codec:v mpeg2video -g 18 -b:v 2040k -maxrate:v 2516k -minrate:v 0 -bufsize:v 1835008 -scan_offset 1  
-ar 44100  
-codec:a mp2 -b:a 224k
```

DVD

```
<pal>:  
-f dvd -muxrate 10080k -packetize 2048  
-s 720x576 -pix_fmt yuv420p -r 25  
-codec:v mpeg2video -g 15 -b:v 6000k -maxrate:v 9000k -minrate:v 0 -bufsize:v 1835008  
-ar 48000  
-codec:a ac3 -b:a 448k
```

```
<ntsc>:  
-f dvd -muxrate 10080k -packetize 2048  
-s 720x480 -pix_fmt yuv420p -r 30000/1001  
-codec:v mpeg2video -g 18 -b:v 6000k -maxrate:v 9000k -minrate:v 0 -bufsize:v 1835008  
-ar 48000  
-codec:a ac3 -b:a 448k
```

```
<film>:  
-f dvd -muxrate 10080k -packetize 2048  
-s 720x480 -pix_fmt yuv420p -r 24000/1001  
-codec:v mpeg2video -g 18 -b:v 6000k -maxrate:v 9000k -minrate:v 0 -bufsize:v 1835008  
-ar 48000  
-codec:a ac3 -b:a 448k
```

DV

```
<pal>:  
-f dv  
-s 720x576 -pix_fmt yuv420p -r 25  
-ar 48000 -ac 2
```

```
<ntsc>:  
-f dv  
-s 720x480 -pix_fmt yuv411p -r 30000/1001  
-ar 48000 -ac 2
```

```
<film>:
```



```
-f dv
-s 720x480 -pix_fmt yuv411p -r 24000/1001
-ar 48000 -ac 2
```

The "dv50" target is identical to the "dv" target except that the pixel format set is "yuv422p" for all three standards.

Any user-set value for a parameter above will override the target preset value. In that case, the output may not comply with the target standard.

-dn (*input/output*)

As an input option, blocks all data streams of a file from being filtered or being automatically selected or mapped for any output. See "-discard" option to disable streams individually.

As an output option, disables data recording i.e. automatic selection or mapping of any data stream. For full manual control see the "-map" option.

-dframes *number* (*output*)

Set the number of data frames to output. This is an obsolete alias for "-frames:d", which you should use instead.

-frames[:*stream_specifier*] *framecount* (*output,per-stream*)

Stop writing to the stream after *framecount* frames.

-q[:*stream_specifier*] *q* (*output,per-stream*)

-qscale[:*stream_specifier*] *q* (*output,per-stream*)

Use fixed quality scale (VBR). The meaning of *q/qscale* is codec-dependent. If *qscale* is used without a *stream_specifier* then it applies only to the video stream, this is to maintain compatibility with previous behavior and as specifying the same codec specific value to 2 different codecs that is audio and video generally is not what is intended when no *stream_specifier* is used.

-filter[:*stream_specifier*] *filtergraph* (*output,per-stream*)

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

filtergraph is a description of the filtergraph to apply to the stream, and must have a single input and a single output of the same type of the stream. In the filtergraph, the input is associated to the label "in", and the output to the label "out". See the ffmpeg-filters manual for more information about the filtergraph syntax.

See the **-filter_complex option** if you want to create filtergraphs with multiple inputs and/or outputs.

-filter_script*[:stream_specifier] filename (output,per-stream)*

This option is similar to **-filter**, the only difference is that its argument is the name of the file from which a filtergraph description is to be read.

-reinit_filter*[:stream_specifier] integer (input,per-stream)*

This boolean option determines if the filtergraph(s) to which this stream is fed gets reinitialized when input frame parameters change mid-stream. This option is enabled by default as most video and all audio filters cannot handle deviation in input frame properties. Upon reinitialization, existing filter state is lost, like e.g. the frame count "n" reference available in some filters. Any frames buffered at time of reinitialization are lost. The properties where a change triggers reinitialization are, for video, frame resolution or pixel format; for audio, sample format, sample rate, channel count or channel layout.

-filter_threads *nb_threads (global)*

Defines how many threads are used to process a filter pipeline. Each pipeline will produce a thread pool with this many threads available for parallel processing. The default is the number of available CPUs.

-pre*[:stream_specifier] preset_name (output,per-stream)*

Specify the preset for matching stream(s).

-stats *(global)*

Print encoding progress/statistics. It is on by default, to explicitly disable it you need to specify "-nostats".

-stats_period *time (global)*

Set period at which encoding progress/statistics are updated. Default is 0.5 seconds.

-progress *url (global)*

Send program-friendly progress information to *url*.

Progress information is written periodically and at the end of the encoding process. It is made of "*key=value*" lines. *key* consists of only alphanumeric characters. The last key of a sequence of progress information is always "progress".

The update period is set using "-stats_period".

-stdin

Enable interaction on standard input. On by default unless standard input is used as an input. To explicitly disable interaction you need to specify "-nostdin".

Disabling interaction on standard input is useful, for example, if ffmpeg is in the background process group. Roughly the same result can be achieved with "ffmpeg ... < /dev/null" but it requires a shell.

-debug_ts (*global*)

Print timestamp information. It is off by default. This option is mostly useful for testing and debugging purposes, and the output format may change from one version to another, so it should not be employed by portable scripts.

See also the option "-fdebug ts".

-attach *filename* (*output*)

Add an attachment to the output file. This is supported by a few formats like Matroska for e.g. fonts used in rendering subtitles. Attachments are implemented as a specific type of stream, so this option will add a new stream to the file. It is then possible to use per-stream options on this stream in the usual way. Attachment streams created with this option will be created after all the other streams (i.e. those created with "-map" or automatic mappings).

Note that for Matroska you also have to set the mimetype metadata tag:

```
ffmpeg -i INPUT -attach DejaVuSans.ttf -metadata:s:2 mimetype=application/x-truetype-font out.mkv
```

(assuming that the attachment stream will be third in the output file).

-dump_attachment[:*stream_specifier*]*filename* (*input,per-stream*)

Extract the matching attachment stream into a file named *filename*. If *filename* is empty, then the value of the "filename" metadata tag will be used.

E.g. to extract the first attachment to a file named 'out.ttf':

```
ffmpeg -dump_attachment:t:0 out.ttf -i INPUT
```

To extract all attachments to files determined by the "filename" tag:

```
ffmpeg -dump_attachment:t "" -i INPUT
```

Technical note -- attachments are implemented as codec extradata, so this option can actually be used to extract extradata from any stream, not just attachments.

Video Options

-vframes *number (output)*

Set the number of video frames to output. This is an obsolete alias for "-frames:v", which you should use instead.

-r*[:stream_specifier] fps (input/output,per-stream)*

Set frame rate (Hz value, fraction or abbreviation).

As an input option, ignore any timestamps stored in the file and instead generate timestamps assuming constant frame rate *fps*. This is not the same as the **-framerate** option used for some input formats like image2 or v4l2 (it used to be the same in older versions of FFmpeg). If in doubt use **-framerate** instead of the input option **-r**.

As an output option:

video encoding

Duplicate or drop frames right before encoding them to achieve constant output frame rate *fps*.

video streamcopy

Indicate to the muxer that *fps* is the stream frame rate. No data is dropped or duplicated in this case. This may produce invalid files if *fps* does not match the actual stream frame rate as determined by packet timestamps. See also the "setts" bitstream filter.

-fpsmax*[:stream_specifier] fps (output,per-stream)*

Set maximum frame rate (Hz value, fraction or abbreviation).

Clamps output frame rate when output framerate is auto-set and is higher than this value. Useful in batch processing or when input framerate is wrongly detected as very high. It cannot be set together with "-r". It is ignored during streamcopy.

-s*[:stream_specifier] size (input/output,per-stream)*

Set frame size.

As an input option, this is a shortcut for the **video_size** private option, recognized by some demuxers for which the frame size is either not stored in the file or is configurable -- e.g. raw video or video grabbers.

As an output option, this inserts the "scale" video filter to the *end* of the corresponding filtergraph. Please use the "scale" filter directly to insert it at the beginning or some other place.

The format is **wxh** (default - same as source).

-aspect[:stream_specifier] aspect (output,per-stream)

Set the video display aspect ratio specified by *aspect*.

aspect can be a floating point number string, or a string of the form *num:den*, where *num* and *den* are the numerator and denominator of the aspect ratio. For example "4:3", "16:9", "1.3333", and "1.7777" are valid argument values.

If used together with **-vcodec copy**, it will affect the aspect ratio stored at container level, but not the aspect ratio stored in encoded frames, if it exists.

-display_rotation[:stream_specifier] rotation (input,per-stream)

Set video rotation metadata.

rotation is a decimal number specifying the amount in degree by which the video should be rotated counter-clockwise before being displayed.

This option overrides the rotation/display transform metadata stored in the file, if any. When the video is being transcoded (rather than copied) and "-autorotate" is enabled, the video will be rotated at the filtering stage. Otherwise, the metadata will be written into the output file if the muxer supports it.

If the "-display_hflip" and/or "-display_vflip" options are given, they are applied after the rotation specified by this option.

-display_hflip[:stream_specifier] (input,per-stream)

Set whether on display the image should be horizontally flipped.

See the "-display_rotation" option for more details.

-display_vflip[:stream_specifier] (input,per-stream)

Set whether on display the image should be vertically flipped.

See the "-display_rotation" option for more details.

-vn (input/output)

As an input option, blocks all video streams of a file from being filtered or being automatically selected or mapped for any output. See "-discard" option to disable streams individually.

As an output option, disables video recording i.e. automatic selection or mapping of any video stream. For full manual control see the "-map" option.

-vcodec *codec (output)*

Set the video codec. This is an alias for "-codec:v".

-pass*[:stream_specifier] n (output,per-stream)*

Select the pass number (1 or 2). It is used to do two-pass video encoding. The statistics of the video are recorded in the first pass into a log file (see also the option -passlogfile), and in the second pass that log file is used to generate the video at the exact requested bitrate. On pass 1, you may just deactivate audio and set output to null, examples for Windows and Unix:

```
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y NUL
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y /dev/null
```

-passlogfile*[:stream_specifier] prefix (output,per-stream)*

Set two-pass log file name prefix to *prefix*, the default file name prefix is "ffmpeg2pass". The complete file name will be *PREFIX-N.log*, where N is a number specific to the output stream

-vf *filtergraph (output)*

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

This is an alias for "-filter:v", see the **-filter option**.

-autorotate

Automatically rotate the video according to file metadata. Enabled by default, use **-noautorotate** to disable it.

-autoscale

Automatically scale the video according to the resolution of first frame. Enabled by default, use **-noautoscale** to disable it. When autoscale is disabled, all output frames of filter graph might not be in the same resolution and may be inadequate for some encoder/muxer. Therefore, it is not recommended to disable it unless you really know what you are doing. Disable autoscale at your own risk.

Advanced Video options

-pix_fmt*[:stream_specifier] format (input/output,per-stream)*

Set pixel format. Use "-pix_fmts" to show all the supported pixel formats. If the selected pixel format can not be selected, ffmpeg will print a warning and select the best pixel format supported by the encoder. If *pix_fmt* is prefixed by a "+", ffmpeg will exit with an error if the requested pixel format can not be selected, and automatic conversions inside filtergraphs are disabled. If *pix_fmt* is a single "+", ffmpeg selects the same pixel format as the input (or graph output) and automatic conversions are disabled.

-sws_flags *flags* (*input/output*)

Set SwScaler flags.

-rc_override[*:stream_specifier*] *override* (*output,per-stream*)

Rate control override for specific intervals, formatted as "int,int,int" list separated with slashes. Two first values are the beginning and end frame numbers, last one is quantizer to use if positive, or quality factor if negative.

-psnr

Calculate PSNR of compressed frames. This option is deprecated, pass the PSNR flag to the encoder instead, using "-flags +psnr".

-vstats

Dump video coding statistics to *vstats_HHMMSS.log*.

-vstats_file *file*

Dump video coding statistics to *file*.

-vstats_version *file*

Specifies which version of the vstats format to use. Default is 2.

version = 1 :

```
"frame= %5d q= %2.1f PSNR= %6.2f f_size= %6d s_size= %8.0fkB time= %0.3f br=
%7.1fkbits/s avg_br= %7.1fkbits/s"
```

version > 1:

```
"out= %2d st= %2d frame= %5d q= %2.1f PSNR= %6.2f f_size= %6d s_size= %8.0fkB time=
%0.3f br= %7.1fkbits/s avg_br= %7.1fkbits/s"
```

-top[*:stream_specifier*] *n* (*output,per-stream*)

top=1/bottom=0/auto=-1 field first

-vtag *fourcc/tag* (*output*)

Force video tag/fourcc. This is an alias for "-tag:v".

-qphist (*global*)

Show QP histogram

-vbsf *bitstream_filter*

Deprecated see -bsf

-force_key_frames[:*stream_specifier*] *time[,time...]* (*output,per-stream*)

-force_key_frames[:*stream_specifier*] **expr**:*expr* (*output,per-stream*)

-force_key_frames[:*stream_specifier*] **source** (*output,per-stream*)

-force_key_frames[:*stream_specifier*] **source_no_drop** (*output,per-stream*)

force_key_frames can take arguments of the following form:

time[,time...]

If the argument consists of timestamps, ffmpeg will round the specified times to the nearest output timestamp as per the encoder time base and force a keyframe at the first frame having timestamp equal or greater than the computed timestamp. Note that if the encoder time base is too coarse, then the keyframes may be forced on frames with timestamps lower than the specified time. The default encoder time base is the inverse of the output framerate but may be set otherwise via "-enc_time_base".

If one of the times is ""*chapters*"[*delta*]", it is expanded into the time of the beginning of all chapters in the file, shifted by *delta*, expressed as a time in seconds. This option can be useful to ensure that a seek point is present at a chapter mark or any other designated place in the output file.

For example, to insert a key frame at 5 minutes, plus key frames 0.1 second before the beginning of every chapter:

```
-force_key_frames 0:05:00,chapters-0.1
```

expr:*expr*

If the argument is prefixed with "expr:", the string *expr* is interpreted like an expression and is evaluated for each frame. A key frame is forced in case the evaluation is non-zero.

The expression in *expr* can contain the following constants:

n the number of current processed frame, starting from 0

n_forced

the number of forced frames

prev_forced_n

the number of the previous forced frame, it is "NAN" when no keyframe was forced yet

prev_forced_t

the time of the previous forced frame, it is "NAN" when no keyframe was forced yet

t the time of the current processed frame

For example to force a key frame every 5 seconds, you can specify:

```
-force_key_frames expr:gte(t,n_forced*5)
```

To force a key frame 5 seconds after the time of the last forced one, starting from second 13:

```
-force_key_frames expr:if(isnan(prev_forced_t),gte(t,13),gte(t,prev_forced_t+5))
```

source

If the argument is "source", ffmpeg will force a key frame if the current frame being encoded is marked as a key frame in its source.

source_no_drop

If the argument is "source_no_drop", ffmpeg will force a key frame if the current frame being encoded is marked as a key frame in its source. In cases where this particular source frame has to be dropped, enforce the next available frame to become a key frame instead.

Note that forcing too many keyframes is very harmful for the lookahead algorithms of certain encoders: using fixed-GOP options or similar would be more efficient.

-copyinkf[:stream_specifier] (*output,per-stream*)

When doing stream copy, copy also non-key frames found at the beginning.

-init_hw_device *type*[=*name*][:*device*[,*key=value*...]]

Initialise a new hardware device of type *type* called *name*, using the given device parameters. If no name is specified it will receive a default name of the form "*type*%d".

The meaning of *device* and the following arguments depends on the device type:

cuda

device is the number of the CUDA device.

The following options are recognized:

primary_ctx

If set to 1, uses the primary device context instead of creating a new one.

Examples:

-init_hw_device cuda:1

Choose the second device on the system.

-init_hw_device cuda:0,primary_ctx=1

Choose the first device and use the primary device context.

dxva2

device is the number of the Direct3D 9 display adapter.

d3d11va

device is the number of the Direct3D 11 display adapter.

vaapi

device is either an X11 display name or a DRM render node. If not specified, it will attempt to open the default X11 display (*\$DISPLAY*) and then the first DRM render node (*/dev/dri/renderD128*).

vdpa

device is an X11 display name. If not specified, it will attempt to open the default X11 display (*\$DISPLAY*).

qsv *device* selects a value in **MXF_IMPL_***. Allowed values are:

auto

sw

hw

auto_any

hw_any

hw2

hw3

hw4

If not specified, **auto_any** is used. (Note that it may be easier to achieve the desired result for QSV by creating the platform-appropriate subdevice (**dxva2** or **d3d11va** or **vaapi**) and then deriving a QSV device from that.)

Alternatively, **child_device_type** helps to choose platform-appropriate subdevice type. On Windows **d3d11va** is used as default subdevice type.

Examples:

```
-init_hw_device qsv:hw,child_device_type=d3d11va
```

Choose the GPU subdevice with type **d3d11va** and create QSV device with **MXF_IMPL_HARDWARE**.

```
-init_hw_device qsv:hw,child_device_type=dxva2
```

Choose the GPU subdevice with type **dxva2** and create QSV device with **MXF_IMPL_HARDWARE**.

openc1

device selects the platform and device as *platform_index.device_index*.

The set of devices can also be filtered using the key-value pairs to find only devices matching particular platform or device strings.

The strings usable as filters are:

platform_profile
platform_version
platform_name
platform_vendor
platform_extensions
device_name
device_vendor
driver_version
device_version
device_profile
device_extensions
device_type

The indices and filters must together uniquely select a device.

Examples:

-init_hw_device opencl:0.1

Choose the second device on the first platform.

-init_hw_device opencl:;device_name=Foo9000

Choose the device with a name containing the string *Foo9000*.

-init_hw_device opencl:1;device_type=gpu;device_extensions=cl_khr_fp16

Choose the GPU device on the second platform supporting the *cl_khr_fp16* extension.

vulkan

If *device* is an integer, it selects the device by its index in a system-dependent list of devices. If *device* is any other string, it selects the first device with a name containing that string as a substring.

The following options are recognized:

debug

If set to 1, enables the validation layer, if installed.

linear_images

If set to 1, images allocated by the hwcontext will be linear and locally mappable.

instance_extensions

A plus separated list of additional instance extensions to enable.

device_extensions

A plus separated list of additional device extensions to enable.

Examples:

-init_hw_device vulkan:1

Choose the second device on the system.

-init_hw_device vulkan:RADV

Choose the first device with a name containing the string *RADV*.

-init_hw_device

vulkan:0,instance_extensions=VK_KHR_wayland_surface+VK_KHR_xcb_surface

Choose the first device and enable the Wayland and XCB instance extensions.

-init_hw_device *type*[=*name*]@*source*

Initialise a new hardware device of type *type* called *name*, deriving it from the existing device with the name *source*.

-init_hw_device list

List all hardware device types supported in this build of ffmpeg.

-filter_hw_device *name*

Pass the hardware device called *name* to all filters in any filter graph. This can be used to set the device to upload to with the "hwupload" filter, or the device to map to with the "hwmap" filter. Other filters may also make use of this parameter when they require a hardware device. Note that this is typically only required when the input is not already in hardware frames - when it is, filters will derive the device they require from the context of the frames they receive as input.

This is a global setting, so all filters will receive the same device.

-hwaccel[:*stream_specifier*] *hwaccel* (*input,per-stream*)

Use hardware acceleration to decode the matching stream(s). The allowed values of *hwaccel* are:

none

Do not use any hardware acceleration (the default).

auto Automatically select the hardware acceleration method.

vdpa

Use VDPAAU (Video Decode and Presentation API for Unix) hardware acceleration.

dxva2

Use DXVA2 (DirectX Video Acceleration) hardware acceleration.

d3d11va

Use D3D11VA (DirectX Video Acceleration) hardware acceleration.

vaapi

Use VAAPI (Video Acceleration API) hardware acceleration.

qsv Use the Intel QuickSync Video acceleration for video transcoding.

Unlike most other values, this option does not enable accelerated decoding (that is used automatically whenever a qsv decoder is selected), but accelerated transcoding, without copying the frames into the system memory.

For it to work, both the decoder and the encoder must support QSV acceleration and no filters must be used.

This option has no effect if the selected hwaccel is not available or not supported by the chosen decoder.

Note that most acceleration methods are intended for playback and will not be faster than software decoding on modern CPUs. Additionally, **ffmpeg** will usually need to copy the decoded frames from the GPU memory into the system memory, resulting in further performance loss. This option is thus mainly useful for testing.

-hwaccel_device[:stream_specifier] hwaccel_device (*input,per-stream*)

Select a device to use for hardware acceleration.

This option only makes sense when the **-hwaccel** option is also specified. It can either refer to an existing device created with **-init_hw_device** by name, or it can create a new device as if **-init_hw_device type:hwaccel_device** were called immediately before.

-hwaccels

List all hardware acceleration components enabled in this build of ffmpeg. Actual runtime availability depends on the hardware and its suitable driver being installed.

-fix_sub_duration_heartbeat[:stream_specifier]

Set a specific output video stream as the heartbeat stream according to which to split and push through currently in-progress subtitle upon receipt of a random access packet.

This lowers the latency of subtitles for which the end packet or the following subtitle has not yet been received. As a drawback, this will most likely lead to duplication of subtitle events in order to cover the full duration, so when dealing with use cases where latency of when the subtitle event is passed on to output is not relevant this option should not be utilized.

Requires **-fix_sub_duration** to be set for the relevant input subtitle stream for this to have any effect, as well as for the input subtitle stream having to be directly mapped to the same output in which the heartbeat stream resides.

Audio Options

-aframes *number (output)*

Set the number of audio frames to output. This is an obsolete alias for "-frames:a", which you should use instead.

-ar*[:stream_specifier] freq (input/output,per-stream)*

Set the audio sampling frequency. For output streams it is set by default to the frequency of the corresponding input stream. For input streams this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

-aq *q (output)*

Set the audio quality (codec-specific, VBR). This is an alias for -q:a.

-ac*[:stream_specifier] channels (input/output,per-stream)*

Set the number of audio channels. For output streams it is set by default to the number of input audio channels. For input streams this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

-an *(input/output)*

As an input option, blocks all audio streams of a file from being filtered or being automatically selected or mapped for any output. See "-discard" option to disable streams individually.

As an output option, disables audio recording i.e. automatic selection or mapping of any audio stream. For full manual control see the "-map" option.

-acodec *codec (input/output)*

Set the audio codec. This is an alias for "-codec:a".

-sample_fmt*[:stream_specifier] sample_fmt (output,per-stream)*

Set the audio sample format. Use "-sample_fmts" to get a list of supported sample formats.

-af *filtergraph (output)*

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

This is an alias for "-filter:a", see the **-filter option**.

Advanced Audio options

-atag *fourcc/tag (output)*

Force audio tag/fourcc. This is an alias for "-tag:a".

-absf *bitstream_filter*

Deprecated, see -bsf

-guess_layout_max *channels (input,per-stream)*

If some input channel layout is not known, try to guess only if it corresponds to at most the specified number of channels. For example, 2 tells to **ffmpeg** to recognize 1 channel as mono and 2 channels as stereo but not 6 channels as 5.1. The default is to always try to guess. Use 0 to disable all guessing.

Subtitle options**-scodec** *codec (input/output)*

Set the subtitle codec. This is an alias for "-codec:s".

-sn *(input/output)*

As an input option, blocks all subtitle streams of a file from being filtered or being automatically selected or mapped for any output. See "-discard" option to disable streams individually.

As an output option, disables subtitle recording i.e. automatic selection or mapping of any subtitle stream. For full manual control see the "-map" option.

-sbsf *bitstream_filter*

Deprecated, see -bsf

Advanced Subtitle options**-fix_sub_duration**

Fix subtitles durations. For each subtitle, wait for the next packet in the same stream and adjust the duration of the first to avoid overlap. This is necessary with some subtitles codecs, especially DVB subtitles, because the duration in the original packet is only a rough estimate and the end is actually marked by an empty subtitle frame. Failing to use this option when necessary can result in exaggerated durations or muxing failures due to non-monotonic timestamps.

Note that this option will delay the output of all data until the next subtitle packet is decoded: it may increase memory consumption and latency a lot.

-canvas_size *size*

Set the size of the canvas used to render subtitles.

Advanced options**-map** [-]*input_file_id[:stream_specifier][?] | [linklabel] (output)*

Create one or more streams in the output file. This option has two forms for specifying the data

source(s): the first selects one or more streams from some input file (specified with "-i"), the second takes an output from some complex filtergraph (specified with "-filter_complex" or "-filter_complex_script").

In the first form, an output stream is created for every stream from the input file with the index *input_file_id*. If *stream_specifier* is given, only those streams that match the specifier are used (see the **Stream specifiers** section for the *stream_specifier* syntax).

A "-" character before the stream identifier creates a "negative" mapping. It disables matching streams from already created mappings.

A trailing "?" after the stream index will allow the map to be optional: if the map matches no streams the map will be ignored instead of failing. Note the map will still fail if an invalid input file index is used; such as if the map refers to a non-existent input.

An alternative [*linklabel*] form will map outputs from complex filter graphs (see the **-filter_complex** option) to the output file. *linklabel* must correspond to a defined output link label in the graph.

This option may be specified multiple times, each adding more streams to the output file. Any given input stream may also be mapped any number of times as a source for different output streams, e.g. in order to use different encoding options and/or filters. The streams are created in the output in the same order in which the "-map" options are given on the commandline.

Using this option disables the default mappings for this output file.

Examples:

map everything

To map ALL streams from the first input file to output

```
ffmpeg -i INPUT -map 0 output
```

select specific stream

If you have two audio streams in the first input file, these streams are identified by *0:0* and *0:1*. You can use "-map" to select which streams to place in an output file. For example:

```
ffmpeg -i INPUT -map 0:1 out.wav
```

will map the second input stream in *INPUT* to the (single) output stream in *out.wav*.

create multiple streams

To select the stream with index 2 from input file *a.mov* (specified by the identifier *0:2*), and stream with index 6 from input *b.mov* (specified by the identifier *1:6*), and copy them to the output file *out.mov*:

```
ffmpeg -i a.mov -i b.mov -c copy -map 0:2 -map 1:6 out.mov
```

create multiple streams 2

To select all video and the third audio stream from an input file:

```
ffmpeg -i INPUT -map 0:v -map 0:a:2 OUTPUT
```

negative map

To map all the streams except the second audio, use negative mappings

```
ffmpeg -i INPUT -map 0 -map -0:a:1 OUTPUT
```

optional map

To map the video and audio streams from the first input, and using the trailing "?", ignore the audio mapping if no audio streams exist in the first input:

```
ffmpeg -i INPUT -map 0:v -map 0:a? OUTPUT
```

map by language

To pick the English audio stream:

```
ffmpeg -i INPUT -map 0:m:language:eng OUTPUT
```

-ignore_unknown

Ignore input streams with unknown type instead of failing if copying such streams is attempted.

-copy_unknown

Allow input streams with unknown type to be copied instead of failing if copying such streams is attempted.

-map_channel [*input_file_id.stream_specifier.channel_id*|-1][?][:*output_file_id.stream_specifier*]

This option is deprecated and will be removed. It can be replaced by the *pan* filter. In some cases it may be easier to use some combination of the *channelsplit*, *channelmap*, or *amerge* filters.

Map an audio channel from a given input to an output. If *output_file_id.stream_specifier* is not

set, the audio channel will be mapped on all the audio streams.

Using "-1" instead of *input_file_id.stream_specifier.channel_id* will map a muted channel.

A trailing "?" will allow the *map_channel* to be optional: if the *map_channel* matches no channel the *map_channel* will be ignored instead of failing.

For example, assuming *INPUT* is a stereo audio file, you can switch the two audio channels with the following command:

```
ffmpeg -i INPUT -map_channel 0.0.1 -map_channel 0.0.0 OUTPUT
```

If you want to mute the first channel and keep the second:

```
ffmpeg -i INPUT -map_channel -1 -map_channel 0.0.1 OUTPUT
```

The order of the "-map_channel" option specifies the order of the channels in the output stream. The output channel layout is guessed from the number of channels mapped (mono if one "-map_channel", stereo if two, etc.). Using "-ac" in combination of "-map_channel" makes the channel gain levels to be updated if input and output channel layouts don't match (for instance two "-map_channel" options and "-ac 6").

You can also extract each channel of an input to specific outputs; the following command extracts two channels of the *INPUT* audio stream (file 0, stream 0) to the respective *OUTPUT_CH0* and *OUTPUT_CH1* outputs:

```
ffmpeg -i INPUT -map_channel 0.0.0 OUTPUT_CH0 -map_channel 0.0.1 OUTPUT_CH1
```

The following example splits the channels of a stereo input into two separate streams, which are put into the same output file:

```
ffmpeg -i stereo.wav -map 0:0 -map 0:0 -map_channel 0.0.0:0.0 -map_channel 0.0.1:0.1 -y out.ogg
```

Note that currently each output stream can only contain channels from a single input stream; you can't for example use "-map_channel" to pick multiple input audio channels contained in different streams (from the same or different files) and merge them into a single output stream. It is therefore not currently possible, for example, to turn two separate mono streams into a single stereo stream. However splitting a stereo stream into two single channel mono streams is possible.

If you need this feature, a possible workaround is to use the *amerge* filter. For example, if you

need to merge a media (here *input.mkv*) with 2 mono audio streams into one single stereo channel audio stream (and keep the video stream), you can use the following command:

```
ffmpeg -i input.mkv -filter_complex "[0:1] [0:2] amerge" -c:a pcm_s16le -c:v copy output.mkv
```

To map the first two audio channels from the first input, and using the trailing "?", ignore the audio channel mapping if the first input is mono instead of stereo:

```
ffmpeg -i INPUT -map_channel 0.0.0 -map_channel 0.0.1? OUTPUT
```

-map_metadata[:*metadata_spec_out*] *infile*[:*metadata_spec_in*] (*output,per-metadata*)

Set metadata information of the next output file from *infile*. Note that those are file indices (zero-based), not filenames. Optional *metadata_spec_in/out* parameters specify, which metadata to copy. A metadata specifier can have the following forms:

g global metadata, i.e. metadata that applies to the whole file

s[:*stream_spec*]

per-stream metadata. *stream_spec* is a stream specifier as described in the **Stream specifiers** chapter. In an input metadata specifier, the first matching stream is copied from. In an output metadata specifier, all matching streams are copied to.

c:*chapter_index*

per-chapter metadata. *chapter_index* is the zero-based chapter index.

p:*program_index*

per-program metadata. *program_index* is the zero-based program index.

If metadata specifier is omitted, it defaults to global.

By default, global metadata is copied from the first input file, per-stream and per-chapter metadata is copied along with streams/chapters. These default mappings are disabled by creating any mapping of the relevant type. A negative file index can be used to create a dummy mapping that just disables automatic copying.

For example to copy metadata from the first stream of the input file to global metadata of the output file:

```
ffmpeg -i in.ogg -map_metadata 0:s:0 out.mp3
```

To do the reverse, i.e. copy global metadata to all audio streams:

```
ffmpeg -i in.mkv -map_metadata:s:a 0:g out.mkv
```

Note that simple 0 would work as well in this example, since global metadata is assumed by default.

-map_chapters *input_file_index (output)*

Copy chapters from input file with index *input_file_index* to the next output file. If no chapter mapping is specified, then chapters are copied from the first input file with at least one chapter. Use a negative file index to disable any chapter copying.

-benchmark (*global*)

Show benchmarking information at the end of an encode. Shows real, system and user time used and maximum memory consumption. Maximum memory consumption is not supported on all systems, it will usually display as 0 if not supported.

-benchmark_all (*global*)

Show benchmarking information during the encode. Shows real, system and user time used in various steps (audio/video encode/decode).

-timelimit *duration (global)*

Exit after ffmpeg has been running for *duration* seconds in CPU user time.

-dump (*global*)

Dump each input packet to stderr.

-hex (*global*)

When dumping packets, also dump the payload.

-readrate *speed (input)*

Limit input read speed.

Its value is a floating-point positive number which represents the maximum duration of media, in seconds, that should be ingested in one second of wallclock time. Default value is zero and represents no imposed limitation on speed of ingestion. Value 1 represents real-time speed and is equivalent to "-re".

Mainly used to simulate a capture device or live input stream (e.g. when reading from a file). Should not be used with a low value when input is an actual capture device or live stream as it

may cause packet loss.

It is useful for when flow speed of output packets is important, such as live streaming.

-re (*input*)

Read input at native frame rate. This is equivalent to setting "-readrate 1".

-vsync *parameter (global)*

-fps_mode[:*stream_specifier*] *parameter (output,per-stream)*

Set video sync method / framerate mode. vsync is applied to all output video streams but can be overridden for a stream by setting fps_mode. vsync is deprecated and will be removed in the future.

For compatibility reasons some of the values for vsync can be specified as numbers (shown in parentheses in the following table).

passthrough (0)

Each frame is passed with its timestamp from the demuxer to the muxer.

cfr (1)

Frames will be duplicated and dropped to achieve exactly the requested constant frame rate.

vfr (2)

Frames are passed through with their timestamp or dropped so as to prevent 2 frames from having the same timestamp.

drop

As passthrough but destroys all timestamps, making the muxer generate fresh timestamps based on frame-rate.

auto (-1)

Chooses between cfr and vfr depending on muxer capabilities. This is the default method.

Note that the timestamps may be further modified by the muxer, after this. For example, in the case that the format option **avoid_negative_ts** is enabled.

With -map you can select from which stream the timestamps should be taken. You can leave either video or audio unchanged and sync the remaining stream(s) to the unchanged one.

-frame_drop_threshold *parameter*

Frame drop threshold, which specifies how much behind video frames can be before they are dropped. In frame rate units, so 1.0 is one frame. The default is -1.1. One possible usecase is to avoid framedrops in case of noisy timestamps or to increase frame drop precision in case of exact timestamps.

-adrift_threshold *time*

Set the minimum difference between timestamps and audio data (in seconds) to trigger adding/dropping samples to make it match the timestamps. This option effectively is a threshold to select between hard (add/drop) and soft (squeeze/stretch) compensation. "-async" must be set to a positive value.

-apad *parameters (output,per-stream)*

Pad the output audio stream(s). This is the same as applying "-af apad". Argument is a string of filter parameters composed the same as with the "apad" filter. "-shortest" must be set for this output for the option to take effect.

-copyts

Do not process input timestamps, but keep their values without trying to sanitize them. In particular, do not remove the initial start time offset value.

Note that, depending on the **vsync** option or on specific muxer processing (e.g. in case the format option **avoid_negative_ts** is enabled) the output timestamps may mismatch with the input timestamps even when this option is selected.

-start_at_zero

When used with **copyts**, shift input timestamps so they start at zero.

This means that using e.g. "-ss 50" will make output timestamps start at 50 seconds, regardless of what timestamp the input file started at.

-copytb *mode*

Specify how to set the encoder timebase when stream copying. *mode* is an integer numeric value, and can assume one of the following values:

- 1 Use the demuxer timebase.

The time base is copied to the output encoder from the corresponding input demuxer. This is sometimes required to avoid non monotonically increasing timestamps when copying video streams with variable frame rate.

- 0** Use the decoder timebase.

The time base is copied to the output encoder from the corresponding input decoder.

- 1** Try to make the choice automatically, in order to generate a sane output.

Default value is -1.

-enc_time_base[:*stream_specifier*] *timebase* (*output,per-stream*)

Set the encoder timebase. *timebase* is a floating point number, and can assume one of the following values:

- 0** Assign a default value according to the media type.

For video - use 1/framerate, for audio - use 1/samplerate.

- 1** Use the input stream timebase when possible.

If an input stream is not available, the default timebase will be used.

- >0** Use the provided number as the timebase.

This field can be provided as a ratio of two integers (e.g. 1:24, 1:48000) or as a floating point number (e.g. 0.04166, 2.0833e-5)

Default value is 0.

-bitexact (*input/output*)

Enable bitexact mode for (de)muxer and (de/en)coder

-shortest (*output*)

Finish encoding when the shortest output stream ends.

Note that this option may require buffering frames, which introduces extra latency. The maximum amount of this latency may be controlled with the "-shortest_buf_duration" option.

-shortest_buf_duration *duration* (*output*)

The "-shortest" option may require buffering potentially large amounts of data when at least one of the streams is "sparse" (i.e. has large gaps between frames X this is typically the case for subtitles).

This option controls the maximum duration of buffered frames in seconds. Larger values may allow the "-shortest" option to produce more accurate results, but increase memory use and latency.

The default value is 10 seconds.

-dts_delta_threshold

Timestamp discontinuity delta threshold.

-dts_error_threshold *seconds*

Timestamp error delta threshold. This threshold use to discard crazy/damaged timestamps and the default is 30 hours which is arbitrarily picked and quite conservative.

-muxdelay *seconds (output)*

Set the maximum demux-decode delay.

-muxpreload *seconds (output)*

Set the initial demux-decode delay.

-streamid *output-stream-index:new-value (output)*

Assign a new stream-id value to an output stream. This option should be specified prior to the output filename to which it applies. For the situation where multiple output files exist, a streamid may be reassigned to a different value.

For example, to set the stream 0 PID to 33 and the stream 1 PID to 36 for an output mpegts file:

```
ffmpeg -i inurl -streamid 0:33 -streamid 1:36 out.ts
```

-bsf[:stream_specifier] *bitstream_filters (output,per-stream)*

Set bitstream filters for matching streams. *bitstream_filters* is a comma-separated list of bitstream filters. Use the "-bsfs" option to get the list of bitstream filters.

```
ffmpeg -i h264.mp4 -c:v copy -bsf:v h264_mp4toannexb -an out.h264
```

```
ffmpeg -i file.mov -an -vn -bsf:s mov2textsub -c:s copy -f rawvideo sub.txt
```

-tag[:stream_specifier] *codec_tag (input/output,per-stream)*

Force a tag/fourcc for matching streams.

-timecode *hh:mm:ssSEPff*

Specify Timecode for writing. *SEP* is ':' for non drop timecode and ';' (or '.' for drop).

```
ffmpeg -i input.mpg -timecode 01:02:03.04 -r 30000/1001 -s ntsc output.mpg
```

-filter_complex *filtergraph (global)*

Define a complex filtergraph, i.e. one with arbitrary number of inputs and/or outputs. For simple graphs -- those with one input and one output of the same type -- see the **-filter** options. *filtergraph* is a description of the filtergraph, as described in the "Filtergraph syntax" section of the ffmpeg-filters manual.

Input link labels must refer to input streams using the "[file_index:stream_specifier]" syntax (i.e. the same as **-map** uses). If *stream_specifier* matches multiple streams, the first one will be used. An unlabeled input will be connected to the first unused input stream of the matching type.

Output link labels are referred to with **-map**. Unlabeled outputs are added to the first output file.

Note that with this option it is possible to use only lavfi sources without normal input files.

For example, to overlay an image over video

```
ffmpeg -i video.mkv -i image.png -filter_complex '[0:v][1:v]overlay[out]' -map
[out] out.mkv
```

Here "[0:v]" refers to the first video stream in the first input file, which is linked to the first (main) input of the overlay filter. Similarly the first video stream in the second input is linked to the second (overlay) input of overlay.

Assuming there is only one video stream in each input file, we can omit input labels, so the above is equivalent to

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay[out]' -map
[out] out.mkv
```

Furthermore we can omit the output label and the single output from the filter graph will be added to the output file automatically, so we can simply write

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay' out.mkv
```

As a special exception, you can use a bitmap subtitle stream as input: it will be converted into a

video with the same size as the largest video in the file, or 720x576 if no video is present. Note that this is an experimental and temporary solution. It will be removed once libavfilter has proper support for subtitles.

For example, to hardcode subtitles on top of a DVB-T recording stored in MPEG-TS format, delaying the subtitles by 1 second:

```
ffmpeg -i input.ts -filter_complex \
    '[#0x2ef] setpts=PTS+1/TB [sub] ; [#0x2d0] [sub] overlay' \
    -sn -map '#0x2dc' output.mkv
```

(0x2d0, 0x2dc and 0x2ef are the MPEG-TS PIDs of respectively the video, audio and subtitles streams; 0:0, 0:3 and 0:7 would have worked too)

To generate 5 seconds of pure red video using lavfi "color" source:

```
ffmpeg -filter_complex 'color=c=red' -t 5 out.mkv
```

-filter_complex_threads *nb_threads* (*global*)

Defines how many threads are used to process a filter_complex graph. Similar to filter_threads but used for "-filter_complex" graphs only. The default is the number of available CPUs.

-lavfi *filtergraph* (*global*)

Define a complex filtergraph, i.e. one with arbitrary number of inputs and/or outputs. Equivalent to **-filter_complex**.

-filter_complex_script *filename* (*global*)

This option is similar to **-filter_complex**, the only difference is that its argument is the name of the file from which a complex filtergraph description is to be read.

-accurate_seek (*input*)

This option enables or disables accurate seeking in input files with the **-ss** option. It is enabled by default, so seeking is accurate when transcoding. Use **-noaccurate_seek** to disable it, which may be useful e.g. when copying some streams and transcoding the others.

-seek_timestamp (*input*)

This option enables or disables seeking by timestamp in input files with the **-ss** option. It is disabled by default. If enabled, the argument to the **-ss** option is considered an actual timestamp, and is not offset by the start time of the file. This matters only for files which do not start from timestamp 0, such as transport streams.

-thread_queue_size *size (input/output)*

For input, this option sets the maximum number of queued packets when reading from the file or device. With low latency / high rate live streams, packets may be discarded if they are not read in a timely manner; setting this value can force ffmpeg to use a separate input thread and read packets as soon as they arrive. By default ffmpeg only does this if multiple inputs are specified.

For output, this option specified the maximum number of packets that may be queued to each muxing thread.

-sdp_file *file (global)*

Print sdp information for an output stream to *file*. This allows dumping sdp information when at least one output isn't an rtp stream. (Requires at least one of the output formats to be rtp).

-discard (*input*)

Allows discarding specific streams or frames from streams. Any input stream can be fully discarded, using value "all" whereas selective discarding of frames from a stream occurs at the demuxer and is not supported by all demuxers.

none

Discard no frame.

default

Default, which discards no frames.

noref

Discard all non-reference frames.

bidir

Discard all bidirectional frames.

nokey

Discard all frames excepts keyframes.

all Discard all frames.

-abort_on *flags (global)*

Stop and abort on various conditions. The following flags are available:

empty_output

No packets were passed to the muxer, the output is empty.

empty_output_stream

No packets were passed to the muxer in some of the output streams.

-max_error_rate (*global*)

Set fraction of decoding frame failures across all inputs which when crossed ffmpeg will return exit code 69. Crossing this threshold does not terminate processing. Range is a floating-point number between 0 to 1. Default is 2/3.

-xerror (*global*)

Stop and exit on error

-max_muxing_queue_size *packets* (*output,per-stream*)

When transcoding audio and/or video streams, ffmpeg will not begin writing into the output until it has one packet for each such stream. While waiting for that to happen, packets for other streams are buffered. This option sets the size of this buffer, in packets, for the matching output stream.

The default value of this option should be high enough for most uses, so only touch this option if you are sure that you need it.

-muxing_queue_data_threshold *bytes* (*output,per-stream*)

This is a minimum threshold until which the muxing queue size is not taken into account. Defaults to 50 megabytes per stream, and is based on the overall size of packets passed to the muxer.

-auto_conversion_filters (*global*)

Enable automatically inserting format conversion filters in all filter graphs, including those defined by **-vf**, **-af**, **-filter_complex** and **-lavfi**. If filter format negotiation requires a conversion, the initialization of the filters will fail. Conversions can still be performed by inserting the relevant conversion filter (scale, aresample) in the graph. On by default, to explicitly disable it you need to specify "-noauto_conversion_filters".

-bits_per_raw_sample[:stream_specifier] *value* (*output,per-stream*)

Declare the number of bits per raw sample in the given output stream to be *value*. Note that this option sets the information provided to the encoder/muxer, it does not change the stream to conform to this value. Setting values that do not match the stream properties may result in encoding failures or invalid output files.

-stats_enc_pre[:stream_specifier] *path* (*output,per-stream*)**-stats_enc_post[:stream_specifier]** *path* (*output,per-stream*)**-stats_mux_pre[:stream_specifier]** *path* (*output,per-stream*)

Write per-frame encoding information about the matching streams into the file given by *path*.

-stats_enc_pre writes information about raw video or audio frames right before they are sent for encoding, while **-stats_enc_post** writes information about encoded packets as they are received from the encoder. **-stats_mux_pre** writes information about packets just as they are about to be sent to the muxer. Every frame or packet produces one line in the specified file. The format of this line is controlled by **-stats_enc_pre_fmt** / **-stats_enc_post_fmt** / **-stats_mux_pre_fmt**.

When stats for multiple streams are written into a single file, the lines corresponding to different streams will be interleaved. The precise order of this interleaving is not specified and not guaranteed to remain stable between different invocations of the program, even with the same options.

-stats_enc_pre_fmt[:stream_specifier] *format_spec* (*output,per-stream*)

-stats_enc_post_fmt[:stream_specifier] *format_spec* (*output,per-stream*)

-stats_mux_pre_fmt[:stream_specifier] *format_spec* (*output,per-stream*)

Specify the format for the lines written with **-stats_enc_pre** / **-stats_enc_post** / **-stats_mux_pre**.

format_spec is a string that may contain directives of the form *{fmt}*. *format_spec* is backslash-escaped --- use `\{`, `\}`, and `\\` to write a literal `{`, `}`, or `\`, respectively, into the output.

The directives given with *fmt* may be one of the following:

fidx Index of the output file.

sidx Index of the output stream in the file.

n Frame number. Pre-encoding: number of frames sent to the encoder so far. Post-encoding: number of packets received from the encoder so far. Muxing: number of packets submitted to the muxer for this stream so far.

ni Input frame number. Index of the input frame (i.e. output by a decoder) that corresponds to this output frame or packet. -1 if unavailable.

tb Encoder timebase, as a rational number *num/den*. Note that this may be different from the timebase used by the muxer.

tbi Timebase for *ptsi*, as a rational number *num/den*. Available when *ptsi* is available, *0/1* otherwise.

pts Presentation timestamp of the frame or packet, as an integer. Should be multiplied by the timebase to compute presentation time.

- pts** Presentation timestamp of the input frame (see *ni*), as an integer. Should be multiplied by *tbi* to compute presentation time. Printed as $(2^{63} - 1 = 9223372036854775807)$ when not available.
- t** Presentation time of the frame or packet, as a decimal number. Equal to *pts* multiplied by *tb*.
- ti** Presentation time of the input frame (see *ni*), as a decimal number. Equal to *pts* multiplied by *tbi*. Printed as *inf* when not available.
- dts** Decoding timestamp of the packet, as an integer. Should be multiplied by the timebase to compute presentation time. Post-encoding only.
- dt** Decoding time of the frame or packet, as a decimal number. Equal to *dts* multiplied by *tb*.
- sn** Number of audio samples sent to the encoder so far. Audio and pre-encoding only.
- samp**
Number of audio samples in the frame. Audio and pre-encoding only.
- size** Size of the encoded packet in bytes. Post-encoding only.
- br** Current bitrate in bits per second. Post-encoding only.
- abr** Average bitrate for the whole stream so far, in bits per second, -1 if it cannot be determined at this point. Post-encoding only.

The default format strings are:

pre-encoding

{fidx} {sidx} {n} {t}

post-encoding

{fidx} {sidx} {n} {t}

In the future, new items may be added to the end of the default formatting strings. Users who depend on the format staying exactly the same, should prescribe it manually.

Note that stats for different streams written into the same file may have different formats.

Preset files

A preset file contains a sequence of *option=value* pairs, one for each line, specifying a sequence of options which would be awkward to specify on the command line. Lines starting with the hash ('#') character are ignored and are used to provide comments. Check the *presets* directory in the FFmpeg source tree for examples.

There are two types of preset files: *ffpreset* and *avpreset* files.

ffpreset files

ffpreset files are specified with the "vpre", "apre", "spre", and "fpre" options. The "fpre" option takes the filename of the preset instead of a preset name as input and can be used for any kind of codec. For the "vpre", "apre", and "spre" options, the options specified in a preset file are applied to the currently selected codec of the same type as the preset option.

The argument passed to the "vpre", "apre", and "spre" preset options identifies the preset file to use according to the following rules:

First *ffmpeg* searches for a file named *arg.ffpreset* in the directories *\$FFMPEG_DATADIR* (if set), and *\$HOME/.ffmpeg*, and in the *datadir* defined at configuration time (usually *PREFIX/share/ffmpeg*) or in a *ffpresets* folder along the executable on win32, in that order. For example, if the argument is "libvpx-1080p", it will search for the file *libvpx-1080p.ffpreset*.

If no such file is found, then *ffmpeg* will search for a file named *codec_name-arg.ffpreset* in the above-mentioned directories, where *codec_name* is the name of the codec to which the preset file options will be applied. For example, if you select the video codec with "-vcodec libvpx" and use "-vpre 1080p", then it will search for the file *libvpx-1080p.ffpreset*.

avpreset files

avpreset files are specified with the "pre" option. They work similar to *ffpreset* files, but they only allow encoder-specific options. Therefore, an *option=value* pair specifying an encoder cannot be used.

When the "pre" option is specified, *ffmpeg* will look for files with the suffix *.avpreset* in the directories *\$AVCONV_DATADIR* (if set), and *\$HOME/.avconv*, and in the *datadir* defined at configuration time (usually *PREFIX/share/ffmpeg*), in that order.

First *ffmpeg* searches for a file named *codec_name-arg.avpreset* in the above-mentioned directories, where *codec_name* is the name of the codec to which the preset file options will be applied. For example, if you select the video codec with "-vcodec libvpx" and use "-pre 1080p", then it will search for the file *libvpx-1080p.avpreset*.

If no such file is found, then ffmpeg will search for a file named *arg.avpreset* in the same directories.

EXAMPLES

Video and Audio grabbing

If you specify the input format and device then ffmpeg can grab video and audio directly.

```
ffmpeg -f oss -i /dev/dsp -f video4linux2 -i /dev/video0 /tmp/out.mpg
```

Or with an ALSA audio source (mono input, card id 1) instead of OSS:

```
ffmpeg -f alsa -ac 1 -i hw:1 -f video4linux2 -i /dev/video0 /tmp/out.mpg
```

Note that you must activate the right video source and channel before launching ffmpeg with any TV viewer such as <http://linux.bytesex.org/xawtv/> by Gerd Knorr. You also have to set the audio recording levels correctly with a standard mixer.

X11 grabbing

Grab the X11 display with ffmpeg via

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0 /tmp/out.mpg
```

0.0 is display.screen number of your X11 server, same as the DISPLAY environment variable.

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0+10,20 /tmp/out.mpg
```

0.0 is display.screen number of your X11 server, same as the DISPLAY environment variable. 10 is the x-offset and 20 the y-offset for the grabbing.

Video and Audio file format conversion

Any supported file format and protocol can serve as input to ffmpeg:

Examples:

- ⊕ You can use YUV files as input:

```
ffmpeg -i /tmp/test%d.Y /tmp/out.mpg
```

It will use the files:

```
/tmp/test0.Y, /tmp/test0.U, /tmp/test0.V,
```

/tmp/test1.Y, /tmp/test1.U, /tmp/test1.V, etc...

The Y files use twice the resolution of the U and V files. They are raw files, without header. They can be generated by all decent video decoders. You must specify the size of the image with the `-s` option if ffmpeg cannot guess it.

- ⊕ You can input from a raw YUV420P file:

```
ffmpeg -i /tmp/test.yuv /tmp/out.avi
```

test.yuv is a file containing raw YUV planar data. Each frame is composed of the Y plane followed by the U and V planes at half vertical and horizontal resolution.

- ⊕ You can output to a raw YUV420P file:

```
ffmpeg -i mydivx.avi hugefile.yuv
```

- ⊕ You can set several input files and output files:

```
ffmpeg -i /tmp/a.wav -s 640x480 -i /tmp/a.yuv /tmp/a.mpg
```

Converts the audio file a.wav and the raw YUV video file a.yuv to MPEG file a.mpg.

- ⊕ You can also do audio and video conversions at the same time:

```
ffmpeg -i /tmp/a.wav -ar 22050 /tmp/a.mp2
```

Converts a.wav to MPEG audio at 22050 Hz sample rate.

- ⊕ You can encode to several formats at the same time and define a mapping from input stream to output streams:

```
ffmpeg -i /tmp/a.wav -map 0:a -b:a 64k /tmp/a.mp2 -map 0:a -b:a 128k /tmp/b.mp2
```

Converts a.wav to a.mp2 at 64 kbits and to b.mp2 at 128 kbits. '-map file:index' specifies which input stream is used for each output stream, in the order of the definition of output streams.

- ⊕ You can transcode decrypted VOBs:

```
ffmpeg -i snatch_1.vob -f avi -c:v mpeg4 -b:v 800k -g 300 -bf 2 -c:a libmp3lame -b:a 128k snatch.avi
```

This is a typical DVD ripping example; the input is a VOB file, the output an AVI file with MPEG-4 video and MP3 audio. Note that in this command we use B-frames so the MPEG-4 stream is DivX5 compatible, and GOP size is 300 which means one intra frame every 10 seconds for 29.97fps input video. Furthermore, the audio stream is MP3-encoded so you need to enable LAME support by passing "--enable-libmp3lame" to configure. The mapping is particularly useful for DVD transcoding to get the desired audio language.

NOTE: To see the supported input formats, use "ffmpeg -demuxers".

- ⊕ You can extract images from a video, or create a video from many images:

For extracting images from a video:

```
ffmpeg -i foo.avi -r 1 -s WxH -f image2 foo-%03d.jpeg
```

This will extract one video frame per second from the video and will output them in files named *foo-001.jpeg*, *foo-002.jpeg*, etc. Images will be rescaled to fit the new WxH values.

If you want to extract just a limited number of frames, you can use the above command in combination with the "-frames:v" or "-t" option, or in combination with -ss to start extracting from a certain point in time.

For creating a video from many images:

```
ffmpeg -f image2 -framerate 12 -i foo-%03d.jpeg -s WxH foo.avi
```

The syntax "foo-%03d.jpeg" specifies to use a decimal number composed of three digits padded with zeroes to express the sequence number. It is the same syntax supported by the C printf function, but only formats accepting a normal integer are suitable.

When importing an image sequence, -i also supports expanding shell-like wildcard patterns (globbing) internally, by selecting the image2-specific "-pattern_type glob" option.

For example, for creating a video from filenames matching the glob pattern "foo-*.jpeg":

```
ffmpeg -f image2 -pattern_type glob -framerate 12 -i 'foo-*.jpeg' -s WxH foo.avi
```

- ⊕ You can put many streams of the same type in the output:

```
ffmpeg -i test1.avi -i test2.avi -map 1:1 -map 1:0 -map 0:1 -map 0:0 -c copy -y test12.nut
```

The resulting output file *test12.nut* will contain the first four streams from the input files in reverse order.

- ⊕ To force CBR video output:

```
ffmpeg -i myfile.avi -b 4000k -minrate 4000k -maxrate 4000k -bufsize 1835k out.m2v
```

- ⊕ The four options *lmin*, *lmax*, *mblmin* and *mblmax* use 'lambda' units, but you may use the *QP2LAMBDA* constant to easily convert from 'q' units:

```
ffmpeg -i src.ext -lmax 21*QP2LAMBDA dst.ext
```

SYNTAX

This section documents the syntax and formats employed by the FFmpeg libraries and tools.

Quoting and escaping

FFmpeg adopts the following quoting and escaping mechanism, unless explicitly specified. The following rules are applied:

- ⊕ ' and \ are special characters (respectively used for quoting and escaping). In addition to them, there might be other special characters depending on the specific syntax where the escaping and quoting are employed.
- ⊕ A special character is escaped by prefixing it with a \.
- ⊕ All characters enclosed between "" are included literally in the parsed string. The quote character ' itself cannot be quoted, so you may need to close the quote and escape it.
- ⊕ Leading and trailing whitespaces, unless escaped or quoted, are removed from the parsed string.

Note that you may need to add a second level of escaping when using the command line or a script, which depends on the syntax of the adopted shell language.

The function "av_get_token" defined in *libavutil/avstring.h* can be used to parse a token quoted or escaped according to the rules defined above.

The tool *tools/ffescape* in the FFmpeg source tree can be used to automatically quote or escape a string in a script.

Examples

- ⊕ Escape the string "Crime d'Amour" containing the "'" special character:

Crime d\’Amour

- ⊕ The string above contains a quote, so the "" needs to be escaped when quoting it:

’Crime d\’Amour’

- ⊕ Include leading or trailing whitespaces using quoting:

’ this string starts and ends with whitespaces ’

- ⊕ Escaping and quoting can be mixed together:

’ The string \’string\’ is a string ’

- ⊕ To include a literal \ you can use either escaping or quoting:

’c:\foo’ can be written as c:\\foo

Date

The accepted syntax is:

```
[(YYYY-MM-DD|YYYYMMDD)[T|t ]][(HH:MM:SS[.m...])|(HHMMSS[.m...])][Z]
now
```

If the value is "now" it takes the current time.

Time is local time unless Z is appended, in which case it is interpreted as UTC. If the year-month-day part is not specified it takes the current year-month-day.

Time duration

There are two accepted syntaxes for expressing time duration.

```
[-][<HH>:<MM>:<SS>[.<m>...]
```

HH expresses the number of hours, *MM* the number of minutes for a maximum of 2 digits, and *SS* the number of seconds for a maximum of 2 digits. The *m* at the end expresses decimal value for *SS*.

or

`[-]<S>+[.<m>...][s|ms|us]`

S expresses the number of seconds, with the optional decimal part *m*. The optional literal suffixes **s**, **ms** or **us** indicate to interpret the value as seconds, milliseconds or microseconds, respectively.

In both expressions, the optional - indicates negative duration.

Examples

The following examples are all valid time duration:

55 55 seconds

0.2 0.2 seconds

200ms

200 milliseconds, that's 0.2s

200000us

200000 microseconds, that's 0.2s

12:03:45

12 hours, 03 minutes and 45 seconds

23.189

23.189 seconds

Video size

Specify the size of the sourced video, it may be a string of the form *widthxheight*, or the name of a size abbreviation.

The following abbreviations are recognized:

ntsc 720x480

pal 720x576

qntsc

352x240

qpai 352x288

sntsc

640x480

spal 768x576

film 352x240

ntsc-film

352x240

sqcif

128x96

qcif 176x144

cif 352x288

4cif 704x576

16cif

1408x1152

qqvga

160x120

qvga

320x240

vga 640x480

svga

800x600

xga 1024x768

uxga

1600x1200

qxga

2048x1536

sxga

1280x1024

qsxga

2560x2048

hsxga

5120x4096

wvga

852x480

wxga

1366x768

wsxga

1600x1024

wuxga

1920x1200

woxga

2560x1600

wqsxga

3200x2048

wquxga

3840x2400

whsxga

6400x4096

whuxga

7680x4800

cga 320x200

ega 640x350

hd480

852x480

hd720

1280x720

hd1080

1920x1080

2k 2048x1080

2kflat

1998x1080

2kscope

2048x858

4k 4096x2160

4kflat

3996x2160

4kscope

4096x1716

nhd 640x360

hqvga

240x160

wqvga

400x240

fwqvga

432x240

hvga

480x320

qhd 960x540

2kdc

2048x1080

4kdc

4096x2160

uhd2160

3840x2160

uhd4320

7680x4320

Video rate

Specify the frame rate of a video, expressed as the number of frames generated per second. It has to be a string in the format *frame_rate_num/frame_rate_den*, an integer number, a float number or a valid video frame rate abbreviation.

The following abbreviations are recognized:

ntsc 30000/1001

pal 25/1

qntsc

30000/1001

qpal 25/1

sntsc

30000/1001

spal 25/1

film 24/1

ntsc-film

24000/1001

Ratio

A ratio can be expressed as an expression, or in the form *numerator:denominator*.

Note that a ratio with infinite (1/0) or negative value is considered valid, so you should check on the returned value if you want to exclude those values.

The undefined value can be expressed using the "0:0" string.

Color

It can be the name of a color as defined below (case insensitive match) or a "[0x#]RRGGBB[AA]" sequence, possibly followed by @ and a string representing the alpha component.

The alpha component may be a string composed by "0x" followed by an hexadecimal number or a decimal number between 0.0 and 1.0, which represents the opacity value (**0x00** or **0.0** means completely transparent, **0xff** or **1.0** completely opaque). If the alpha component is not specified then **0xff** is assumed.

The string **random** will result in a random color.

The following names of colors are recognized:

AliceBlue

0xF0F8FF

AntiqueWhite

0xFAEBD7

Aqua

0x00FFFF

Aquamarine

0x7FFFD4

Azure

0xF0FFFF

Beige

0xF5F5DC

Bisque

0xFFE4C4

Black

0x000000

BlanchedAlmond

0xFFEBCD

Blue

0x0000FF

BlueViolet

0x8A2BE2

Brown

0xA52A2A

BurlyWood

0xDEB887

CadetBlue

0x5F9EA0

Chartreuse

0x7FFF00

Chocolate

0xD2691E

Coral

0xFF7F50

CornflowerBlue

0x6495ED

Cornsilk

0xFFFF8DC

Crimson

0xDC143C

Cyan

0x00FFFF

DarkBlue

0x00008B

DarkCyan

0x008B8B

DarkGoldenRod

0xB8860B

DarkGray

0xA9A9A9

DarkGreen

0x006400

DarkKhaki

0xBDB76B

DarkMagenta

0x8B008B

DarkOliveGreen

0x556B2F

Darkorange

0xFF8C00

DarkOrchid

0x9932CC

DarkRed

0x8B0000

DarkSalmon

0xE9967A

DarkSeaGreen

FFMPEG-ALL(1)

0x8FBC8F

DarkSlateBlue

0x483D8B

DarkSlateGray

0x2F4F4F

DarkTurquoise

0x00CED1

DarkViolet

0x9400D3

DeepPink

0xFF1493

DeepSkyBlue

0x00BFFF

DimGray

0x696969

DodgerBlue

0x1E90FF

FireBrick

0xB22222

FloralWhite

0xFFFFAF0

ForestGreen

0x228B22

Fuchsia

0xFF00FF

Gainsboro

0xDCDCDC

FFMPEG-ALL(1)

FFMPEG-ALL(1)

GhostWhite

0xF8F8FF

Gold

0xFFD700

GoldenRod

0xDAA520

Gray

0x808080

Green

0x008000

GreenYellow

0xADFF2F

HoneyDew

0xF0FFF0

HotPink

0xFF69B4

IndianRed

0xCD5C5C

Indigo

0x4B0082

Ivory

0xFFFFF0

Khaki

0xF0E68C

Lavender

0xE6E6FA

LavenderBlush

0xFFFF0F5

LawnGreen

0x7CFC00

LemonChiffon

0xFFFFACD

LightBlue

0xADD8E6

LightCoral

0xF08080

LightCyan

0xE0FFFF

LightGoldenRodYellow

0xFAFAD2

LightGreen

0x90EE90

LightGrey

0xD3D3D3

LightPink

0xFFB6C1

LightSalmon

0xFFA07A

LightSeaGreen

0x20B2AA

LightSkyBlue

0x87CEFA

LightSlateGray

0x778899

LightSteelBlue
0xB0C4DE

LightYellow
0xFFFFE0

Lime
0x00FF00

LimeGreen
0x32CD32

Linen
0xFAF0E6

Magenta
0xFF00FF

Maroon
0x800000

MediumAquaMarine
0x66CDAA

MediumBlue
0x0000CD

MediumOrchid
0xBA55D3

MediumPurple
0x9370D8

MediumSeaGreen
0x3CB371

MediumSlateBlue
0x7B68EE

MediumSpringGreen

0x00FA9A

MediumTurquoise

0x48D1CC

MediumVioletRed

0xC71585

MidnightBlue

0x191970

MintCream

0xF5FFFA

MistyRose

0xFFE4E1

Moccasin

0xFFE4B5

NavajoWhite

0xFFDEAD

Navy

0x000080

OldLace

0xFDF5E6

Olive

0x808000

OliveDrab

0x6B8E23

Orange

0xFFA500

OrangeRed

0xFF4500

Orchid

0xDA70D6

PaleGoldenRod

0xEEEE8AA

PaleGreen

0x98FB98

PaleTurquoise

0xAFEEEE

PaleVioletRed

0xD87093

PapayaWhip

0xFFEFD5

PeachPuff

0xFFDAB9

Peru

0xCD853F

Pink

0xFFC0CB

Plum

0xDDA0DD

PowderBlue

0xB0E0E6

Purple

0x800080

Red 0xFF0000

RosyBrown

0xBC8F8F

RoyalBlue

0x4169E1

SaddleBrown

0x8B4513

Salmon

0xFA8072

SandyBrown

0xF4A460

SeaGreen

0x2E8B57

SeaShell

0xFFF5EE

Sienna

0xA0522D

Silver

0xC0C0C0

SkyBlue

0x87CEEB

SlateBlue

0x6A5ACD

SlateGray

0x708090

Snow

0xFFFAFA

SpringGreen

0x00FF7F

SteelBlue

0x4682B4

Tan 0xD2B48C

Teal

0x008080

Thistle

0xD8BFD8

Tomato

0xFF6347

Turquoise

0x40E0D0

Violet

0xEE82EE

Wheat

0xF5DEB3

White

0xFFFFFFFF

WhiteSmoke

0xF5F5F5

Yellow

0xFFFF00

YellowGreen

0x9ACD32

Channel Layout

A channel layout specifies the spatial disposition of the channels in a multi-channel audio stream. To specify a channel layout, Ffmpeg makes use of a special syntax.

Individual channels are identified by an id, as given by the table below:

FL front left

FR front right

FC front center

LFE
low frequency

BL back left

BR back right

FLC
front left-of-center

FRC
front right-of-center

BC back center

SL side left

SR side right

TC top center

TFL
top front left

TFC
top front center

TFR
top front right

TBL
top back left

TBC

top back center

TBR

top back right

DL downmix left

DR downmix right

WL wide left

WR wide right

SDL

surround direct left

SDR

surround direct right

LFE2

low frequency 2

Standard channel layout compositions can be specified by using the following identifiers:

mono

FC

stereo

FL+FR

2.1 FL+FR+LFE

3.0 FL+FR+FC

3.0(back)

FL+FR+BC

4.0 FL+FR+FC+BC

quad

FL+FR+BL+BR

quad(side)

FL+FR+SL+SR

3.1 FL+FR+FC+LFE

5.0 FL+FR+FC+BL+BR

5.0(side)

FL+FR+FC+SL+SR

4.1 FL+FR+FC+LFE+BC

5.1 FL+FR+FC+LFE+BL+BR

5.1(side)

FL+FR+FC+LFE+SL+SR

6.0 FL+FR+FC+BC+SL+SR

6.0(front)

FL+FR+FLC+FRC+SL+SR

hexagonal

FL+FR+FC+BL+BR+BC

6.1 FL+FR+FC+LFE+BC+SL+SR

6.1 FL+FR+FC+LFE+BL+BR+BC

6.1(front)

FL+FR+LFE+FLC+FRC+SL+SR

7.0 FL+FR+FC+BL+BR+SL+SR

7.0(front)

FL+FR+FC+FLC+FRC+SL+SR

7.1 FL+FR+FC+LFE+BL+BR+SL+SR

7.1(wide)

FL+FR+FC+LFE+BL+BR+FLC+FRC

7.1(wide-side)

FL+FR+FC+LFE+FLC+FRC+SL+SR

7.1(top)

FL+FR+FC+LFE+BL+BR+TFL+TFR

octagonal

FL+FR+FC+BL+BR+BC+SL+SR

cube

FL+FR+BL+BR+TFL+TFR+TBL+TBR

hexadecagonal

FL+FR+FC+BL+BR+BC+SL+SR+WL+WR+TBL+TBR+TBC+TFC+TFL+TFR

downmix

DL+DR

22.2 FL+FR+FC+LFE+BL+BR+FLC+FRC+BC+SL+SR+TC+TFL+TFC+TFR+TBL+TBC+TBR+LFE2+TSL+TSR+

A custom channel layout can be specified as a sequence of terms, separated by '+'. Each term can be:

- ⊕ the name of a single channel (e.g. **FL**, **FR**, **FC**, **LFE**, etc.), each optionally containing a custom name after a '@', (e.g. **FL@Left**, **FR@Right**, **FC@Center**, **LFE@Low_Frequency**, etc.)

A standard channel layout can be specified by the following:

- ⊕ the name of a single channel (e.g. **FL**, **FR**, **FC**, **LFE**, etc.)
- ⊕ the name of a standard channel layout (e.g. **mono**, **stereo**, **4.0**, **quad**, **5.0**, etc.)
- ⊕ a number of channels, in decimal, followed by 'c', yielding the default channel layout for that number of channels (see the function "av_channel_layout_default"). Note that not all channel counts have a default layout.
- ⊕ a number of channels, in decimal, followed by 'C', yielding an unknown channel layout with the specified number of channels. Note that not all channel layout specification strings support

unknown channel layouts.

- ⊕ a channel layout mask, in hexadecimal starting with "0x" (see the "AV_CH_*" macros in *libavutil/channel_layout.h*).

Before libavutil version 53 the trailing character "c" to specify a number of channels was optional, but now it is required, while a channel layout mask can also be specified as a decimal number (if and only if not followed by "c" or "C").

See also the function "av_channel_layout_from_string" defined in *libavutil/channel_layout.h*.

EXPRESSION EVALUATION

When evaluating an arithmetic expression, FFmpeg uses an internal formula evaluator, implemented through the *libavutil/eval.h* interface.

An expression may contain unary, binary operators, constants, and functions.

Two expressions *expr1* and *expr2* can be combined to form another expression "*expr1;expr2*". *expr1* and *expr2* are evaluated in turn, and the new expression evaluates to the value of *expr2*.

The following binary operators are available: "+", "-", "*", "/", "^".

The following unary operators are available: "+", "-".

The following functions are available:

abs(x)

Compute absolute value of *x*.

acos(x)

Compute arccosine of *x*.

asin(x)

Compute arcsine of *x*.

atan(x)

Compute arctangent of *x*.

atan2(x, y)

Compute principal value of the arc tangent of *y/x*.

between(x, min, max)

Return 1 if x is greater than or equal to min and lesser than or equal to max , 0 otherwise.

bitand(x, y)**bitor(x, y)**

Compute bitwise and/or operation on x and y .

The results of the evaluation of x and y are converted to integers before executing the bitwise operation.

Note that both the conversion to integer and the conversion back to floating point can lose precision. Beware of unexpected results for large numbers (usually 2^{53} and larger).

ceil(expr)

Round the value of expression $expr$ upwards to the nearest integer. For example, "ceil(1.5)" is "2.0".

clip(x, min, max)

Return the value of x clipped between min and max .

cos(x)

Compute cosine of x .

cosh(x)

Compute hyperbolic cosine of x .

eq(x, y)

Return 1 if x and y are equivalent, 0 otherwise.

exp(x)

Compute exponential of x (with base "e", the Euler's number).

floor(expr)

Round the value of expression $expr$ downwards to the nearest integer. For example, "floor(-1.5)" is "-2.0".

gauss(x)

Compute Gauss function of x , corresponding to " $\exp(-x*x/2) / \sqrt{2*PI}$ ".

gcd(x, y)

Return the greatest common divisor of x and y . If both x and y are 0 or either or both are less than zero then behavior is undefined.

gt(x, y)

Return 1 if x is greater than y , 0 otherwise.

gte(x, y)

Return 1 if x is greater than or equal to y , 0 otherwise.

hypot(x, y)

This function is similar to the C function with the same name; it returns " $\sqrt{x*x + y*y}$ ", the length of the hypotenuse of a right triangle with sides of length x and y , or the distance of the point (x, y) from the origin.

if(x, y)

Evaluate x , and if the result is non-zero return the result of the evaluation of y , return 0 otherwise.

if(x, y, z)

Evaluate x , and if the result is non-zero return the evaluation result of y , otherwise the evaluation result of z .

ifnot(x, y)

Evaluate x , and if the result is zero return the result of the evaluation of y , return 0 otherwise.

ifnot(x, y, z)

Evaluate x , and if the result is zero return the evaluation result of y , otherwise the evaluation result of z .

isinf(x)

Return 1.0 if x is +/-INFINITY, 0.0 otherwise.

isnan(x)

Return 1.0 if x is NAN, 0.0 otherwise.

ld(var)

Load the value of the internal variable with number var , which was previously stored with $st(var, expr)$. The function returns the loaded value.

lerp(x, y, z)

Return linear interpolation between x and y by amount of z .

log(x)

Compute natural logarithm of x .

lt(x, y)

Return 1 if x is lesser than y , 0 otherwise.

lte(x, y)

Return 1 if x is lesser than or equal to y , 0 otherwise.

max(x, y)

Return the maximum between x and y .

min(x, y)

Return the minimum between x and y .

mod(x, y)

Compute the remainder of division of x by y .

not(expr)

Return 1.0 if $expr$ is zero, 0.0 otherwise.

pow(x, y)

Compute the power of x elevated y , it is equivalent to " $(x)^{(y)}$ ".

print(t)**print(t, l)**

Print the value of expression t with loglevel l . If l is not specified then a default log level is used.
Returns the value of the expression printed.

Prints t with loglevel l

random(x)

Return a pseudo random value between 0.0 and 1.0. x is the index of the internal variable which will be used to save the seed/state.

root(expr, max)

Find an input value for which the function represented by $expr$ with argument $ld(0)$ is 0 in the interval $0..max$.

The expression in $expr$ must denote a continuous function or the result is undefined.

ld(0) is used to represent the function input value, which means that the given expression will be evaluated multiple times with various input values that the expression can access through *ld(0)*. When the expression evaluates to 0 then the corresponding input value will be returned.

round(expr)

Round the value of expression *expr* to the nearest integer. For example, "round(1.5)" is "2.0".

sgn(x)

Compute sign of *x*.

sin(x)

Compute sine of *x*.

sinh(x)

Compute hyperbolic sine of *x*.

sqrt(expr)

Compute the square root of *expr*. This is equivalent to "*expr*^.5".

squish(x)

Compute expression "1/(1 + exp(4*x))".

st(var, expr)

Store the value of the expression *expr* in an internal variable. *var* specifies the number of the variable where to store the value, and it is a value ranging from 0 to 9. The function returns the value stored in the internal variable. Note, Variables are currently not shared between expressions.

tan(x)

Compute tangent of *x*.

tanh(x)

Compute hyperbolic tangent of *x*.

taylor(expr, x)**taylor(expr, x, id)**

Evaluate a Taylor series at *x*, given an expression representing the "*ld(id)*"-th derivative of a function at 0.

When the series does not converge the result is undefined.

ld(id) is used to represent the derivative order in *expr*, which means that the given expression will be evaluated multiple times with various input values that the expression can access through "ld(id)". If *id* is not specified then 0 is assumed.

Note, when you have the derivatives at *y* instead of 0, "taylor(*expr*, *x-y*)" can be used.

time(0)

Return the current (wallclock) time in seconds.

trunc(*expr*)

Round the value of expression *expr* towards zero to the nearest integer. For example, "trunc(-1.5)" is "-1.0".

while(*cond*, *expr*)

Evaluate expression *expr* while the expression *cond* is non-zero, and returns the value of the last *expr* evaluation, or NAN if *cond* was always false.

The following constants are available:

PI area of the unit disc, approximately 3.14

E **exp(1)** (Euler's number), approximately 2.718

PHI golden ratio $(1+\sqrt{5})/2$, approximately 1.618

Assuming that an expression is considered "true" if it has a non-zero value, note that:

"*" works like AND

"+" works like OR

For example the construct:

if (A AND B) then C

is equivalent to:

if(A*B, C)

In your C code, you can extend the list of unary and binary functions, and define recognized constants,

so that they are available for your expressions.

The evaluator also recognizes the International System unit prefixes. If 'i' is appended after the prefix, binary prefixes are used, which are based on powers of 1024 instead of powers of 1000. The 'B' postfix multiplies the value by 8, and can be appended after a unit prefix or used alone. This allows using for example 'KB', 'MiB', 'G' and 'B' as number postfix.

The list of available International System prefixes follows, with indication of the corresponding powers of 10 and of 2.

y $10^{-24} / 2^{-80}$

z $10^{-21} / 2^{-70}$

a $10^{-18} / 2^{-60}$

f $10^{-15} / 2^{-50}$

p $10^{-12} / 2^{-40}$

n $10^{-9} / 2^{-30}$

u $10^{-6} / 2^{-20}$

m $10^{-3} / 2^{-10}$

c 10^{-2}

d 10^{-1}

h 10^2

k $10^3 / 2^{10}$

K $10^3 / 2^{10}$

M $10^6 / 2^{20}$

G $10^9 / 2^{30}$

T $10^{12} / 2^{40}$

P $10^{15} / 2^{50}$

E $10^{18} / 2^{60}$

Z $10^{21} / 2^{70}$

Y $10^{24} / 2^{80}$

CODEC OPTIONS

libavcodec provides some generic global options, which can be set on all the encoders and decoders. In addition each codec may support so-called private options, which are specific for a given codec.

Sometimes, a global option may only affect a specific kind of codec, and may be nonsensical or ignored by another, so you need to be aware of the meaning of the specified options. Also some options are meant only for decoding or encoding.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the "AVCodecContext" options or using the *libavutil/opt.h* API for programmatic use.

The list of supported options follow:

b *integer (encoding, audio, video)*

Set bitrate in bits/s. Default value is 200K.

ab *integer (encoding, audio)*

Set audio bitrate (in bits/s). Default value is 128K.

bt *integer (encoding, video)*

Set video bitrate tolerance (in bits/s). In 1-pass mode, bitrate tolerance specifies how far ratecontrol is willing to deviate from the target average bitrate value. This is not related to min/max bitrate. Lowering tolerance too much has an adverse effect on quality.

flags *flags (decoding/encoding, audio, video, subtitles)*

Set generic flags.

Possible values:

mv4

Use four motion vector by macroblock (mpeg4).

qpel Use 1/4 pel motion compensation.

loop

Use loop filter.

qscale

Use fixed qscale.

pass1

Use internal 2pass ratecontrol in first pass mode.

pass2

Use internal 2pass ratecontrol in second pass mode.

gray

Only decode/encode grayscale.

psnr Set error[?] variables during encoding.

truncated

Input bitstream might be randomly truncated.

drop_changed

Don't output frames whose parameters differ from first decoded frame in stream. Error AVERROR_INPUT_CHANGED is returned when a frame is dropped.

ildct

Use interlaced DCT.

low_delay

Force low delay.

global_header

Place global headers in extradata instead of every keyframe.

bitexact

Only write platform-, build- and time-independent data. (except (I)DCT). This ensures that file and data checksums are reproducible and match between platforms. Its primary use is for

regression testing.

aic Apply H263 advanced intra coding / mpeg4 ac prediction.

ilme

Apply interlaced motion estimation.

cgop

Use closed gop.

output_corrupt

Output even potentially corrupted frames.

time_base *rational number*

Set codec time base.

It is the fundamental unit of time (in seconds) in terms of which frame timestamps are represented. For fixed-fps content, timebase should be "1 / frame_rate" and timestamp increments should be identically 1.

g *integer (encoding,video)*

Set the group of picture (GOP) size. Default value is 12.

ar *integer (decoding/encoding,audio)*

Set audio sampling rate (in Hz).

ac *integer (decoding/encoding,audio)*

Set number of audio channels.

cutoff *integer (encoding,audio)*

Set cutoff bandwidth. (Supported only by selected encoders, see their respective documentation sections.)

frame_size *integer (encoding,audio)*

Set audio frame size.

Each submitted frame except the last must contain exactly frame_size samples per channel. May be 0 when the codec has CODEC_CAP_VARIABLE_FRAME_SIZE set, in that case the frame size is not restricted. It is set by some decoders to indicate constant frame size.

frame_number *integer*

Set the frame number.

delay *integer***qcomp** *float (encoding,video)*

Set video quantizer scale compression (VBR). It is used as a constant in the ratecontrol equation. Recommended range for default rc_eq: 0.0-1.0.

qblur *float (encoding,video)*

Set video quantizer scale blur (VBR).

qmin *integer (encoding,video)*

Set min video quantizer scale (VBR). Must be included between -1 and 69, default value is 2.

qmax *integer (encoding,video)*

Set max video quantizer scale (VBR). Must be included between -1 and 1024, default value is 31.

qdiff *integer (encoding,video)*

Set max difference between the quantizer scale (VBR).

bf *integer (encoding,video)*

Set max number of B frames between non-B-frames.

Must be an integer between -1 and 16. 0 means that B-frames are disabled. If a value of -1 is used, it will choose an automatic value depending on the encoder.

Default value is 0.

b_qfactor *float (encoding,video)*

Set qp factor between P and B frames.

codec_tag *integer***bug_flags** *(decoding,video)*

Workaround not auto detected encoder bugs.

Possible values:

autodetect**xvid_ilace**

Xvid interlacing bug (autodetected if fourcc==XVIX)

ump4

(autodetected if fourcc==UMP4)

no_padding

padding bug (autodetected)

amv**qpel_chroma****std_qpel**

old standard qpel (autodetected per fourcc/version)

qpel_chroma2**direct_blocksize**

direct-qpel-blocksize bug (autodetected per fourcc/version)

edge

edge padding bug (autodetected per fourcc/version)

hpel_chroma**dc_clip**

ms Workaround various bugs in microsoft broken decoders.

trunc

truncated frames

strict *integer (decoding/encoding, audio, video)*

Specify how strictly to follow the standards.

Possible values:

very

strictly conform to an older more strict version of the spec or reference software

strict

strictly conform to all the things in the spec no matter what consequences

normal**unofficial**

allow unofficial extensions

experimental

allow non standardized experimental things, experimental (unfinished/work in progress/not well tested) decoders and encoders. Note: experimental decoders can pose a security risk, do not use this for decoding untrusted input.

b_qoffset *float (encoding,video)*

Set QP offset between P and B frames.

err_detect *flags (decoding,audio,video)*

Set error detection flags.

Possible values:

crccheck

verify embedded CRCs

bitstream

detect bitstream specification deviations

buffer

detect improper bitstream length

explode

abort decoding on minor error detection

ignore_err

ignore decoding errors, and continue decoding. This is useful if you want to analyze the content of a video and thus want everything to be decoded no matter what. This option will not result in a video that is pleasing to watch in case of errors.

careful

consider things that violate the spec and have not been seen in the wild as errors

compliant

consider all spec non compliances as errors

aggressive

consider things that a sane encoder should not do as an error

has_b_frames *integer*

block_align *integer*

rc_override_count *integer*

maxrate *integer (encoding, audio, video)*

Set max bitrate tolerance (in bits/s). Requires bufsize to be set.

minrate *integer (encoding, audio, video)*

Set min bitrate tolerance (in bits/s). Most useful in setting up a CBR encode. It is of little use otherwise.

bufsize *integer (encoding, audio, video)*

Set ratecontrol buffer size (in bits).

i_qfactor *float (encoding, video)*

Set QP factor between P and I frames.

i_qoffset *float (encoding, video)*

Set QP offset between P and I frames.

dct *integer (encoding, video)*

Set DCT algorithm.

Possible values:

auto autoselect a good one (default)

fastint

fast integer

int accurate integer

mmx

altivec

faan floating point AAN DCT

lumi_mask *float (encoding, video)*

Compress bright areas stronger than medium ones.

tcplx_mask *float (encoding, video)*

Set temporal complexity masking.

scplx_mask *float (encoding,video)*

Set spatial complexity masking.

p_mask *float (encoding,video)*

Set inter masking.

dark_mask *float (encoding,video)*

Compress dark areas stronger than medium ones.

idct *integer (decoding/encoding,video)*

Select IDCT implementation.

Possible values:

auto

int

simple

simplemmx

simpleauto

Automatically pick a IDCT compatible with the simple one

arm

altivec

sh4

simplearm

simplearmv5te

simplearmv6

simpleneon

xvid

faani

floating point AAN IDCT

slice_count *integer*

ec_flags *(decoding,video)*

Set error concealment strategy.

Possible values:

guess_mvs

iterative motion vector (MV) search (slow)

deblock

use strong deblock filter for damaged MBs

favor_inter

favor predicting from the previous frame instead of the current

bits_per_coded_sample *integer*

aspect *rational number (encoding,video)*

Set sample aspect ratio.

sar *rational number (encoding,video)*

Set sample aspect ratio. Alias to *aspect*.

debug *flags (decoding/encoding,audio,video,subtitles)*

Print specific debug info.

Possible values:

pict picture info

rc rate control

bitstream

mb_type

macroblock (MB) type

qp per-block quantization parameter (QP)

dct_coeff

green_metadata

display complexity metadata for the upcoming frame, GoP or for a given duration.

skip

startcode

er error recognition

mmco

memory management control operations (H.264)

bugs

buffers

picture buffer allocations

thread_ops

threading operations

nomc

skip motion compensation

cmp *integer (encoding, video)*

Set full pel me compare function.

Possible values:

sad sum of absolute differences, fast (default)

sse sum of squared errors

satd sum of absolute Hadamard transformed differences

dct sum of absolute DCT transformed differences

psnr sum of squared quantization errors (avoid, low quality)

bit number of bits needed for the block

rd rate distortion optimal, slow

zero 0

vsad

sum of absolute vertical differences

vsse sum of squared vertical differences

nsse noise preserving sum of squared differences

w53 5/3 wavelet, only used in snow

w97 9/7 wavelet, only used in snow

dctmax

chroma

subcmp *integer (encoding,video)*

Set sub pel me compare function.

Possible values:

sad sum of absolute differences, fast (default)

sse sum of squared errors

satd sum of absolute Hadamard transformed differences

dct sum of absolute DCT transformed differences

psnr sum of squared quantization errors (avoid, low quality)

bit number of bits needed for the block

rd rate distortion optimal, slow

zero 0

vsad

sum of absolute vertical differences

vsse sum of squared vertical differences

nsse noise preserving sum of squared differences

w53 5/3 wavelet, only used in snow

w97 9/7 wavelet, only used in snow

dctmax

chroma

mbcmp *integer (encoding,video)*

Set macroblock compare function.

Possible values:

sad sum of absolute differences, fast (default)

sse sum of squared errors

satd sum of absolute Hadamard transformed differences

dct sum of absolute DCT transformed differences

psnr sum of squared quantization errors (avoid, low quality)

bit number of bits needed for the block

rd rate distortion optimal, slow

zero 0

vsad

sum of absolute vertical differences

vsse sum of squared vertical differences

nsse noise preserving sum of squared differences

w53 5/3 wavelet, only used in snow

w97 9/7 wavelet, only used in snow

dctmax

chroma

ildctcmp *integer (encoding,video)*

Set interlaced dct compare function.

Possible values:

sad sum of absolute differences, fast (default)

sse sum of squared errors

satd sum of absolute Hadamard transformed differences

dct sum of absolute DCT transformed differences

psnr sum of squared quantization errors (avoid, low quality)

bit number of bits needed for the block

rd rate distortion optimal, slow

zero 0

vsad

sum of absolute vertical differences

vsse sum of squared vertical differences

nsse noise preserving sum of squared differences

w53 5/3 wavelet, only used in snow

w97 9/7 wavelet, only used in snow

dctmax

chroma

dia_size *integer (encoding, video)*

Set diamond type & size for motion estimation.

(1024, INT_MAX)

full motion estimation (slowest)

(768, 1024]

umh motion estimation

(512, 768]

hex motion estimation

(256, 512]

12s diamond motion estimation

[2, 256]

var diamond motion estimation

(-1, 2)

small diamond motion estimation

-1 funny diamond motion estimation

(INT_MIN, -1)

sab diamond motion estimation

last_pred *integer (encoding,video)*

Set amount of motion predictors from the previous frame.

precmp *integer (encoding,video)*

Set pre motion estimation compare function.

Possible values:

sad sum of absolute differences, fast (default)

sse sum of squared errors

satd sum of absolute Hadamard transformed differences

dct sum of absolute DCT transformed differences

psnr sum of squared quantization errors (avoid, low quality)

bit number of bits needed for the block

rd rate distortion optimal, slow

zero 0

vsad

sum of absolute vertical differences

vsse sum of squared vertical differences

nsse noise preserving sum of squared differences

w53 5/3 wavelet, only used in snow

w97 9/7 wavelet, only used in snow

dctmax

chroma

pre_dia_size *integer (encoding,video)*

Set diamond type & size for motion estimation pre-pass.

subq *integer (encoding,video)*

Set sub pel motion estimation quality.

me_range *integer (encoding,video)*

Set limit motion vectors range (1023 for DivX player).

global_quality *integer (encoding,audio,video)*

slice_flags *integer*

mbd *integer (encoding,video)*

Set macroblock decision algorithm (high quality mode).

Possible values:

simple

use mbcmp (default)

bits use fewest bits

rd use best rate distortion

rc_init_occupancy *integer (encoding,video)*

Set number of bits which should be loaded into the rc buffer before decoding starts.

flags2 *flags (decoding/encoding,audio,video,subtitles)*

Possible values:

fast Allow non spec compliant speedup tricks.

noout

Skip bitstream encoding.

ignorecrop

Ignore cropping information from sps.

local_header

Place global headers at every keyframe instead of in extradata.

chunks

Frame data might be split into multiple chunks.

showall

Show all frames before the first keyframe.

export_mvs

Export motion vectors into frame side-data (see "AV_FRAME_DATA_MOTION_VECTORS") for codecs that support it. See also *doc/examples/export_mvs.c*.

skip_manual

Do not skip samples and export skip information as frame side data.

ass_ro_flush_noop

Do not reset ASS ReadOrder field on flush.

icc_profiles

Generate/parse embedded ICC profiles from/to colorimetry tags.

export_side_data *flags (decoding/encoding, audio, video, subtitles)*

Possible values:

mvs Export motion vectors into frame side-data (see "AV_FRAME_DATA_MOTION_VECTORS") for codecs that support it. See also *doc/examples/export_mvs.c*.

prft Export encoder Producer Reference Time into packet side-data (see "AV_PKT_DATA_PRFT") for codecs that support it.

venc_params

Export video encoding parameters through frame side data (see "AV_FRAME_DATA_VIDEO_ENC_PARAMS") for codecs that support it. At present, those are H.264 and VP9.

film_grain

Export film grain parameters through frame side data (see

"AV_FRAME_DATA_FILM_GRAIN_PARAMS"). Supported at present by AV1 decoders.

threads *integer (decoding/encoding,video)*

Set the number of threads to be used, in case the selected codec implementation supports multi-threading.

Possible values:

auto, 0

automatically select the number of threads to set

Default value is **auto**.

dc *integer (encoding,video)*

Set `intra_dc_precision`.

nssew *integer (encoding,video)*

Set `nsse_weight`.

skip_top *integer (decoding,video)*

Set number of macroblock rows at the top which are skipped.

skip_bottom *integer (decoding,video)*

Set number of macroblock rows at the bottom which are skipped.

profile *integer (encoding,audio,video)*

Set encoder codec profile. Default value is **unknown**. Encoder specific profiles are documented in the relevant encoder documentation.

level *integer (encoding,audio,video)*

Possible values:

unknown

lowres *integer (decoding,audio,video)*

Decode at 1= 1/2, 2=1/4, 3=1/8 resolutions.

mblmin *integer (encoding,video)*

Set min macroblock lagrange factor (VBR).

mblmax *integer (encoding,video)*

Set max macroblock lagrange factor (VBR).

skip_loop_filter *integer (decoding,video)*

skip_idct *integer (decoding,video)*

skip_frame *integer (decoding,video)*

Make decoder discard processing depending on the frame type selected by the option value.

skip_loop_filter skips frame loop filtering, **skip_idct** skips frame IDCT/dequantization, **skip_frame** skips decoding.

Possible values:

none

Discard no frame.

default

Discard useless frames like 0-sized frames.

noref

Discard all non-reference frames.

bidir

Discard all bidirectional frames.

nokey

Discard all frames excepts keyframes.

nointra

Discard all frames except I frames.

all Discard all frames.

Default value is **default**.

bidir_refine *integer (encoding,video)*

Refine the two motion vectors used in bidirectional macroblocks.

keyint_min *integer (encoding,video)*

Set minimum interval between IDR-frames.

refs *integer (encoding,video)*

Set reference frames to consider for motion compensation.

trellis *integer (encoding,audio,video)*

Set rate-distortion optimal quantization.

mv0_threshold *integer (encoding,video)*

compression_level *integer (encoding,audio,video)*

bits_per_raw_sample *integer*

channel_layout *integer (decoding/encoding,audio)*

Possible values:

request_channel_layout *integer (decoding,audio)*

Possible values:

rc_max_vbv_use *float (encoding,video)*

rc_min_vbv_use *float (encoding,video)*

ticks_per_frame *integer (decoding/encoding,audio,video)*

color_primaries *integer (decoding/encoding,video)*

Possible values:

bt709

BT.709

bt470m

BT.470 M

bt470bg

BT.470 BG

smpte170m

SMPTE 170 M

smpte240m

SMPTE 240 M

film Film

bt2020

BT.2020

smpte428

smpte428_1

SMPTE ST 428-1

smpte431

SMPTE 431-2

smpte432

SMPTE 432-1

jedec-p22

JEDEC P22

color_trc *integer (decoding/encoding,video)*

Possible values:

bt709

BT.709

gamma22

BT.470 M

gamma28

BT.470 BG

smpte170m

SMPTE 170 M

smpte240m

SMPTE 240 M

linear

Linear

log

log100

Log

log_sqrt

log316

Log square root

iec61966_2_4

iec61966-2-4

IEC 61966-2-4

bt1361

bt1361e

BT.1361

iec61966_2_1

iec61966-2-1

IEC 61966-2-1

bt2020_10

bt2020_10bit

BT.2020 - 10 bit

bt2020_12

bt2020_12bit

BT.2020 - 12 bit

smpte2084

SMPTE ST 2084

smpte428

smpte428_1

SMPTE ST 428-1

arib-std-b67

ARIB STD-B67

colorspace *integer (decoding/encoding,video)*

Possible values:

rgb RGB

bt709

BT.709

fcc FCC

bt470bg

BT.470 BG

smpte170m

SMPTE 170 M

smpte240m

SMPTE 240 M

ycocg

YCOCG

bt2020nc

bt2020_ncl

BT.2020 NCL

bt2020c

bt2020_cl

BT.2020 CL

smpte2085

SMPTE 2085

chroma-derived-nc

Chroma-derived NCL

chroma-derived-c

Chroma-derived CL

ictcp

ICtCp

color_range *integer (decoding/encoding,video)*

If used as input parameter, it serves as a hint to the decoder, which color_range the input has.

Possible values:

tv

mpeg

MPEG (219*2⁽ⁿ⁻⁸⁾)

pc

jpeg JPEG (2ⁿ⁻¹)

chroma_sample_location *integer (decoding/encoding,video)*

Possible values:

left

center

topleft

top

bottomleft

bottom

log_level_offset *integer*

Set the log level offset.

slices *integer (encoding,video)*

Number of slices, used in parallelized encoding.

thread_type *flags (decoding/encoding,video)*

Select which multithreading methods to use.

Use of **frame** will increase decoding delay by one frame per thread, so clients which cannot provide future frames should not use it.

Possible values:

slice

Decode more than one part of a single frame at once.

Multithreading using slices works only when the video was encoded with slices.

frame

Decode more than one frame at once.

Default value is **slice+frame**.

audio_service_type *integer (encoding,audio)*

Set audio service type.

Possible values:

ma Main Audio Service

ef Effects

vi Visually Impaired

hi Hearing Impaired

di Dialogue

co Commentary

em Emergency

vo Voice Over

ka Karaoke

request_sample_fmt *sample_fmt (decoding, audio)*

Set sample format audio decoders should prefer. Default value is "none".

pkt_timebase *rational number*

sub_charenc *encoding (decoding, subtitles)*

Set the input subtitles character encoding.

field_order *field_order (video)*

Set/override the field order of the video. Possible values:

progressive

Progressive video

tt Interlaced video, top field coded and displayed first

bb Interlaced video, bottom field coded and displayed first

tb Interlaced video, top coded first, bottom displayed first

bt Interlaced video, bottom coded first, top displayed first

skip_alpha *bool (decoding,video)*

Set to 1 to disable processing alpha (transparency). This works like the **gray** flag in the **flags** option which skips chroma information instead of alpha. Default is 0.

codec_whitelist *list (input)*

", " separated list of allowed decoders. By default all are allowed.

dump_separator *string (input)*

Separator used to separate the fields printed on the command line about the Stream parameters. For example, to separate the fields with newlines and indentation:

```
ffprobe -dump_separator "
    " -i ~/videos/matrixbench_mpeg2.mpg
```

max_pixels *integer (decoding/encoding,video)*

Maximum number of pixels per image. This value can be used to avoid out of memory failures due to large images.

apply_cropping *bool (decoding,video)*

Enable cropping if cropping parameters are multiples of the required alignment for the left and top parameters. If the alignment is not met the cropping will be partially applied to maintain alignment. Default is 1 (enabled). Note: The required alignment depends on if "AV_CODEC_FLAG_UNALIGNED" is set and the CPU. "AV_CODEC_FLAG_UNALIGNED" cannot be changed from the command line. Also hardware decoders will not apply left/top Cropping.

DECODERS

Decoders are configured elements in FFmpeg which allow the decoding of multimedia streams.

When you configure your FFmpeg build, all the supported native decoders are enabled by default. Decoders requiring an external library must be enabled manually via the corresponding "--enable-lib" option. You can list all available decoders using the configure option "--list-decoders".

You can disable all the decoders with the configure option "--disable-decoders" and selectively enable / disable single decoders with the options "--enable-decoder=*DECODER*" / "--disable-decoder=*DECODER*".

The option "-decoders" of the ff* tools will display the list of enabled decoders.

VIDEO DECODERS

A description of some of the currently available video decoders follows.

av1

AOMedia Video 1 (AV1) decoder.

*Options***operating_point**

Select an operating point of a scalable AV1 bitstream (0 - 31). Default is 0.

rawvideo

Raw video decoder.

This decoder decodes rawvideo streams.

*Options***top** *top_field_first*

Specify the assumed field type of the input video.

-1 the video is assumed to be progressive (default)

0 bottom-field-first is assumed

1 top-field-first is assumed

libdav1d

dav1d AV1 decoder.

libdav1d allows libavcodec to decode the AOMedia Video 1 (AV1) codec. Requires the presence of the libdav1d headers and library during configuration. You need to explicitly configure the build with "--enable-libdav1d".

Options

The following options are supported by the libdav1d wrapper.

framethreads

Set amount of frame threads to use during decoding. The default value is 0 (autodetect). This option is deprecated for libdav1d >= 1.0 and will be removed in the future. Use the option

"max_frame_delay" and the global option "threads" instead.

tilethreads

Set amount of tile threads to use during decoding. The default value is 0 (autodetect). This option is deprecated for libdav1d >= 1.0 and will be removed in the future. Use the global option "threads" instead.

max_frame_delay

Set max amount of frames the decoder may buffer internally. The default value is 0 (autodetect).

filmgrain

Apply film grain to the decoded video if present in the bitstream. Defaults to the internal default of the library. This option is deprecated and will be removed in the future. See the global option "export_side_data" to export Film Grain parameters instead of applying it.

oppoint

Select an operating point of a scalable AV1 bitstream (0 - 31). Defaults to the internal default of the library.

alllayers

Output all spatial layers of a scalable AV1 bitstream. The default value is false.

libdavs2

AVS2-P2/IEEE1857.4 video decoder wrapper.

This decoder allows libavcodec to decode AVS2 streams with dav2 library.

libuavs3d

AVS3-P2/IEEE1857.10 video decoder.

libuavs3d allows libavcodec to decode AVS3 streams. Requires the presence of the libuavs3d headers and library during configuration. You need to explicitly configure the build with "--enable-libuavs3d".

Options

The following option is supported by the libuavs3d wrapper.

frame_threads

Set amount of frame threads to use during decoding. The default value is 0 (autodetect).

QSV Decoders

The family of Intel QuickSync Video decoders (VC1, MPEG-2, H.264, HEVC, JPEG/MJPEG, VP8, VP9, AV1).

Common Options

The following options are supported by all qsv decoders.

async_depth

Internal parallelization depth, the higher the value the higher the latency.

gpu_copy

A GPU-accelerated copy between video and system memory

default

on

off

HEVC Options

Extra options for hevc_qsv.

load_plugin

A user plugin to load in an internal session

none

hevc_sw

hevc_hw

load_plugins

A :-separate list of hexadecimal plugin UIDs to load in an internal session

v210

Uncompressed 4:2:2 10-bit decoder.

Options

custom_stride

Set the line size of the v210 data in bytes. The default value is 0 (autodetect). You can use the special -1 value for a strideless v210 as seen in BOXX files.

AUDIO DECODERS

A description of some of the currently available audio decoders follows.

ac3

AC-3 audio decoder.

This decoder implements part of ATSC A/52:2010 and ETSI TS 102 366, as well as the undocumented RealAudio 3 (a.k.a. dnet).

AC-3 Decoder Options

-drc_scale value

Dynamic Range Scale Factor. The factor to apply to dynamic range values from the AC-3 stream. This factor is applied exponentially. The default value is 1. There are 3 notable scale factor ranges:

drc_scale == 0

DRC disabled. Produces full range audio.

0 < drc_scale <= 1

DRC enabled. Applies a fraction of the stream DRC value. Audio reproduction is between full range and full compression.

drc_scale > 1

DRC enabled. Applies drc_scale asymmetrically. Loud sounds are fully compressed. Soft sounds are enhanced.

flac

FLAC audio decoder.

This decoder aims to implement the complete FLAC specification from Xiph.

FLAC Decoder options

-use_buggy_lpc

The lavc FLAC encoder used to produce buggy streams with high lpc values (like the default value). This option makes it possible to decode such streams correctly by using lavc's old buggy lpc logic for decoding.

ffwavesynth

Internal wave synthesizer.

This decoder generates wave patterns according to predefined sequences. Its use is purely internal and the format of the data it accepts is not publicly documented.

libcelt

libcelt decoder wrapper.

libcelt allows libavcodec to decode the Xiph CELT ultra-low delay audio codec. Requires the presence of the libcelt headers and library during configuration. You need to explicitly configure the build with "--enable-libcelt".

libgsm

libgsm decoder wrapper.

libgsm allows libavcodec to decode the GSM full rate audio codec. Requires the presence of the libgsm headers and library during configuration. You need to explicitly configure the build with "--enable-libgsm".

This decoder supports both the ordinary GSM and the Microsoft variant.

libilbc

libilbc decoder wrapper.

libilbc allows libavcodec to decode the Internet Low Bitrate Codec (iLBC) audio codec. Requires the presence of the libilbc headers and library during configuration. You need to explicitly configure the build with "--enable-libilbc".

Options

The following option is supported by the libilbc wrapper.

enhance

Enable the enhancement of the decoded audio when set to 1. The default value is 0 (disabled).

libopencore-amrnb

libopencore-amrnb decoder wrapper.

libopencore-amrnb allows libavcodec to decode the Adaptive Multi-Rate Narrowband audio codec. Using it requires the presence of the libopencore-amrnb headers and library during configuration. You

need to explicitly configure the build with "--enable-libopencore-amrnb".

An FFmpeg native decoder for AMR-NB exists, so users can decode AMR-NB without this library.

libopencore-amrwb

libopencore-amrwb decoder wrapper.

libopencore-amrwb allows libavcodec to decode the Adaptive Multi-Rate Wideband audio codec. Using it requires the presence of the libopencore-amrwb headers and library during configuration. You need to explicitly configure the build with "--enable-libopencore-amrwb".

An FFmpeg native decoder for AMR-WB exists, so users can decode AMR-WB without this library.

libopus

libopus decoder wrapper.

libopus allows libavcodec to decode the Opus Interactive Audio Codec. Requires the presence of the libopus headers and library during configuration. You need to explicitly configure the build with "--enable-libopus".

An FFmpeg native decoder for Opus exists, so users can decode Opus without this library.

SUBTITLES DECODERS

libaribb24

ARIB STD-B24 caption decoder.

Implements profiles A and C of the ARIB STD-B24 standard.

libaribb24 Decoder Options

-aribb24-base-path *path*

Sets the base path for the libaribb24 library. This is utilized for reading of configuration files (for custom unicode conversions), and for dumping of non-text symbols as images under that location.

Unset by default.

-aribb24-skip-ruby-text *boolean*

Tells the decoder wrapper to skip text blocks that contain half-height ruby text.

Enabled by default.

dvbsub*Options***compute_clut**

- 2 Compute clut once if no matching CLUT is in the stream.
- 1 Compute clut if no matching CLUT is in the stream.
- 0 Never compute CLUT
- 1 Always compute CLUT and override the one provided in the stream.

dvb_substream

Selects the dvb substream, or all substreams if -1 which is default.

dvdsb

This codec decodes the bitmap subtitles used in DVDs; the same subtitles can also be found in VobSub file pairs and in some Matroska files.

*Options***palette**

Specify the global palette used by the bitmaps. When stored in VobSub, the palette is normally specified in the index file; in Matroska, the palette is stored in the codec extra-data in the same format as in VobSub. In DVDs, the palette is stored in the IFO file, and therefore not available when reading from dumped VOB files.

The format for this option is a string containing 16 24-bits hexadecimal numbers (without 0x prefix) separated by commas, for example "0d00ee, ee450d, 101010, eaeaea, 0ce60b, ec14ed, ebff0b, 0d617a, 7b7b7b, d1d1d1, 7b2a0e, 0d950c, 0f007b, cf0dec, cfa80c, 7c127b".

ifo_palette

Specify the IFO file from which the global palette is obtained. (experimental)

forced_subs_only

Only decode subtitle entries marked as forced. Some titles have forced and non-forced subtitles in the same track. Setting this flag to 1 will only keep the forced subtitles. Default value is 0.

libzvbi-teletext

Libzvbi allows libavcodec to decode DVB teletext pages and DVB teletext subtitles. Requires the

presence of the libzvbi headers and library during configuration. You need to explicitly configure the build with "--enable-libzvbi".

Options

txt_page

List of teletext page numbers to decode. Pages that do not match the specified list are dropped. You may use the special "*" string to match all pages, or "subtitle" to match all subtitle pages. Default value is *.

txt_default_region

Set default character set used for decoding, a value between 0 and 87 (see ETS 300 706, Section 15, Table 32). Default value is -1, which does not override the libzvbi default. This option is needed for some legacy level 1.0 transmissions which cannot signal the proper charset.

txt_chop_top

Discards the top teletext line. Default value is 1.

txt_format

Specifies the format of the decoded subtitles.

bitmap

The default format, you should use this for teletext pages, because certain graphics and colors cannot be expressed in simple text or even ASS.

text Simple text based output without formatting.

ass Formatted ASS output, subtitle pages and teletext pages are returned in different styles, subtitle pages are stripped down to text, but an effort is made to keep the text alignment and the formatting.

txt_left

X offset of generated bitmaps, default is 0.

txt_top

Y offset of generated bitmaps, default is 0.

txt_chop_spaces

Chops leading and trailing spaces and removes empty lines from the generated text. This option is useful for teletext based subtitles where empty spaces may be present at the start or at the end of

the lines or empty lines may be present between the subtitle lines because of double-sized teletext characters. Default value is 1.

txt_duration

Sets the display duration of the decoded teletext pages or subtitles in milliseconds. Default value is -1 which means infinity or until the next subtitle event comes.

txt_transparent

Force transparent background of the generated teletext bitmaps. Default value is 0 which means an opaque background.

txt_opacity

Sets the opacity (0-255) of the teletext background. If **txt_transparent** is not set, it only affects characters between a start box and an end box, typically subtitles. Default value is 0 if **txt_transparent** is set, 255 otherwise.

ENCODERS

Encoders are configured elements in FFmpeg which allow the encoding of multimedia streams.

When you configure your FFmpeg build, all the supported native encoders are enabled by default. Encoders requiring an external library must be enabled manually via the corresponding "--enable-lib" option. You can list all available encoders using the configure option "--list-encoders".

You can disable all the encoders with the configure option "--disable-encoders" and selectively enable / disable single encoders with the options "--enable-encoder=*ENCODER*" / "--disable-encoder=*ENCODER*".

The option "-encoders" of the ff* tools will display the list of enabled encoders.

AUDIO ENCODERS

A description of some of the currently available audio encoders follows.

aac

Advanced Audio Coding (AAC) encoder.

This encoder is the default AAC encoder, natively implemented into FFmpeg.

Options

b Set bit rate in bits/s. Setting this automatically activates constant bit rate (CBR) mode. If this

option is unspecified it is set to 128kbps.

- q** Set quality for variable bit rate (VBR) mode. This option is valid only using the **ffmpeg** command-line tool. For library interface users, use **global_quality**.

cutoff

Set cutoff frequency. If unspecified will allow the encoder to dynamically adjust the cutoff to improve clarity on low bitrates.

aac_coder

Set AAC encoder coding method. Possible values:

twoloop

Two loop searching (TLS) method. This is the default method.

This method first sets quantizers depending on band thresholds and then tries to find an optimal combination by adding or subtracting a specific value from all quantizers and adjusting some individual quantizer a little. Will tune itself based on whether **aac_is**, **aac_ms** and **aac_pns** are enabled.

anmr

Average noise to mask ratio (ANMR) trellis-based solution.

This is an experimental coder which currently produces a lower quality, is more unstable and is slower than the default twoloop coder but has potential. Currently has no support for the **aac_is** or **aac_pns** options. Not currently recommended.

fast Constant quantizer method.

Uses a cheaper version of twoloop algorithm that doesn't try to do as many clever adjustments. Worse with low bitrates (less than 64kbps), but is better and much faster at higher bitrates.

aac_ms

Sets mid/side coding mode. The default value of "auto" will automatically use M/S with bands which will benefit from such coding. Can be forced for all bands using the value "enable", which is mainly useful for debugging or disabled using "disable".

aac_is

Sets intensity stereo coding tool usage. By default, it's enabled and will automatically toggle IS

for similar pairs of stereo bands if it's beneficial. Can be disabled for debugging by setting the value to "disable".

aac_pns

Uses perceptual noise substitution to replace low entropy high frequency bands with imperceptible white noise during the decoding process. By default, it's enabled, but can be disabled for debugging purposes by using "disable".

aac_tns

Enables the use of a multitap FIR filter which spans through the high frequency bands to hide quantization noise during the encoding process and is reverted by the decoder. As well as decreasing unpleasant artifacts in the high range this also reduces the entropy in the high bands and allows for more bits to be used by the mid-low bands. By default it's enabled but can be disabled for debugging by setting the option to "disable".

aac_ltp

Enables the use of the long term prediction extension which increases coding efficiency in very low bandwidth situations such as encoding of voice or solo piano music by extending constant harmonic peaks in bands throughout frames. This option is implied by profile:aac_low and is incompatible with aac_pred. Use in conjunction with **-ar** to decrease the samplerate.

aac_pred

Enables the use of a more traditional style of prediction where the spectral coefficients transmitted are replaced by the difference of the current coefficients minus the previous "predicted" coefficients. In theory and sometimes in practice this can improve quality for low to mid bitrate audio. This option implies the aac_main profile and is incompatible with aac_ltp.

profile

Sets the encoding profile, possible values:

aac_low

The default, AAC "Low-complexity" profile. Is the most compatible and produces decent quality.

mpeg2_aac_low

Equivalent to "-profile:a aac_low -aac_pns 0". PNS was introduced with the MPEG4 specifications.

aac_ltp

Long term prediction profile, is enabled by and will enable the **aac_ltp** option. Introduced in

MPEG4.

aac_main

Main-type prediction profile, is enabled by and will enable the **aac_pred** option. Introduced in MPEG2.

If this option is unspecified it is set to **aac_low**.

ac3 and ac3_fixed

AC-3 audio encoders.

These encoders implement part of ATSC A/52:2010 and ETSI TS 102 366, as well as the undocumented RealAudio 3 (a.k.a. dnet).

The *ac3* encoder uses floating-point math, while the *ac3_fixed* encoder only uses fixed-point integer math. This does not mean that one is always faster, just that one or the other may be better suited to a particular system. The *ac3_fixed* encoder is not the default codec for any of the output formats, so it must be specified explicitly using the option "-acodec ac3_fixed" in order to use it.

AC-3 Metadata

The AC-3 metadata options are used to set parameters that describe the audio, but in most cases do not affect the audio encoding itself. Some of the options do directly affect or influence the decoding and playback of the resulting bitstream, while others are just for informational purposes. A few of the options will add bits to the output stream that could otherwise be used for audio data, and will thus affect the quality of the output. Those will be indicated accordingly with a note in the option list below.

These parameters are described in detail in several publicly-available documents.

*<<http://www.atsc.org/cms/standards/a_52-2010.pdf>>

*<<http://www.atsc.org/cms/standards/a_54a_with_corr_1.pdf>>

*<<http://www.dolby.com/uploadedFiles/zz-_Shared_Assets/English_PDFs/Professional/18_Metadata.Guide.pdf>>

*<<http://www.dolby.com/uploadedFiles/zz-_Shared_Assets/English_PDFs/Professional/46_DDEncodingGuidelines.pdf>>

Metadata Control Options

-per_frame_metadata *boolean*

Allow Per-Frame Metadata. Specifies if the encoder should check for changing metadata for each frame.

- 0** The metadata values set at initialization will be used for every frame in the stream. (default)
- 1** Metadata values can be changed before encoding each frame.

Downmix Levels

-center_mixlev *level*

Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo. This field will only be written to the bitstream if a center channel is present. The value is specified as a scale factor. There are 3 valid values:

0.707

Apply -3dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6dB gain

-surround_mixlev *level*

Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo. This field will only be written to the bitstream if one or more surround channels are present. The value is specified as a scale factor. There are 3 valid values:

0.707

Apply -3dB gain

0.500

Apply -6dB gain (default)

0.000

Silence Surround Channel(s)

Audio Production Information

Audio Production Information is optional information describing the mixing environment. Either none or both of the fields are written to the bitstream.

-mixing_level *number*

Mixing Level. Specifies peak sound pressure level (SPL) in the production environment when the mix was mastered. Valid values are 80 to 111, or -1 for unknown or not indicated. The default value is -1, but that value cannot be used if the Audio Production Information is written to the bitstream. Therefore, if the "room_type" option is not the default value, the "mixing_level" option must not be -1.

-room_type *type*

Room Type. Describes the equalization used during the final mixing session at the studio or on the dubbing stage. A large room is a dubbing stage with the industry standard X-curve equalization; a small room has flat equalization. This field will not be written to the bitstream if both the "mixing_level" option and the "room_type" option have the default values.

0

notindicated

Not Indicated (default)

1

large

Large Room

2

small

Small Room

Other Metadata Options

-copyright *boolean*

Copyright Indicator. Specifies whether a copyright exists for this audio.

0

off No Copyright Exists (default)

1

on Copyright Exists

-dialnorm *value*

Dialogue Normalization. Indicates how far the average dialogue level of the program is below digital 100% full scale (0 dBFS). This parameter determines a level shift during audio reproduction that sets the average volume of the dialogue to a preset level. The goal is to match volume level between program sources. A value of -31dB will result in no volume level change,

relative to the source volume, during audio reproduction. Valid values are whole numbers in the range -31 to -1, with -31 being the default.

-dsur_mode *mode*

Dolby Surround Mode. Specifies whether the stereo signal uses Dolby Surround (Pro Logic). This field will only be written to the bitstream if the audio stream is stereo. Using this option does **NOT** mean the encoder will actually apply Dolby Surround processing.

0

notindicated

Not Indicated (default)

1

off Not Dolby Surround Encoded

2

on Dolby Surround Encoded

-original *boolean*

Original Bit Stream Indicator. Specifies whether this audio is from the original source and not a copy.

0

off Not Original Source

1

on Original Source (default)

Extended Bitstream Information

The extended bitstream options are part of the Alternate Bit Stream Syntax as specified in Annex D of the A/52:2010 standard. It is grouped into 2 parts. If any one parameter in a group is specified, all values in that group will be written to the bitstream. Default values are used for those that are written but have not been specified. If the mixing levels are written, the decoder will use these values instead of the ones specified in the "center_mixlev" and "surround_mixlev" options if it supports the Alternate Bit Stream Syntax.

Extended Bitstream Information - Part 1

-dmix_mode *mode*

Preferred Stereo Downmix Mode. Allows the user to select either Lt/Rt (Dolby Surround) or Lo/Ro (normal stereo) as the preferred stereo downmix mode.

0

notindicated

Not Indicated (default)

1

lrrt Lt/Rt Downmix Preferred

2

lrrr Lo/Ro Downmix Preferred

-lrrr_cmixlev *level*

Lt/Rt Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo in Lt/Rt mode.

1.414

Apply +3dB gain

1.189

Apply +1.5dB gain

1.000

Apply 0dB gain

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6.0dB gain

0.000

Silence Center Channel

-ltrt_surmixlev *level*

Lt/Rt Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo in Lt/Rt mode.

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain

0.500

Apply -6.0dB gain (default)

0.000

Silence Surround Channel(s)

-loro_cmixlev *level*

Lo/Ro Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo in Lo/Ro mode.

1.414

Apply +3dB gain

1.189

Apply +1.5dB gain

1.000

Apply 0dB gain

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6.0dB gain

0.000

Silence Center Channel

-loro_surmixlev *level*

Lo/Ro Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo in Lo/Ro mode.

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain

0.500

Apply -6.0dB gain (default)

0.000

Silence Surround Channel(s)

Extended Bitstream Information - Part 2

-dsurex_mode *mode*

Dolby Surround EX Mode. Indicates whether the stream uses Dolby Surround EX (7.1 matrixed to 5.1). Using this option does **NOT** mean the encoder will actually apply Dolby Surround EX processing.

0**notindicated**

Not Indicated (default)

1

on Dolby Surround EX Off

2

off Dolby Surround EX On

-dheadphone_mode *mode*

Dolby Headphone Mode. Indicates whether the stream uses Dolby Headphone encoding (multi-channel matrixed to 2.0 for use with headphones). Using this option does **NOT** mean the encoder will actually apply Dolby Headphone processing.

0

notindicated

Not Indicated (default)

1

on Dolby Headphone Off

2

off Dolby Headphone On

-ad_conv_type *type*

A/D Converter Type. Indicates whether the audio has passed through HDCD A/D conversion.

0

standard

Standard A/D Converter (default)

1

hdc

HDCD A/D Converter

Other AC-3 Encoding Options

-stereo_rematrixing *boolean*

Stereo Rematrixing. Enables/Disables use of rematrixing for stereo input. This is an optional AC-3 feature that increases quality by selectively encoding the left/right channels as mid/side. This option is enabled by default, and it is highly recommended that it be left as enabled except for testing purposes.

cutoff *frequency*

Set lowpass cutoff frequency. If unspecified, the encoder selects a default determined by various other encoding parameters.

Floating-Point-Only AC-3 Encoding Options

These options are only valid for the floating-point encoder and do not exist for the fixed-point encoder due to the corresponding features not being implemented in fixed-point.

-channel_coupling *boolean*

Enables/Disables use of channel coupling, which is an optional AC-3 feature that increases quality by combining high frequency information from multiple channels into a single channel. The per-channel high frequency information is sent with less accuracy in both the frequency and time domains. This allows more bits to be used for lower frequencies while preserving enough information to reconstruct the high frequencies. This option is enabled by default for the floating-point encoder and should generally be left as enabled except for testing purposes or to increase encoding speed.

-1**auto** Selected by Encoder (default)**0****off** Disable Channel Coupling**1****on** Enable Channel Coupling**-cpl_start_band** *number*

Coupling Start Band. Sets the channel coupling start band, from 1 to 15. If a value higher than the bandwidth is used, it will be reduced to 1 less than the coupling end band. If *auto* is used, the start band will be determined by the encoder based on the bit rate, sample rate, and channel layout. This option has no effect if channel coupling is disabled.

-1**auto** Selected by Encoder (default)**flac**

FLAC (Free Lossless Audio Codec) Encoder

Options

The following options are supported by FFmpeg's flac encoder.

compression_level

Sets the compression level, which chooses defaults for many other options if they are not set explicitly. Valid values are from 0 to 12, 5 is the default.

frame_size

Sets the size of the frames in samples per channel.

lpc_coeff_precision

Sets the LPC coefficient precision, valid values are from 1 to 15, 15 is the default.

lpc_type

Sets the first stage LPC algorithm

none

LPC is not used

fixed

fixed LPC coefficients

levinson**cholesky****lpc_passes**

Number of passes to use for Cholesky factorization during LPC analysis

min_partition_order

The minimum partition order

max_partition_order

The maximum partition order

prediction_order_method**estimation****2level****4level****8level****search**

Bruteforce search

log**ch_mode**

Channel mode

auto The mode is chosen automatically for each frame

indep

Channels are independently coded

left_side

right_side

mid_side

exact_rice_parameters

Chooses if rice parameters are calculated exactly or approximately. if set to 1 then they are chosen exactly, which slows the code down slightly and improves compression slightly.

multi_dim_quant

Multi Dimensional Quantization. If set to 1 then a 2nd stage LPC algorithm is applied after the first stage to finetune the coefficients. This is quite slow and slightly improves compression.

opus

Opus encoder.

This is a native FFmpeg encoder for the Opus format. Currently its in development and only implements the CELT part of the codec. Its quality is usually worse and at best is equal to the libopus encoder.

Options

b Set bit rate in bits/s. If unspecified it uses the number of channels and the layout to make a good guess.

opus_delay

Sets the maximum delay in milliseconds. Lower delays than 20ms will very quickly decrease quality.

libfdk_aac

libfdk-aac AAC (Advanced Audio Coding) encoder wrapper.

The libfdk-aac library is based on the Fraunhofer FDK AAC code from the Android project.

Requires the presence of the libfdk-aac headers and library during configuration. You need to explicitly configure the build with "--enable-libfdk-aac". The library is also incompatible with GPL, so if you allow the use of GPL, you should configure with "--enable-gpl --enable-nonfree --enable-libfdk-aac".

This encoder has support for the AAC-HE profiles.

VBR encoding, enabled through the **vbr** or **flags +qscale** options, is experimental and only works with some combinations of parameters.

Support for encoding 7.1 audio is only available with libfdk-aac 0.1.3 or higher.

For more information see the fdk-aac project at <<http://sourceforge.net/p/opencore-amr/fdk-aac/>>.

Options

The following options are mapped on the shared FFmpeg codec options.

b Set bit rate in bits/s. If the bitrate is not explicitly specified, it is automatically set to a suitable value depending on the selected profile.

In case VBR mode is enabled the option is ignored.

ar Set audio sampling rate (in Hz).

channels

Set the number of audio channels.

flags +qscale

Enable fixed quality, VBR (Variable Bit Rate) mode. Note that VBR is implicitly enabled when the **vbr** value is positive.

cutoff

Set cutoff frequency. If not specified (or explicitly set to 0) it will use a value automatically computed by the library. Default value is 0.

profile

Set audio profile.

The following profiles are recognized:

aac_low

Low Complexity AAC (LC)

aac_he

High Efficiency AAC (HE-AAC)

aac_he_v2

High Efficiency AAC version 2 (HE-AACv2)

aac_ld

Low Delay AAC (LD)

aac_eld

Enhanced Low Delay AAC (ELD)

If not specified it is set to **aac_low**.

The following are private options of the libfdk_aac encoder.

afterburner

Enable afterburner feature if set to 1, disabled if set to 0. This improves the quality but also the required processing power.

Default value is 1.

eld_sbr

Enable SBR (Spectral Band Replication) for ELD if set to 1, disabled if set to 0.

Default value is 0.

eld_v2

Enable ELDv2 (LD-MPS extension for ELD stereo signals) for ELDv2 if set to 1, disabled if set to 0.

Note that option is available when fdk-aac version
(AACENCODER_LIB_VL0.AACENCODER_LIB_VL1.AACENCODER_LIB_VL2) > (4.0.0).

Default value is 0.

signaling

Set SBR/PS signaling style.

It can assume one of the following values:

default

choose signaling implicitly (explicit hierarchical by default, implicit if global header is disabled)

implicit

implicit backwards compatible signaling

explicit_sbr

explicit SBR, implicit PS signaling

explicit_hierarchical

explicit hierarchical signaling

Default value is **default**.

latm

Output LATM/LOAS encapsulated data if set to 1, disabled if set to 0.

Default value is 0.

header_period

Set StreamMuxConfig and PCE repetition period (in frames) for sending in-band configuration buffers within LATM/LOAS transport layer.

Must be a 16-bits non-negative integer.

Default value is 0.

vbr Set VBR mode, from 1 to 5. 1 is lowest quality (though still pretty good) and 5 is highest quality. A value of 0 will disable VBR, and CBR (Constant Bit Rate) is enabled.

Currently only the **aac_low** profile supports VBR encoding.

VBR modes 1-5 correspond to roughly the following average bit rates:

- 1** 32 kbps/channel
- 2** 40 kbps/channel
- 3** 48-56 kbps/channel

4 64 kbps/channel

5 about 80-96 kbps/channel

Default value is 0.

Examples

- ⊕ Use **ffmpeg** to convert an audio file to VBR AAC in an M4A (MP4) container:

```
ffmpeg -i input.wav -codec:a libfdk_aac -vbr 3 output.m4a
```

- ⊕ Use **ffmpeg** to convert an audio file to CBR 64k kbps AAC, using the High-Efficiency AAC profile:

```
ffmpeg -i input.wav -c:a libfdk_aac -profile:a aac_he -b:a 64k output.m4a
```

libmp3lame

LAME (Lame Ain't an MP3 Encoder) MP3 encoder wrapper.

Requires the presence of the libmp3lame headers and library during configuration. You need to explicitly configure the build with "--enable-libmp3lame".

See **libshine** for a fixed-point MP3 encoder, although with a lower quality.

Options

The following options are supported by the libmp3lame wrapper. The **lame**-equivalent of the options are listed in parentheses.

b (-b)

Set bitrate expressed in bits/s for CBR or ABR. LAME "bitrate" is expressed in kilobits/s.

q (-V)

Set constant quality setting for VBR. This option is valid only using the **ffmpeg** command-line tool. For library interface users, use **global_quality**.

compression_level (-q)

Set algorithm quality. Valid arguments are integers in the 0-9 range, with 0 meaning highest quality but slowest, and 9 meaning fastest while producing the worst quality.

cutoff (*--lowpass*)

Set lowpass cutoff frequency. If unspecified, the encoder dynamically adjusts the cutoff.

reservoir

Enable use of bit reservoir when set to 1. Default value is 1. LAME has this enabled by default, but can be overridden by use **--nores** option.

joint_stereo (*-m j*)

Enable the encoder to use (on a frame by frame basis) either L/R stereo or mid/side stereo. Default value is 1.

abr (*--abr*)

Enable the encoder to use ABR when set to 1. The **lame --abr** sets the target bitrate, while this options only tells FFmpeg to use ABR still relies on **b** to set bitrate.

libopencore-amrnb

OpenCORE Adaptive Multi-Rate Narrowband encoder.

Requires the presence of the libopencore-amrnb headers and library during configuration. You need to explicitly configure the build with "**--enable-libopencore-amrnb --enable-version3**".

This is a mono-only encoder. Officially it only supports 8000Hz sample rate, but you can override it by setting **strict** to **unofficial** or lower.

Options

b Set bitrate in bits per second. Only the following bitrates are supported, otherwise libavcodec will round to the nearest valid bitrate.

4750

5150

5900

6700

7400

7950

10200

12200

dtx Allow discontinuous transmission (generate comfort noise) when set to 1. The default value is 0 (disabled).

libopus

libopus Opus Interactive Audio Codec encoder wrapper.

Requires the presence of the libopus headers and library during configuration. You need to explicitly configure the build with "--enable-libopus".

Option Mapping

Most libopus options are modelled after the **opusenc** utility from opus-tools. The following is an option mapping chart describing options supported by the libopus wrapper, and their **opusenc**-equivalent in parentheses.

b (*bitrate*)

Set the bit rate in bits/s. FFmpeg's **b** option is expressed in bits/s, while **opusenc**'s **bitrate** in kilobits/s.

vbr (*vbr, hard-cbr, and cvbr*)

Set VBR mode. The FFmpeg **vbr** option has the following valid arguments, with the **opusenc** equivalent options in parentheses:

off (*hard-cbr*)

Use constant bit rate encoding.

on (*vbr*)

Use variable bit rate encoding (the default).

constrained (*cvbr*)

Use constrained variable bit rate encoding.

compression_level (*comp*)

Set encoding algorithm complexity. Valid options are integers in the 0-10 range. 0 gives the fastest encodes but lower quality, while 10 gives the highest quality but slowest encoding. The default is 10.

frame_duration (*framesize*)

Set maximum frame size, or duration of a frame in milliseconds. The argument must be exactly the following: 2.5, 5, 10, 20, 40, 60. Smaller frame sizes achieve lower latency but less quality at a given bitrate. Sizes greater than 20ms are only interesting at fairly low bitrates. The default is 20ms.

packet_loss (*expect-loss*)

Set expected packet loss percentage. The default is 0.

fec (*n/a*)

Enable inband forward error correction. **packet_loss** must be non-zero to take advantage - frequency of FEC 'side-data' is proportional to expected packet loss. Default is disabled.

application (N.A.)

Set intended application type. Valid options are listed below:

voip

Favor improved speech intelligibility.

audio

Favor faithfulness to the input (the default).

lowdelay

Restrict to only the lowest delay modes.

cutoff (N.A.)

Set cutoff bandwidth in Hz. The argument must be exactly one of the following: 4000, 6000, 8000, 12000, or 20000, corresponding to narrowband, mediumband, wideband, super wideband, and fullband respectively. The default is 0 (cutoff disabled).

mapping_family (*mapping_family*)

Set channel mapping family to be used by the encoder. The default value of -1 uses mapping family 0 for mono and stereo inputs, and mapping family 1 otherwise. The default also disables the surround masking and LFE bandwidth optimizations in libopus, and requires that the input contains 8 channels or fewer.

Other values include 0 for mono and stereo, 1 for surround sound with masking and LFE bandwidth optimizations, and 255 for independent streams with an unspecified channel layout.

apply_phase_inv (N.A.) (requires **libopus** \geq 1.2)

If set to 0, disables the use of phase inversion for intensity stereo, improving the quality of mono downmixes, but slightly reducing normal stereo quality. The default is 1 (phase inversion enabled).

libshine

Shine Fixed-Point MP3 encoder wrapper.

Shine is a fixed-point MP3 encoder. It has a far better performance on platforms without an FPU, e.g. armel CPUs, and some phones and tablets. However, as it is more targeted on performance than quality, it is not on par with LAME and other production-grade encoders quality-wise. Also, according to the project's homepage, this encoder may not be free of bugs as the code was written a long time ago and the project was dead for at least 5 years.

This encoder only supports stereo and mono input. This is also CBR-only.

The original project (last updated in early 2007) is at <http://sourceforge.net/projects/libshine-fxp/>. We only support the updated fork by the Savonet/Liquidsoap project at <https://github.com/savonet/shine>.

Requires the presence of the libshine headers and library during configuration. You need to explicitly configure the build with "--enable-libshine".

See also **libmp3lame**.

Options

The following options are supported by the libshine wrapper. The **shineenc**-equivalent of the options are listed in parentheses.

b (-b)

Set bitrate expressed in bits/s for CBR. **shineenc -b** option is expressed in kilobits/s.

libtwolame

TwoLAME MP2 encoder wrapper.

Requires the presence of the libtwolame headers and library during configuration. You need to explicitly configure the build with "--enable-libtwolame".

Options

The following options are supported by the libtwolame wrapper. The **twolame**-equivalent options follow the FFmpeg ones and are in parentheses.

b (-b)

Set bitrate expressed in bits/s for CBR. **twolame b** option is expressed in kilobits/s. Default value is 128k.

q (-V)

Set quality for experimental VBR support. Maximum value range is from -50 to 50, useful range is from -10 to 10. The higher the value, the better the quality. This option is valid only using the **ffmpeg** command-line tool. For library interface users, use **global_quality**.

mode (--mode)

Set the mode of the resulting audio. Possible values:

auto Choose mode automatically based on the input. This is the default.

stereo

Stereo

joint_stereo

Joint stereo

dual_channel

Dual channel

mono

Mono

psymodel (--psyc-mode)

Set psychoacoustic model to use in encoding. The argument must be an integer between -1 and 4, inclusive. The higher the value, the better the quality. The default value is 3.

energy_levels (--energy)

Enable energy levels extensions when set to 1. The default value is 0 (disabled).

error_protection (--protect)

Enable CRC error protection when set to 1. The default value is 0 (disabled).

copyright (--copyright)

Set MPEG audio copyright flag when set to 1. The default value is 0 (disabled).

original (--original)

Set MPEG audio original flag when set to 1. The default value is 0 (disabled).

libvo-amrwbenc

VisualOn Adaptive Multi-Rate Wideband encoder.

Requires the presence of the libvo-amrwbenc headers and library during configuration. You need to explicitly configure the build with "--enable-libvo-amrwbenc --enable-version3".

This is a mono-only encoder. Officially it only supports 16000Hz sample rate, but you can override it by setting **strict** to **unofficial** or lower.

Options

- b** Set bitrate in bits/s. Only the following bitrates are supported, otherwise libavcodec will round to the nearest valid bitrate.

6600

8850

12650

14250

15850

18250

19850

23050

23850

- dtx** Allow discontinuous transmission (generate comfort noise) when set to 1. The default value is 0 (disabled).

libvorbis

libvorbis encoder wrapper.

Requires the presence of the libvorbisenc headers and library during configuration. You need to explicitly configure the build with "--enable-libvorbis".

Options

The following options are supported by the libvorbis wrapper. The **oggenc**-equivalent of the options are listed in parentheses.

To get a more accurate and extensive documentation of the libvorbis options, consult the libvorbisenc's and **oggenc**'s documentations. See <<http://xiph.org/vorbis/>>, <<http://wiki.xiph.org/Vorbis-tools>>, and **oggenc**(1).

b (-b)

Set bitrate expressed in bits/s for ABR. **oggenc -b** is expressed in kilobits/s.

q (-q)

Set constant quality setting for VBR. The value should be a float number in the range of -1.0 to 10.0. The higher the value, the better the quality. The default value is **3.0**.

This option is valid only using the **ffmpeg** command-line tool. For library interface users, use **global_quality**.

cutoff (*--advanced-encode-option lowpass_frequency=N*)

Set cutoff bandwidth in Hz, a value of 0 disables cutoff. **oggenc**'s related option is expressed in kHz. The default value is **0** (cutoff disabled).

minrate (-m)

Set minimum bitrate expressed in bits/s. **oggenc -m** is expressed in kilobits/s.

maxrate (-M)

Set maximum bitrate expressed in bits/s. **oggenc -M** is expressed in kilobits/s. This only has effect on ABR mode.

iblock (*--advanced-encode-option impulse_noisetune=N*)

Set noise floor bias for impulse blocks. The value is a float number from -15.0 to 0.0. A negative bias instructs the encoder to pay special attention to the crispness of transients in the encoded audio. The tradeoff for better transient response is a higher bitrate.

mjpeg

Motion JPEG encoder.

*Options***huffman**

Set the huffman encoding strategy. Possible values:

default

Use the default huffman tables. This is the default strategy.

optimal

Compute and use optimal huffman tables.

wavpack

WavPack lossless audio encoder.

Options

The equivalent options for **wavpack** command line utility are listed in parentheses.

Shared options

The following shared options are effective for this encoder. Only special notes about this particular encoder will be documented here. For the general meaning of the options, see **the Codec Options chapter**.

frame_size (*--blocksize*)

For this encoder, the range for this option is between 128 and 131072. Default is automatically decided based on sample rate and number of channel.

For the complete formula of calculating default, see *libavcodec/wavpackenc.c*.

compression_level (*-f, -h, -hh, and -x*)

Private options

joint_stereo (*-j*)

Set whether to enable joint stereo. Valid values are:

on (*1*)

Force mid/side audio encoding.

off (*0*)

Force left/right audio encoding.

auto Let the encoder decide automatically.

optimize_mono

Set whether to enable optimization for mono. This option is only effective for non-mono streams. Available values:

on enabled

off disabled

VIDEO ENCODERS

A description of some of the currently available video encoders follows.

a64_multi, a64_multi5

A64 / Commodore 64 multicolor charset encoder. "a64_multi5" is extended with 5th color (colram).

Cinepak

Cinepak aka CVID encoder. Compatible with Windows 3.1 and vintage MacOS.

Options

g *integer*

Keyframe interval. A keyframe is inserted at least every "-g" frames, sometimes sooner.

q:v *integer*

Quality factor. Lower is better. Higher gives lower bitrate. The following table lists bitrates when encoding akiyo_cif.y4m for various values of "-q:v" with "-g 100":

"-q:v 1" 1918 kb/s

"-q:v 2" 1735 kb/s

"-q:v 4" 1500 kb/s

"-q:v 10" 1041 kb/s

"-q:v 20" 826 kb/s

"-q:v 40" 553 kb/s

"-q:v 100" 394 kb/s

"-q:v 200" 312 kb/s

"-q:v 400" 266 kb/s

"-q:v 1000" 237 kb/s

max_extra_cb_iterations *integer*

Max extra codebook recalculation passes, more is better and slower.

skip_empty_cb *boolean*

Avoid wasting bytes, ignore vintage MacOS decoder.

max_strips *integer*

min_strips *integer*

The minimum and maximum number of strips to use. Wider range sometimes improves quality. More strips is generally better quality but costs more bits. Fewer strips tend to yield more keyframes. Vintage compatible is 1..3.

strip_number_adaptivity *integer*

How much number of strips is allowed to change between frames. Higher is better but slower.

GIF

GIF image/animation encoder.

Options

gifflags *integer*

Sets the flags used for GIF encoding.

offsetting

Enables picture offsetting.

Default is enabled.

transdiff

Enables transparency detection between frames.

Default is enabled.

gifimage *integer*

Enables encoding one full GIF image per frame, rather than an animated GIF.

Default value is **0**.

global_palette *integer*

Writes a palette to the global GIF header where feasible.

If disabled, every frame will always have a palette written, even if there is a global palette supplied.

Default value is **1**.

Hap

Vidvox Hap video encoder.

Options

format *integer*

Specifies the Hap format to encode.

hap
hap_alpha
hap_q

Default value is **hap**.

chunks *integer*

Specifies the number of chunks to split frames into, between 1 and 64. This permits multithreaded decoding of large frames, potentially at the cost of data-rate. The encoder may modify this value to divide frames evenly.

Default value is *1*.

compressor *integer*

Specifies the second-stage compressor to use. If set to **none**, **chunks** will be limited to 1, as chunked uncompressed frames offer no benefit.

none
snappy

Default value is **snappy**.

jpeg2000

The native jpeg 2000 encoder is lossy by default, the "-q:v" option can be used to set the encoding quality. Lossless encoding can be selected with "-pred 1".

Options

format *integer*

Can be set to either "j2k" or "jp2" (the default) that makes it possible to store non-rgb pix_fmts.

tile_width *integer*

Sets tile width. Range is 1 to 1073741824. Default is 256.

tile_height *integer*

Sets tile height. Range is 1 to 1073741824. Default is 256.

pred *integer*

Allows setting the discrete wavelet transform (DWT) type

dwt97int (Lossy)

dwt53 (Lossless)

Default is "dwt97int"

sop *boolean*

Enable this to add SOP marker at the start of each packet. Disabled by default.

eph *boolean*

Enable this to add EPH marker at the end of each packet header. Disabled by default.

prog *integer*

Sets the progression order to be used by the encoder. Possible values are:

lrp

rlcp

rpcl

pcl

cpcl

Set to "lrp" by default.

layer_rates *string*

By default, when this option is not used, compression is done using the quality metric. This option allows for compression using compression ratio. The compression ratio for each level could be specified. The compression ratio of a layer "I" species the what ratio of total file size is contained in the first "I" layers.

Example usage:

```
ffmpeg -i input.bmp -c:v jpeg2000 -layer_rates "100,10,1" output.j2k
```

This would compress the image to contain 3 layers, where the data contained in the first layer would be compressed by 1000 times, compressed by 100 in the first two layers, and shall contain all data while using all 3 layers.

librav1e

rav1e AV1 encoder wrapper.

Requires the presence of the rav1e headers and library during configuration. You need to explicitly

configure the build with "--enable-librav1e".

Options

qmax

Sets the maximum quantizer to use when using bitrate mode.

qmin

Sets the minimum quantizer to use when using bitrate mode.

qp Uses quantizer mode to encode at the given quantizer (0-255).

speed

Selects the speed preset (0-10) to encode with.

tiles Selects how many tiles to encode with.

tile-rows

Selects how many rows of tiles to encode with.

tile-columns

Selects how many columns of tiles to encode with.

rav1e-params

Set rav1e options using a list of *key=value* pairs separated by ":". See **rav1e --help** for a list of options.

For example to specify librav1e encoding options with **-rav1e-params**:

```
ffmpeg -i input -c:v librav1e -b:v 500K -rav1e-params speed=5:low_latency=true output.mp4
```

libaom-av1

libaom AV1 encoder wrapper.

Requires the presence of the libaom headers and library during configuration. You need to explicitly configure the build with "--enable-libaom".

Options

The wrapper supports the following standard libavcodec options:

- b** Set bitrate target in bits/second. By default this will use variable-bitrate mode. If **maxrate** and **minrate** are also set to the same value then it will use constant-bitrate mode, otherwise if **crf** is set as well then it will use constrained-quality mode.

g keyint_min

Set key frame placement. The GOP size sets the maximum distance between key frames; if zero the output stream will be intra-only. The minimum distance is ignored unless it is the same as the GOP size, in which case key frames will always appear at a fixed interval. Not set by default, so without this option the library has completely free choice about where to place key frames.

qmin qmax

Set minimum/maximum quantisation values. Valid range is from 0 to 63 (warning: this does not match the quantiser values actually used by AV1 - divide by four to map real quantiser values to this range). Defaults to min/max (no constraint).

minrate maxrate bufsize rc_init_occupancy

Set rate control buffering parameters. Not used if not set - defaults to unconstrained variable bitrate.

threads

Set the number of threads to use while encoding. This may require the **tiles** or **row-nt** options to also be set to actually use the specified number of threads fully. Defaults to the number of hardware threads supported by the host machine.

profile

Set the encoding profile. Defaults to using the profile which matches the bit depth and chroma subsampling of the input.

The wrapper also has some specific options:

cpu-used

Set the quality/encoding speed tradeoff. Valid range is from 0 to 8, higher numbers indicating greater speed and lower quality. The default value is 1, which will be slow and high quality.

auto-alt-ref

Enable use of alternate reference frames. Defaults to the internal default of the library.

arnr-max-frames (*frames*)

Set altref noise reduction max frame count. Default is -1.

arnr-strength (*strength*)

Set alref noise reduction filter strength. Range is -1 to 6. Default is -1.

aq-mode (*aq-mode*)

Set adaptive quantization mode. Possible values:

none (*0*)

Disabled.

variance (*1*)

Variance-based.

complexity (*2*)

Complexity-based.

cyclic (*3*)

Cyclic refresh.

tune (*tune*)

Set the distortion metric the encoder is tuned with. Default is "psnr".

psnr (*0*)**ssim** (*1*)**lag-in-frames**

Set the maximum number of frames which the encoder may keep in flight at any one time for lookahead purposes. Defaults to the internal default of the library.

error-resilience

Enable error resilience features:

default

Improve resilience against losses of whole frames.

Not enabled by default.

crf Set the quality/size tradeoff for constant-quality (no bitrate target) and constrained-quality (with maximum bitrate target) modes. Valid range is 0 to 63, higher numbers indicating lower quality and smaller output size. Only used if set; by default only the bitrate target is used.

static-thresh

Set a change threshold on blocks below which they will be skipped by the encoder. Defined in arbitrary units as a nonnegative integer, defaulting to zero (no blocks are skipped).

drop-threshold

Set a threshold for dropping frames when close to rate control bounds. Defined as a percentage of the target buffer - when the rate control buffer falls below this percentage, frames will be dropped until it has refilled above the threshold. Defaults to zero (no frames are dropped).

denoise-noise-level (*level*)

Amount of noise to be removed for grain synthesis. Grain synthesis is disabled if this option is not set or set to 0.

denoise-block-size (*pixels*)

Block size used for denoising for grain synthesis. If not set, AV1 codec uses the default value of 32.

undershoot-pct (*pct*)

Set datarate undershoot (min) percentage of the target bitrate. Range is -1 to 100. Default is -1.

overshoot-pct (*pct*)

Set datarate overshoot (max) percentage of the target bitrate. Range is -1 to 1000. Default is -1.

minsection-pct (*pct*)

Minimum percentage variation of the GOP bitrate from the target bitrate. If minsection-pct is not set, the libaomenc wrapper computes it as follows: "(minrate * 100 / bitrate)". Range is -1 to 100. Default is -1 (unset).

maxsection-pct (*pct*)

Maximum percentage variation of the GOP bitrate from the target bitrate. If maxsection-pct is not set, the libaomenc wrapper computes it as follows: "(maxrate * 100 / bitrate)". Range is -1 to 5000. Default is -1 (unset).

frame-parallel (*boolean*)

Enable frame parallel decodability features. Default is true.

tiles Set the number of tiles to encode the input video with, as columns x rows. Larger numbers allow greater parallelism in both encoding and decoding, but may decrease coding efficiency. Defaults to the minimum number of tiles required by the size of the input video (this is 1x1 (that is, a single tile) for sizes up to and including 4K).

tile-columns tile-rows

Set the number of tiles as log2 of the number of tile rows and columns. Provided for compatibility with libvpx/VP9.

row-nt (Requires libaom >= 1.0.0-759-g90a15f4f2)

Enable row based multi-threading. Disabled by default.

enable-cdef (boolean)

Enable Constrained Directional Enhancement Filter. The libaom-av1 encoder enables CDEF by default.

enable-restoration (boolean)

Enable Loop Restoration Filter. Default is true for libaom-av1.

enable-global-motion (boolean)

Enable the use of global motion for block prediction. Default is true.

enable-intrabc (boolean)

Enable block copy mode for intra block prediction. This mode is useful for screen content. Default is true.

enable-rect-partitions (boolean) (Requires libaom >= v2.0.0)

Enable rectangular partitions. Default is true.

enable-1to4-partitions (boolean) (Requires libaom >= v2.0.0)

Enable 1:4/4:1 partitions. Default is true.

enable-ab-partitions (boolean) (Requires libaom >= v2.0.0)

Enable AB shape partitions. Default is true.

enable-angle-delta (boolean) (Requires libaom >= v2.0.0)

Enable angle delta intra prediction. Default is true.

enable-cfl-intra (boolean) (Requires libaom >= v2.0.0)

Enable chroma predicted from luma intra prediction. Default is true.

enable-filter-intra (boolean) (Requires libaom >= v2.0.0)

Enable filter intra predictor. Default is true.

enable-intra-edge-filter (boolean) (Requires libaom >= v2.0.0)

Enable intra edge filter. Default is true.

enable-smooth-intra (*boolean*) (Requires libaom >= v2.0.0)

Enable smooth intra prediction mode. Default is true.

enable-paeth-intra (*boolean*) (Requires libaom >= v2.0.0)

Enable paeth predictor in intra prediction. Default is true.

enable-palette (*boolean*) (Requires libaom >= v2.0.0)

Enable palette prediction mode. Default is true.

enable-flip-idx (*boolean*) (Requires libaom >= v2.0.0)

Enable extended transform type, including FLIPADST_DCT, DCT_FLIPADST, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, IDTX, V_DCT, H_DCT, V_ADST, H_ADST, V_FLIPADST, H_FLIPADST. Default is true.

enable-tx64 (*boolean*) (Requires libaom >= v2.0.0)

Enable 64-pt transform. Default is true.

reduced-tx-type-set (*boolean*) (Requires libaom >= v2.0.0)

Use reduced set of transform types. Default is false.

use-intra-dct-only (*boolean*) (Requires libaom >= v2.0.0)

Use DCT only for INTRA modes. Default is false.

use-inter-dct-only (*boolean*) (Requires libaom >= v2.0.0)

Use DCT only for INTER modes. Default is false.

use-intra-default-tx-only (*boolean*) (Requires libaom >= v2.0.0)

Use Default-transform only for INTRA modes. Default is false.

enable-ref-frame-mvs (*boolean*) (Requires libaom >= v2.0.0)

Enable temporal mv prediction. Default is true.

enable-reduced-reference-set (*boolean*) (Requires libaom >= v2.0.0)

Use reduced set of single and compound references. Default is false.

enable-obmc (*boolean*) (Requires libaom >= v2.0.0)

Enable obmc. Default is true.

enable-dual-filter (*boolean*) (Requires libaom >= v2.0.0)

Enable dual filter. Default is true.

enable-diff-wtd-comp (*boolean*) (Requires libaom >= v2.0.0)

Enable difference-weighted compound. Default is true.

enable-dist-wtd-comp (*boolean*) (Requires libaom >= v2.0.0)

Enable distance-weighted compound. Default is true.

enable-onesided-comp (*boolean*) (Requires libaom >= v2.0.0)

Enable one sided compound. Default is true.

enable-interinter-wedge (*boolean*) (Requires libaom >= v2.0.0)

Enable interinter wedge compound. Default is true.

enable-interintra-wedge (*boolean*) (Requires libaom >= v2.0.0)

Enable interintra wedge compound. Default is true.

enable-masked-comp (*boolean*) (Requires libaom >= v2.0.0)

Enable masked compound. Default is true.

enable-interintra-comp (*boolean*) (Requires libaom >= v2.0.0)

Enable interintra compound. Default is true.

enable-smooth-interintra (*boolean*) (Requires libaom >= v2.0.0)

Enable smooth interintra mode. Default is true.

aom-params

Set libaom options using a list of *key=value* pairs separated by ":". For a list of supported options, see **aomenc --help** under the section "AV1 Specific Options".

For example to specify libaom encoding options with **-aom-params**:

```
ffmpeg -i input -c:v libaom-av1 -b:v 500K -aom-params tune=psnr:enable-tpl-model=1 output.mp4
```

libsvtav1

SVT-AV1 encoder wrapper.

Requires the presence of the SVT-AV1 headers and library during configuration. You need to explicitly configure the build with "--enable-libsvtav1".

*Options***profile**

Set the encoding profile.

main

high

professional

level

Set the operating point level. For example: '4.0'

hielevel

Set the Hierarchical prediction levels.

3level

4level

This is the default.

tier Set the operating point tier.

main

This is the default.

high

qmax

Set the maximum quantizer to use when using a bitrate mode.

qmin

Set the minimum quantizer to use when using a bitrate mode.

crf Constant rate factor value used in crf rate control mode (0-63).

qp Set the quantizer used in cqp rate control mode (0-63).

sc_detection

Enable scene change detection.

la_depth

Set number of frames to look ahead (0-120).

preset

Set the quality-speed tradeoff, in the range 0 to 13. Higher values are faster but lower quality.

tile_rows

Set log2 of the number of rows of tiles to use (0-6).

tile_columns

Set log2 of the number of columns of tiles to use (0-4).

svtav1-params

Set SVT-AV1 options using a list of *key=value* pairs separated by ":". See the SVT-AV1 encoder user guide for a list of accepted parameters.

libjxl

libjxl JPEG XL encoder wrapper.

Requires the presence of the libjxl headers and library during configuration. You need to explicitly configure the build with "--enable-libjxl".

Options

The libjxl wrapper supports the following options:

distance

Set the target Butteraugli distance. This is a quality setting: lower distance yields higher quality, with distance=1.0 roughly comparable to libjpeg Quality 90 for photographic content. Setting distance=0.0 yields true lossless encoding. Valid values range between 0.0 and 15.0, and sane values rarely exceed 5.0. Setting distance=0.1 usually attains transparency for most input. The default is 1.0.

effort

Set the encoding effort used. Higher effort values produce more consistent quality and usually produces a better quality/bpp curve, at the cost of more CPU time required. Valid values range from 1 to 9, and the default is 7.

modular

Force the encoder to use Modular mode instead of choosing automatically. The default is to use VarDCT for lossy encoding and Modular for lossless. VarDCT is generally superior to Modular for lossy encoding but does not support lossless encoding.

libkvazaar

Kvazaar H.265/HEVC encoder.

Requires the presence of the libkvazaar headers and library during configuration. You need to explicitly configure the build with **--enable-libkvazaar**.

Options

b Set target video bitrate in bit/s and enable rate control.

kvazaar-params

Set kvazaar parameters as a list of *name=value* pairs separated by commas (.). See kvazaar documentation for a list of options.

libopenh264

Cisco libopenh264 H.264/MPEG-4 AVC encoder wrapper.

This encoder requires the presence of the libopenh264 headers and library during configuration. You need to explicitly configure the build with "--enable-libopenh264". The library is detected using **pkg-config**.

For more information about the library see <<http://www.openh264.org>>.

Options

The following FFmpeg global options affect the configurations of the libopenh264 encoder.

b Set the bitrate (as a number of bits per second).

g Set the GOP size.

maxrate

Set the max bitrate (as a number of bits per second).

flags +global_header

Set global header in the bitstream.

slices

Set the number of slices, used in parallelized encoding. Default value is 0. This is only used when **slice_mode** is set to **fixed**.

loopfilter

Enable loop filter, if set to 1 (automatically enabled). To disable set a value of 0.

profile

Set profile restrictions. If set to the value of **main** enable CABAC (set the "SEncParamExt.iEntropyCodingModeFlag" flag to 1).

max_nal_size

Set maximum NAL size in bytes.

allow_skip_frames

Allow skipping frames to hit the target bitrate if set to 1.

libtheora

libtheora Theora encoder wrapper.

Requires the presence of the libtheora headers and library during configuration. You need to explicitly configure the build with "--enable-libtheora".

For more information about the libtheora project see <<http://www.theora.org/>>.

Options

The following global options are mapped to internal libtheora options which affect the quality and the bitrate of the encoded stream.

b Set the video bitrate in bit/s for CBR (Constant Bit Rate) mode. In case VBR (Variable Bit Rate) mode is enabled this option is ignored.

flags

Used to enable constant quality mode (VBR) encoding through the **qscale** flag, and to enable the "pass1" and "pass2" modes.

g Set the GOP size.

global_quality

Set the global quality as an integer in lambda units.

Only relevant when VBR mode is enabled with "flags +qscale". The value is converted to QP units by dividing it by "FF_QP2LAMBDA", clipped in the [0 - 10] range, and then multiplied by

6.3 to get a value in the native libtheora range [0-63]. A higher value corresponds to a higher quality.

- q** Enable VBR mode when set to a non-negative value, and set constant quality value as a double floating point value in QP units.

The value is clipped in the [0-10] range, and then multiplied by 6.3 to get a value in the native libtheora range [0-63].

This option is valid only using the **ffmpeg** command-line tool. For library interface users, use **global_quality**.

Examples

- ⊕ Set maximum constant quality (VBR) encoding with **ffmpeg**:

```
ffmpeg -i INPUT -codec:v libtheora -q:v 10 OUTPUT.ogg
```

- ⊕ Use **ffmpeg** to convert a CBR 1000 kbps Theora video stream:

```
ffmpeg -i INPUT -codec:v libtheora -b:v 1000k OUTPUT.ogg
```

libvpx

VP8/VP9 format supported through libvpx.

Requires the presence of the libvpx headers and library during configuration. You need to explicitly configure the build with "--enable-libvpx".

Options

The following options are supported by the libvpx wrapper. The **vpxenc**-equivalent options or values are listed in parentheses for easy migration.

To reduce the duplication of documentation, only the private options and some others requiring special attention are documented here. For the documentation of the undocumented generic options, see **the Codec Options chapter**.

To get more documentation of the libvpx options, invoke the command **ffmpeg -h encoder=libvpx**, **ffmpeg -h encoder=libvpx-vp9** or **vpxenc --help**. Further information is available in the libvpx API documentation.

b (*target-bitrate*)

Set bitrate in bits/s. Note that FFMpeg's **b** option is expressed in bits/s, while **vp8enc**'s **target-bitrate** is in kilobits/s.

g (*kf-max-dist*)**keyint_min** (*kf-min-dist*)**qmin** (*min-q*)

Minimum (Best Quality) Quantizer.

qmax (*max-q*)

Maximum (Worst Quality) Quantizer. Can be changed per-frame.

bufsize (*buf-sz, buf-optimal-sz*)

Set ratecontrol buffer size (in bits). Note **vp8enc**'s options are specified in milliseconds, the libvpx wrapper converts this value as follows: "**buf-sz** = **bufsize** * 1000 / **bitrate**", "**buf-optimal-sz** = **bufsize** * 1000 / **bitrate** * 5 / 6".

rc_init_occupancy (*buf-initial-sz*)

Set number of bits which should be loaded into the rc buffer before decoding starts. Note **vp8enc**'s option is specified in milliseconds, the libvpx wrapper converts this value as follows: "**rc_init_occupancy** * 1000 / **bitrate**".

undershoot-pct

Set datarate undershoot (min) percentage of the target bitrate.

overshoot-pct

Set datarate overshoot (max) percentage of the target bitrate.

skip_threshold (*drop-frame*)**qcomp** (*bias-pct*)**maxrate** (*maxsection-pct*)

Set GOP max bitrate in bits/s. Note **vp8enc**'s option is specified as a percentage of the target bitrate, the libvpx wrapper converts this value as follows: "(**maxrate** * 100 / **bitrate**)".

minrate (*minsection-pct*)

Set GOP min bitrate in bits/s. Note **vp8enc**'s option is specified as a percentage of the target bitrate, the libvpx wrapper converts this value as follows: "(**minrate** * 100 / **bitrate**)".

minrate, maxrate, b *end-usage=cbr*

"(**minrate** == **maxrate** == **bitrate**)".

crf (*end-usage=cq, cq-level*)

tune (*tune*)

psnr (*psnr*)

ssim (*ssim*)

quality, deadline (*deadline*)

best Use best quality deadline. Poorly named and quite slow, this option should be avoided as it may give worse quality output than good.

good

Use good quality deadline. This is a good trade-off between speed and quality when used with the **cpu-used** option.

realtime

Use realtime quality deadline.

speed, cpu-used (*cpu-used*)

Set quality/speed ratio modifier. Higher values speed up the encode at the cost of quality.

nr (*noise-sensitivity*)

static-thresh

Set a change threshold on blocks below which they will be skipped by the encoder.

slices (*token-parts*)

Note that Ffmpeg's **slices** option gives the total number of partitions, while **vpxenc**'s **token-parts** is given as "log2(partitions)".

max-intra-rate

Set maximum I-frame bitrate as a percentage of the target bitrate. A value of 0 means unlimited.

force_key_frames

"VPX_EFLAG_FORCE_KF"

Alternate reference frame related

auto-alt-ref

Enable use of alternate reference frames (2-pass only). Values greater than 1 enable multi-layer alternate reference frames (VP9 only).

arnr-maxframes

Set altref noise reduction max frame count.

arnr-type

Set altref noise reduction filter type: backward, forward, centered.

arnr-strength

Set altref noise reduction filter strength.

rc-lookahead, lag-in-frames (*lag-in-frames*)

Set number of frames to look ahead for frametype and ratecontrol.

min-gf-interval

Set minimum golden/alternate reference frame interval (VP9 only).

error-resilient

Enable error resiliency features.

sharpness *integer*

Increase sharpness at the expense of lower PSNR. The valid range is [0, 7].

ts-parameters

Sets the temporal scalability configuration using a :-separated list of key=value pairs. For example, to specify temporal scalability parameters with "ffmpeg":

```
ffmpeg -i INPUT -c:v libvpx -ts-parameters ts_number_layers=3:\
ts_target_bitrate=250,500,1000:ts_rate_decimator=4,2,1:\
ts_periodicity=4:ts_layer_id=0,2,1,2:ts_layering_mode=3 OUTPUT
```

Below is a brief explanation of each of the parameters, please refer to "struct vpx_codec_enc_cfg" in "vpx/vpx_encoder.h" for more details.

ts_number_layers

Number of temporal coding layers.

ts_target_bitrate

Target bitrate for each temporal layer (in kbps). (bitrate should be inclusive of the lower temporal layer).

ts_rate_decimator

Frame rate decimation factor for each temporal layer.

ts_periodicity

Length of the sequence defining frame temporal layer membership.

ts_layer_id

Template defining the membership of frames to temporal layers.

ts_layering_mode

(optional) Selecting the temporal structure from a set of pre-defined temporal layering modes. Currently supports the following options.

- 0** No temporal layering flags are provided internally, relies on flags being passed in using "metadata" field in "AVFrame" with following keys.

vp8-flags

Sets the flags passed into the encoder to indicate the referencing scheme for the current frame. Refer to function "vpx_codec_encode" in "vpx/vpx_encoder.h" for more details.

temporal_id

Explicitly sets the temporal id of the current frame to encode.

- 2** Two temporal layers. 0-1...
- 3** Three temporal layers. 0-2-1-2...; with single reference frame.
- 4** Same as option "3", except there is a dependency between the two temporal layer 2 frames within the temporal period.

VP9-specific options**lossless**

Enable lossless mode.

tile-columns

Set number of tile columns to use. Note this is given as " $\log_2(\text{tile_columns})$ ". For example, 8 tile columns would be requested by setting the **tile-columns** option to 3.

tile-rows

Set number of tile rows to use. Note this is given as " $\log_2(\text{tile_rows})$ ". For example, 4 tile rows would be requested by setting the **tile-rows** option to 2.

frame-parallel

Enable frame parallel decodability features.

aq-mode

Set adaptive quantization mode (0: off (default), 1: variance 2: complexity, 3: cyclic refresh, 4: equator360).

colorspace *color-space*

Set input color space. The VP9 bitstream supports signaling the following colorspace:

rgb *sRGB*

bt709 *bt709*

unspecified *unknown*

bt470bg *bt601*

smpte170m *smpte170*

smpte240m *smpte240*

bt2020_ncl *bt2020*

row-nt *boolean*

Enable row based multi-threading.

tune-content

Set content type: default (0), screen (1), film (2).

corpus-complexity

Corpus VBR mode is a variant of standard VBR where the complexity distribution midpoint is passed in rather than calculated for a specific clip or chunk.

The valid range is [0, 10000]. 0 (default) uses standard VBR.

enable-tpl *boolean*

Enable temporal dependency model.

ref-frame-config

Using per-frame metadata, set members of the structure "vpx_svc_ref_frame_config_t" in "vpx/vp8cx.h" to fine-control referencing schemes and frame buffer management. Use a :-separated list of key=value pairs. For example,

```
av_dict_set(&av_frame->metadata, "ref-frame-config", \
"rfc_update_buffer_slot=7:rfc_lst_fb_idx=0:rfc_gld_fb_idx=1:rfc_alt_fb_idx=2:rfc_reference_last=0:rfc_
```

rfc_update_buffer_slot

Indicates the buffer slot number to update

rfc_update_last

Indicates whether to update the LAST frame

rfc_update_golden

Indicates whether to update GOLDEN frame

rfc_update_alt_ref

Indicates whether to update ALT_REF frame

rfc_lst_fb_idx

LAST frame buffer index

rfc_gld_fb_idx

GOLDEN frame buffer index

rfc_alt_fb_idx

ALT_REF frame buffer index

rfc_reference_last

Indicates whether to reference LAST frame

rfc_reference_golden

Indicates whether to reference GOLDEN frame

rfc_reference_alt_ref

Indicates whether to reference ALT_REF frame

rfc_reference_duration

Indicates frame duration

For more information about libvpx see: <http://www.webmproject.org/>

libwebp

libwebp WebP Image encoder wrapper

libwebp is Google's official encoder for WebP images. It can encode in either lossy or lossless mode. Lossy images are essentially a wrapper around a VP8 frame. Lossless images are a separate codec developed by Google.

Pixel Format

Currently, libwebp only supports YUV420 for lossy and RGB for lossless due to limitations of the format and libwebp. Alpha is supported for either mode. Because of API limitations, if RGB is passed in when encoding lossy or YUV is passed in for encoding lossless, the pixel format will automatically be converted using functions from libwebp. This is not ideal and is done only for convenience.

Options

-lossless *boolean*

Enables/Disables use of lossless mode. Default is 0.

-compression_level *integer*

For lossy, this is a quality/speed tradeoff. Higher values give better quality for a given size at the cost of increased encoding time. For lossless, this is a size/speed tradeoff. Higher values give smaller size at the cost of increased encoding time. More specifically, it controls the number of extra algorithms and compression tools used, and varies the combination of these tools. This maps to the *method* option in libwebp. The valid range is 0 to 6. Default is 4.

-quality *float*

For lossy encoding, this controls image quality. For lossless encoding, this controls the effort and time spent in compression. Range is 0 to 100. Default is 75.

-preset *type*

Configuration preset. This does some automatic settings based on the general type of the image.

none

Do not use a preset.

default

Use the encoder default.

picture

Digital picture, like portrait, inner shot

photo

Outdoor photograph, with natural lighting

drawing

Hand or line drawing, with high-contrast details

icon Small-sized colorful images

text Text-like

libx264, libx264rgb

x264 H.264/MPEG-4 AVC encoder wrapper.

This encoder requires the presence of the libx264 headers and library during configuration. You need to explicitly configure the build with "--enable-libx264".

libx264 supports an impressive number of features, including 8x8 and 4x4 adaptive spatial transform, adaptive B-frame placement, CAVLC/CABAC entropy coding, interlacing (MBAFF), lossless mode, psy optimizations for detail retention (adaptive quantization, psy-RD, psy-trellis).

Many libx264 encoder options are mapped to FFmpeg global codec options, while unique encoder options are provided through private options. Additionally the **x264opts** and **x264-params** private options allows one to pass a list of key=value tuples as accepted by the libx264 "x264_param_parse" function.

The x264 project website is at <<http://www.videolan.org/developers/x264.html>>.

The libx264rgb encoder is the same as libx264, except it accepts packed RGB pixel formats as input instead of YUV.

Supported Pixel Formats

x264 supports 8- to 10-bit color spaces. The exact bit depth is controlled at x264's configure time.

Options

The following options are supported by the libx264 wrapper. The **x264**-equivalent options or values are listed in parentheses for easy migration.

To reduce the duplication of documentation, only the private options and some others requiring special attention are documented here. For the documentation of the undocumented generic options, see **the Codec Options chapter**.

To get a more accurate and extensive documentation of the libx264 options, invoke the command **x264 --fullhelp** or consult the libx264 documentation.

b (*bitrate*)

Set bitrate in bits/s. Note that FFmpeg's **b** option is expressed in bits/s, while **x264**'s **bitrate** is in kilobits/s.

bf (*bframes*)**g** (*keyint*)**qmin** (*qpmin*)

Minimum quantizer scale.

qmax (*qpmax*)

Maximum quantizer scale.

qdiff (*qpstep*)

Maximum difference between quantizer scales.

qblur (*qblur*)

Quantizer curve blur

qcomp (*qcomp*)

Quantizer curve compression factor

refs (*ref*)

Number of reference frames each P-frame can use. The range is from 0-16.

sc_threshold (*scenecut*)

Sets the threshold for the scene change detection.

trellis (*trellis*)

Performs Trellis quantization to increase efficiency. Enabled by default.

nr (*nr*)**me_range** (*merange*)

Maximum range of the motion search in pixels.

me_method (*me*)

Set motion estimation method. Possible values in the decreasing order of speed:

dia (*dia*)**epzs** (*dia*)

Diamond search with radius 1 (fastest). **epzs** is an alias for **dia**.

hex (*hex*)

Hexagonal search with radius 2.

umh (*umh*)

Uneven multi-hexagon search.

esa (*esa*)

Exhaustive search.

tesa (*tesa*)

Hadamard exhaustive search (slowest).

forced-idr

Normally, when forcing a I-frame type, the encoder can select any type of I-frame. This option forces it to choose an IDR-frame.

subq (*subme*)

Sub-pixel motion estimation method.

b_strategy (*b-adapt*)

Adaptive B-frame placement decision algorithm. Use only on first-pass.

keyint_min (*min-keyint*)

Minimum GOP size.

coder

Set entropy encoder. Possible values:

ac Enable CABAC.

vlc Enable CAVLC and disable CABAC. It generates the same effect as **x264**'s **--no-cabac** option.

cmp Set full pixel motion estimation comparison algorithm. Possible values:

chroma

Enable chroma in motion estimation.

sad Ignore chroma in motion estimation. It generates the same effect as **x264**'s **--no-chroma-me** option.

threads (*threads*)

Number of encoding threads.

thread_type

Set multithreading technique. Possible values:

slice

Slice-based multithreading. It generates the same effect as **x264**'s **--sliced-threads** option.

frame

Frame-based multithreading.

flags

Set encoding flags. It can be used to disable closed GOP and enable open GOP by setting it to "-cgot". The result is similar to the behavior of **x264**'s **--open-gop** option.

rc_init_occupancy (*vbv-init*)**preset** (*preset*)

Set the encoding preset.

tune (*tune*)

Set tuning of the encoding params.

profile (*profile*)

Set profile restrictions.

fastfirstpass

Enable fast settings when encoding first pass, when set to 1. When set to 0, it has the same effect of **x264**'s **--slow-firstpass** option.

crf (*crf*)

Set the quality for constant quality mode.

crf_max (*crf-max*)

In CRF mode, prevents VBV from lowering quality beyond this point.

qp (*qp*)

Set constant quantization rate control method parameter.

aq-mode (*aq-mode*)

Set AQ method. Possible values:

none (0)

Disabled.

variance (1)

Variance AQ (complexity mask).

autovariance (2)

Auto-variance AQ (experimental).

aq-strength (*aq-strength*)

Set AQ strength, reduce blocking and blurring in flat and textured areas.

psy Use psychovisual optimizations when set to 1. When set to 0, it has the same effect as **x264**'s **--no-psy** option.

psy-rd (*psy-rd*)

Set strength of psychovisual optimization, in *psy-rd:psy-trellis* format.

rc-lookahead (*rc-lookahead*)

Set number of frames to look ahead for frametype and ratecontrol.

weightb

Enable weighted prediction for B-frames when set to 1. When set to 0, it has the same effect as **x264**'s **--no-weightb** option.

weightp (*weightp*)

Set weighted prediction method for P-frames. Possible values:

none (0)

Disabled

simple (1)

Enable only weighted refs

smart (2)

Enable both weighted refs and duplicates

ssim (*ssim*)

Enable calculation and printing SSIM stats after the encoding.

intra-refresh (*intra-refresh*)

Enable the use of Periodic Intra Refresh instead of IDR frames when set to 1.

avcintra-class (*class*)

Configure the encoder to generate AVC-Intra. Valid values are 50,100 and 200

bluray-compat (*bluray-compat*)

Configure the encoder to be compatible with the bluray standard. It is a shorthand for setting "bluray-compat=1 force-cfr=1".

b-bias (*b-bias*)

Set the influence on how often B-frames are used.

b-pyramid (*b-pyramid*)

Set method for keeping of some B-frames as references. Possible values:

none (*none*)

Disabled.

strict (*strict*)

Strictly hierarchical pyramid.

normal (*normal*)

Non-strict (not Blu-ray compatible).

mixed-refs

Enable the use of one reference per partition, as opposed to one reference per macroblock when set to 1. When set to 0, it has the same effect as **x264**'s **--no-mixed-refs** option.

8x8dct

Enable adaptive spatial transform (high profile 8x8 transform) when set to 1. When set to 0, it has the same effect as **x264**'s **--no-8x8dct** option.

fast-pskip

Enable early SKIP detection on P-frames when set to 1. When set to 0, it has the same effect as **x264**'s **--no-fast-pskip** option.

aud (*aud*)

Enable use of access unit delimiters when set to 1.

mbtree

Enable use macroblock tree ratecontrol when set to 1. When set to 0, it has the same effect as **x264**'s **--no-mbtree** option.

deblock (*deblock*)

Set loop filter parameters, in *alpha:beta* form.

cplxblur (*cplxblur*)

Set fluctuations reduction in QP (before curve compression).

partitions (*partitions*)

Set partitions to consider as a comma-separated list of. Possible values in the list:

p8x8

8x8 P-frame partition.

p4x4

4x4 P-frame partition.

b8x8

4x4 B-frame partition.

i8x8

8x8 I-frame partition.

i4x4

4x4 I-frame partition. (Enabling **p4x4** requires **p8x8** to be enabled. Enabling **i8x8** requires adaptive spatial transform (**8x8dct** option) to be enabled.)

none (*none*)

Do not consider any partitions.

all (*all*)

Consider every partition.

direct-pred (*direct*)

Set direct MV prediction mode. Possible values:

none (*none*)

Disable MV prediction.

spatial (*spatial*)

Enable spatial predicting.

temporal (*temporal*)

Enable temporal predicting.

auto (*auto*)

Automatically decided.

slice-max-size (*slice-max-size*)

Set the limit of the size of each slice in bytes. If not specified but RTP payload size (**ps**) is specified, that is used.

stats (*stats*)

Set the file name for multi-pass stats.

nal-hrd (*nal-hrd*)

Set signal HRD information (requires **vbv-buFSIZE** to be set). Possible values:

none (*none*)

Disable HRD information signaling.

vbr (*vbr*)

Variable bit rate.

cbr (*cbr*)

Constant bit rate (not allowed in MP4 container).

x264opts (**N.A.**)

Set any x264 option, see **x264 --fullhelp** for a list.

Argument is a list of *key=value* couples separated by ":". In *filter* and *psy-rd* options that use ":" as a separator themselves, use "," instead. They accept it as well since long ago but this is kept undocumented for some reason.

For example to specify libx264 encoding options with **ffmpeg**:

```
ffmpeg -i foo.mpg -c:v libx264 -x264opts keyint=123:min-keyint=20 -an out.mkv
```

a53cc *boolean*

Import closed captions (which must be ATSC compatible format) into output. Only the mpeg2 and h264 decoders provide these. Default is 1 (on).

udu_sei *boolean*

Import user data unregistered SEI if available into output. Default is 0 (off).

x264-params (N.A.)

Override the x264 configuration using a :-separated list of key=value parameters.

This option is functionally the same as the **x264opts**, but is duplicated for compatibility with the Libav fork.

For example to specify libx264 encoding options with **ffmpeg**:

```
ffmpeg -i INPUT -c:v libx264 -x264-params level=30:bframes=0:weightp=0:\
cabac=0:ref=1:vbv-maxrate=768:vbv-bufsize=2000:analyse=all:me=umh:\
no-fast-pskip=1:subq=6:8x8dct=0:trellis=0 OUTPUT
```

Encoding ffpresets for common usages are provided so they can be used with the general presets system (e.g. passing the **pre** option).

libx265

x265 H.265/HEVC encoder wrapper.

This encoder requires the presence of the libx265 headers and library during configuration. You need to explicitly configure the build with **--enable-libx265**.

Options

b Sets target video bitrate.

bf

g Set the GOP size.

keyint_min

Minimum GOP size.

refs Number of reference frames each P-frame can use. The range is from *1-16*.

preset

Set the x265 preset.

tune Set the x265 tune parameter.

profile

Set profile restrictions.

crf Set the quality for constant quality mode.

qp Set constant quantization rate control method parameter.

qmin

Minimum quantizer scale.

qmax

Maximum quantizer scale.

qdiff

Maximum difference between quantizer scales.

qblur

Quantizer curve blur

qcomp

Quantizer curve compression factor

i_qfactor

b_qfactor

forced-idr

Normally, when forcing a I-frame type, the encoder can select any type of I-frame. This option forces it to choose an IDR-frame.

udu_sei *boolean*

Import user data unregistered SEI if available into output. Default is 0 (off).

x265-params

Set x265 options using a list of *key=value* couples separated by ":". See **x265 --help** for a list of

options.

For example to specify libx265 encoding options with **-x265-params**:

```
ffmpeg -i input -c:v libx265 -x265-params crf=26:psy-rd=1 output.mp4
```

libxavs2

xavs2 AVS2-P2/IEEE1857.4 encoder wrapper.

This encoder requires the presence of the libxavs2 headers and library during configuration. You need to explicitly configure the build with **--enable-libxavs2**.

The following standard libavcodec options are used:

- ⊕ **b / bit_rate**
- ⊕ **g / gop_size**
- ⊕ **bf / max_b_frames**

The encoder also has its own specific options:

Options

lcu_row_threads

Set the number of parallel threads for rows from 1 to 8 (default 5).

initial_qp

Set the xavs2 quantization parameter from 1 to 63 (default 34). This is used to set the initial qp for the first frame.

qp Set the xavs2 quantization parameter from 1 to 63 (default 34). This is used to set the qp value under constant-QP mode.

max_qp

Set the max qp for rate control from 1 to 63 (default 55).

min_qp

Set the min qp for rate control from 1 to 63 (default 20).

speed_level

Set the Speed level from 0 to 9 (default 0). Higher is better but slower.

log_level

Set the log level from -1 to 3 (default 0). -1: none, 0: error, 1: warning, 2: info, 3: debug.

xavs2-params

Set xavs2 options using a list of *key=value* couples separated by ":".

For example to specify libxavs2 encoding options with **-xavs2-params**:

```
ffmpeg -i input -c:v libxavs2 -xavs2-params RdoqLevel=0 output.avc2
```

libxvid

Xvid MPEG-4 Part 2 encoder wrapper.

This encoder requires the presence of the libxvidcore headers and library during configuration. You need to explicitly configure the build with "--enable-libxvid --enable-gpl".

The native "mpeg4" encoder supports the MPEG-4 Part 2 format, so users can encode to this format without this library.

Options

The following options are supported by the libxvid wrapper. Some of the following options are listed but are not documented, and correspond to shared codec options. See **the Codec Options chapter** for their documentation. The other shared options which are not listed have no effect for the libxvid encoder.

b

g

qmin

qmax

mpeg_quant

threads

bf

b_qfactor

b_qoffset

flags

Set specific encoding flags. Possible values:

mv4

Use four motion vector by macroblock.

aic Enable high quality AC prediction.

gray

Only encode grayscale.

gmc Enable the use of global motion compensation (GMC).

qpel Enable quarter-pixel motion compensation.

cgop

Enable closed GOP.

global_header

Place global headers in extradata instead of every keyframe.

trellis**me_method**

Set motion estimation method. Possible values in decreasing order of speed and increasing order of quality:

zero Use no motion estimation (default).

phods**x1**

log Enable advanced diamond zonal search for 16x16 blocks and half-pixel refinement for 16x16 blocks. **x1** and **log** are aliases for **phods**.

epzs

Enable all of the things described above, plus advanced diamond zonal search for 8x8 blocks, half-pixel refinement for 8x8 blocks, and motion estimation on chroma planes.

full Enable all of the things described above, plus extended 16x16 and 8x8 blocks search.

mbd

Set macroblock decision algorithm. Possible values in the increasing order of quality:

simple

Use macroblock comparing function algorithm (default).

bits Enable rate distortion-based half pixel and quarter pixel refinement for 16x16 blocks.

rd Enable all of the things described above, plus rate distortion-based half pixel and quarter pixel refinement for 8x8 blocks, and rate distortion-based search using square pattern.

lumi_aq

Enable lumi masking adaptive quantization when set to 1. Default is 0 (disabled).

variance_aq

Enable variance adaptive quantization when set to 1. Default is 0 (disabled).

When combined with **lumi_aq**, the resulting quality will not be better than any of the two specified individually. In other words, the resulting quality will be the worse one of the two effects.

ssim

Set structural similarity (SSIM) displaying method. Possible values:

off Disable displaying of SSIM information.

avg Output average SSIM at the end of encoding to stdout. The format of showing the average SSIM is:

```
Average SSIM: %f
```

For users who are not familiar with C, %f means a float number, or a decimal (e.g. 0.939232).

frame

Output both per-frame SSIM data during encoding and average SSIM at the end of encoding to stdout. The format of per-frame information is:

```
SSIM: avg: %1.3f min: %1.3f max: %1.3f
```

For users who are not familiar with C, %1.3f means a float number rounded to 3 digits after the dot (e.g. 0.932).

ssim_acc

Set SSIM accuracy. Valid options are integers within the range of 0-4, while 0 gives the most

accurate result and 4 computes the fastest.

MediaFoundation

This provides wrappers to encoders (both audio and video) in the MediaFoundation framework. It can access both SW and HW encoders. Video encoders can take input in either of nv12 or yuv420p form (some encoders support both, some support only either - in practice, nv12 is the safer choice, especially among HW encoders).

mpeg2

MPEG-2 video encoder.

Options

profile

Select the mpeg2 profile to encode:

422

high

ss Spatially Scalable

snr SNR Scalable

main

simple

level

Select the mpeg2 level to encode:

high

high1440

main

low

seq_disp_ext *integer*

Specifies if the encoder should write a sequence_display_extension to the output.

-1

auto Decide automatically to write it or not (this is the default) by checking if the data to be written is different from the default or unspecified values.

0

never

Never write it.

1

always

Always write it.

video_format *integer*

Specifies the video_format written into the sequence display extension indicating the source of the video pictures. The default is **unspecified**, can be **component**, **pal**, **ntsc**, **secam** or **mac**. For maximum compatibility, use **component**.

a53cc *boolean*

Import closed captions (which must be ATSC compatible format) into output. Default is 1 (on).

png

PNG image encoder.

Private options

dpi *integer*

Set physical density of pixels, in dots per inch, unset by default

dpm *integer*

Set physical density of pixels, in dots per meter, unset by default

ProRes

Apple ProRes encoder.

FFmpeg contains 2 ProRes encoders, the prores-aw and prores-ks encoder. The used encoder can be chosen with the "-vcodec" option.

Private Options for prores-ks

profile *integer*

Select the ProRes profile to encode

proxy

lt

standard

hq

4444

4444xq

quant_mat *integer*

Select quantization matrix.

auto

default

proxy

lt

standard

hq

If set to *auto*, the matrix matching the profile will be picked. If not set, the matrix providing the highest quality, *default*, will be picked.

bits_per_mb *integer*

How many bits to allot for coding one macroblock. Different profiles use between 200 and 2400 bits per macroblock, the maximum is 8000.

mbs_per_slice *integer*

Number of macroblocks in each slice (1-8); the default value (8) should be good in almost all situations.

vendor *string*

Override the 4-byte vendor ID. A custom vendor ID like *apl0* would claim the stream was produced by the Apple encoder.

alpha_bits *integer*

Specify number of bits for alpha component. Possible values are *0*, *8* and *16*. Use *0* to disable alpha plane coding.

Speed considerations

In the default mode of operation the encoder has to honor frame constraints (i.e. not produce frames with size bigger than requested) while still making output picture as good as possible. A frame containing a lot of small details is harder to compress and the encoder would spend more time searching for appropriate quantizers for each slice.

Setting a higher **bits_per_mb** limit will improve the speed.

For the fastest encoding speed set the **qscale** parameter (4 is the recommended value) and do not set a size constraint.

QSV Encoders

The family of Intel QuickSync Video encoders (MPEG-2, H.264, HEVC, JPEG/MJPEG and VP9)

Ratecontrol Method

The ratecontrol method is selected as follows:

- ⊕ When **global_quality** is specified, a quality-based mode is used. Specifically this means either
 - *CQP* - constant quantizer scale, when the **qscale** codec flag is also set (the **-qscale** ffmpeg option).
 - *LA_ICQ* - intelligent constant quality with lookahead, when the **look_ahead** option is also set.
 - *ICQ* -- intelligent constant quality otherwise. For the ICQ modes, global quality range is 1 to 51, with 1 being the best quality.
- ⊕ Otherwise, a bitrate-based mode is used. For all of those, you should specify at least the desired average bitrate with the **b** option.
 - *LA* - VBR with lookahead, when the **look_ahead** option is specified.
 - *VCM* - video conferencing mode, when the **vcm** option is set.
 - *CBR* - constant bitrate, when **maxrate** is specified and equal to the average bitrate.
 - *VBR* - variable bitrate, when **maxrate** is specified, but is higher than the average bitrate.
 - *AVBR* - average VBR mode, when **maxrate** is not specified, both **avbr_accuracy** and **avbr_convergence** are set to non-zero. This mode is available for H264 and HEVC on Windows.

Note that depending on your system, a different mode than the one you specified may be selected by the encoder. Set the verbosity level to *verbose* or higher to see the actual settings used by the QSV runtime.

Global Options -> MSDK Options

Additional libavcodec global options are mapped to MSDK options as follows:

- ⊕ **g/gop_size -> GopPicSize**
- ⊕ **bf/max_b_frames+1 -> GopRefDist**
- ⊕ **rc_init_occupancy/rc_initial_buffer_occupancy -> InitialDelayInKB**
- ⊕ **slices -> NumSlice**
- ⊕ **refs -> NumRefFrame**
- ⊕ **b_strategy/b_frame_strategy -> BRefType**
- ⊕ **cgop/CLOSED_GOP** codec flag -> **GopOptFlag**
- ⊕ For the *CQP* mode, the **i_qfactor/i_qoffset** and **b_qfactor/b_qoffset** set the difference between *QPP* and *QPI*, and *QPP* and *QPB* respectively.
- ⊕ Setting the **coder** option to the value *vlc* will make the H.264 encoder use CAVLC instead of CABAC.

Common Options

Following options are used by all qsv encoders.

async_depth

Specifies how many asynchronous operations an application performs before the application explicitly synchronizes the result. If zero, the value is not specified.

preset

This option itemizes a range of choices from veryfast (best speed) to veryslow (best quality).

veryfast
faster
fast
medium
slow

slower

veryslow

forced_idr

Forcing I frames as IDR frames.

low_power

For encoders set this flag to ON to reduce power consumption and GPU usage.

Runtime Options

Following options can be used during qsv encoding.

global_quality

i_quant_factor

i_quant_offset

b_quant_factor

b_quant_offset

Supported in h264_qsv and hevc_qsv. Change these value to reset qsv codec's qp configuration.

max_frame_size

Supported in h264_qsv and hevc_qsv. Change this value to reset qsv codec's MaxFrameSize configuration.

gop_size

Change this value to reset qsv codec's gop configuration.

int_ref_type

int_ref_cycle_size

int_ref_qp_delta

int_ref_cycle_dist

Supported in h264_qsv and hevc_qsv. Change these value to reset qsv codec's Intra Refresh configuration.

qmax

qmin

max_qp_i

min_qp_i

max_qp_p

min_qp_p

max_qp_b

min_qp_b

Supported in h264_qsv. Change these value to reset qsv codec's max/min qp configuration.

low_delay_brc

Supported in h264_qsv and hevc_qsv. Change this value to reset qsv codec's low_delay_brc configuration.

framerate

Change this value to reset qsv codec's framerate configuration.

*bit_rate**rc_buffer_size**rc_initial_buffer_occupancy**rc_max_rate*

Change these value to reset qsv codec's bitrate control configuration.

pic_timing_sei

Supported in h264_qsv and hevc_qsv. Change this value to reset qsv codec's pic_timing_sei configuration.

H264 options

These options are used by h264_qsv

extbrc

Extended bitrate control.

recovery_point_sei

Set this flag to insert the recovery point SEI message at the beginning of every intra refresh cycle.

rdo Enable rate distortion optimization.

max_frame_size

Maximum encoded frame size in bytes.

max_frame_size_i

Maximum encoded frame size for I frames in bytes. If this value is set as larger than zero, then for I frames the value set by max_frame_size is ignored.

max_frame_size_p

Maximum encoded frame size for P frames in bytes. If this value is set as larger than zero, then for P frames the value set by `max_frame_size` is ignored.

max_slice_size

Maximum encoded slice size in bytes.

bitrate_limit

Toggle bitrate limitations. Modifies bitrate to be in the range imposed by the QSV encoder. Setting this flag off may lead to violation of HRD conformance. Mind that specifying bitrate below the QSV encoder range might significantly affect quality. If on this option takes effect in non CQP modes: if bitrate is not in the range imposed by the QSV encoder, it will be changed to be in the range.

mbbrc

Setting this flag enables macroblock level bitrate control that generally improves subjective visual quality. Enabling this flag may have negative impact on performance and objective visual quality metric.

low_delay_brc

Setting this flag turns on or off LowDelayBRC feature in qsv plugin, which provides more accurate bitrate control to minimize the variance of bitstream size frame by frame. Value: -1-default 0-off 1-on

adaptive_i

This flag controls insertion of I frames by the QSV encoder. Turn ON this flag to allow changing of frame type from P and B to I.

adaptive_b

This flag controls changing of frame type from B to P.

p_strategy

Enable P-pyramid: 0-default 1-simple 2-pyramid(bf need to be set to 0).

b_strategy

This option controls usage of B frames as reference.

dblkc_idc

This option disable deblocking. It has value in range 0~2.

cavlc

If set, CAVLC is used; if unset, CABAC is used for encoding.

vcm Video conferencing mode, please see ratecontrol method.

idr_interval

Distance (in I-frames) between IDR frames.

pic_timing_sei

Insert picture timing SEI with pic_struct_syntax element.

single_sei_nal_unit

Put all the SEI messages into one NALU.

max_dec_frame_buffering

Maximum number of frames buffered in the DPB.

look_ahead

Use VBR algorithm with look ahead.

look_ahead_depth

Depth of look ahead in number frames.

look_ahead_downsampling

Downscaling factor for the frames saved for the lookahead analysis.

unknown

auto

off

2x

4x

int_ref_type

Specifies intra refresh type. The major goal of intra refresh is improvement of error resilience without significant impact on encoded bitstream size caused by I frames. The SDK encoder achieves this by encoding part of each frame in refresh cycle using intra MBs. *none* means no refresh. *vertical* means vertical refresh, by column of MBs. *horizontal* means horizontal refresh, by rows of MBs. *slice* means horizontal refresh by slices without overlapping. In case of *slice*, *in_ref_cycle_size* is ignored. To enable intra refresh, B frame should be set to 0.

int_ref_cycle_size

Specifies number of pictures within refresh cycle starting from 2. 0 and 1 are invalid values.

int_ref_qp_delta

Specifies QP difference for inserted intra MBs. This is signed value in [-51, 51] range if target encoding bit-depth for luma samples is 8 and this range is [-63, 63] for 10 bit-depth or [-75, 75] for 12 bit-depth respectively.

int_ref_cycle_dist

Distance between the beginnings of the intra-refresh cycles in frames.

profile

unknown

baseline

main

high

a53cc

Use A53 Closed Captions (if available).

aud Insert the Access Unit Delimiter NAL.

mfmode

Multi-Frame Mode.

off

auto

repeat_pps

Repeat pps for every frame.

max_qp_i

Maximum video quantizer scale for I frame.

min_qp_i

Minimum video quantizer scale for I frame.

max_qp_p

Maximum video quantizer scale for P frame.

min_qp_p

Minimum video quantizer scale for P frame.

max_qp_b

Maximum video quantizer scale for B frame.

min_qp_b

Minimum video quantizer scale for B frame.

scenario

Provides a hint to encoder about the scenario for the encoding session.

unknown

displayremoting

videoconference

archive

livestreaming

cameracapture

videosurveillance

gamestreaming

remotegaming

avbr_accuracy

Accuracy of the AVBR ratecontrol (unit of tenth of percent).

avbr_convergence

Convergence of the AVBR ratecontrol (unit of 100 frames)

The parameters *avbr_accuracy* and *avbr_convergence* are for the average variable bitrate control (AVBR) algorithm. The algorithm focuses on overall encoding quality while meeting the specified bitrate, *target_bitrate*, within the accuracy range *avbr_accuracy*, after a *avbr_Convergence* period. This method does not follow HRD and the instant bitrate is not capped or padded.

skip_frame

Use per-frame metadata "qsv_skip_frame" to skip frame when encoding. This option defines the usage of this metadata.

no_skip

Frame skipping is disabled.

insert_dummy

Encoder inserts into bitstream frame where all macroblocks are encoded as skipped.

insert_nothing

Similar to *insert_dummy*, but encoder inserts nothing into bitstream. The skipped frames are still used in brc. For example, gop still include skipped frames, and the frames after skipped

frames will be larger in size.

brc_only

skip_frame metadata indicates the number of missed frames before the current frame.

HEVC Options

These options are used by hevc_qsv

extbrc

Extended bitrate control.

recovery_point_sei

Set this flag to insert the recovery point SEI message at the beginning of every intra refresh cycle.

rdo Enable rate distortion optimization.

max_frame_size

Maximum encoded frame size in bytes.

max_frame_size_i

Maximum encoded frame size for I frames in bytes. If this value is set as larger than zero, then for I frames the value set by max_frame_size is ignored.

max_frame_size_p

Maximum encoded frame size for P frames in bytes. If this value is set as larger than zero, then for P frames the value set by max_frame_size is ignored.

max_slice_size

Maximum encoded slice size in bytes.

mbbrc

Setting this flag enables macroblock level bitrate control that generally improves subjective visual quality. Enabling this flag may have negative impact on performance and objective visual quality metric.

low_delay_brc

Setting this flag turns on or off LowDelayBRC feature in qsv plugin, which provides more accurate bitrate control to minimize the variance of bitstream size frame by frame. Value:

-1-default 0-off 1-on

adaptive_i

This flag controls insertion of I frames by the QSV encoder. Turn ON this flag to allow changing of frame type from P and B to I.

adaptive_b

This flag controls changing of frame type from B to P.

p_strategy

Enable P-pyramid: 0-default 1-simple 2-pyramid(bf need to be set to 0).

b_strategy

This option controls usage of B frames as reference.

dblkc_idc

This option disable deblocking. It has value in range 0~2.

idr_interval

Distance (in I-frames) between IDR frames.

begin_only

Output an IDR-frame only at the beginning of the stream.

load_plugin

A user plugin to load in an internal session.

none**hevc_sw****hevc_hw***load_plugins*

A :-separate list of hexadecimal plugin UIDs to load in an internal session.

look_ahead_depth

Depth of look ahead in number frames, available when extbrc option is enabled.

profile

Set the encoding profile (scc requires libmfx >= 1.32).

unknown**main****main10**

mainsp**rext****scc**

tier Set the encoding tier (only level ≥ 4 can support high tier). This option only takes effect when the level option is specified.

main**high**

gpb 1: GPB (generalized P/B frame)

0: regular P frame.

tile_cols

Number of columns for tiled encoding.

tile_rows

Number of rows for tiled encoding.

aud Insert the Access Unit Delimiter NAL.

pic_timing_sei

Insert picture timing SEI with *pic_struct_syntax* element.

transform_skip

Turn this option ON to enable transformskip. It is supported on platform equal or newer than ICL.

int_ref_type

Specifies intra refresh type. The major goal of intra refresh is improvement of error resilience without significant impact on encoded bitstream size caused by I frames. The SDK encoder achieves this by encoding part of each frame in refresh cycle using intra MBs. *none* means no refresh. *vertical* means vertical refresh, by column of MBs. *horizontal* means horizontal refresh, by rows of MBs. *slice* means horizontal refresh by slices without overlapping. In case of *slice*, *in_ref_cycle_size* is ignored. To enable intra refresh, B frame should be set to 0.

int_ref_cycle_size

Specifies number of pictures within refresh cycle starting from 2. 0 and 1 are invalid values.

int_ref_qp_delta

Specifies QP difference for inserted intra MBs. This is signed value in $[-51, 51]$ range if target encoding bit-depth for luma samples is 8 and this range is $[-63, 63]$ for 10 bit-depth or $[-75, 75]$

for 12 bit-depth respectively.

int_ref_cycle_dist

Distance between the beginnings of the intra-refresh cycles in frames.

max_qp_i

Maximum video quantizer scale for I frame.

min_qp_i

Minimum video quantizer scale for I frame.

max_qp_p

Maximum video quantizer scale for P frame.

min_qp_p

Minimum video quantizer scale for P frame.

max_qp_b

Maximum video quantizer scale for B frame.

min_qp_b

Minimum video quantizer scale for B frame.

scenario

Provides a hint to encoder about the scenario for the encoding session.

unknown

displayremoting

videoconference

archive

livestreaming

cameracapture

videosurveillance

gamestreaming

remotegaming

avbr_accuracy

Accuracy of the AVBR ratecontrol (unit of tenth of percent).

avbr_convergence

Convergence of the AVBR ratecontrol (unit of 100 frames)

The parameters *avbr_accuracy* and *avbr_convergence* are for the average variable bitrate control (AVBR) algorithm. The algorithm focuses on overall encoding quality while meeting the specified bitrate, *target_bitrate*, within the accuracy range *avbr_accuracy*, after a *avbr_Convergence* period. This method does not follow HRD and the instant bitrate is not capped or padded.

skip_frame

Use per-frame metadata "qsv_skip_frame" to skip frame when encoding. This option defines the usage of this metadata.

no_skip

Frame skipping is disabled.

insert_dummy

Encoder inserts into bitstream frame where all macroblocks are encoded as skipped.

insert_nothing

Similar to *insert_dummy*, but encoder inserts nothing into bitstream. The skipped frames are still used in brc. For example, gop still include skipped frames, and the frames after skipped frames will be larger in size.

brc_only

skip_frame metadata indicates the number of missed frames before the current frame.

MPEG2 Options

These options are used by *mpeg2_qsv*

profile

unknown
simple
main
high

VP9 Options

These options are used by *vp9_qsv*

profile

unknown

profile0

profile1

profile2

profile3

tile_cols

Number of columns for tiled encoding (requires libmfx >= 1.29).

tile_rows

Number of rows for tiled encoding (requires libmfx >= 1.29).

AV1 Options

These options are used by av1_qsv (requires libvpl).

profile

unknown

main

tile_cols

Number of columns for tiled encoding.

tile_rows

Number of rows for tiled encoding.

adaptive_i

This flag controls insertion of I frames by the QSV encoder. Turn ON this flag to allow changing of frame type from P and B to I.

adaptive_b

This flag controls changing of frame type from B to P.

b_strategy

This option controls usage of B frames as reference.

extbrc

Extended bitrate control.

look_ahead_depth

Depth of look ahead in number frames, available when extbrc option is enabled.

low_delay_brc

Setting this flag turns on or off LowDelayBRC feature in qsv plugin, which provides more accurate bitrate control to minimize the variance of bitstream size frame by frame. Value:
-1-default 0-off 1-on

max_frame_size

Set the allowed max size in bytes for each frame. If the frame size exceeds the limitation, encoder will adjust the QP value to control the frame size. Invalid in CQP rate control mode.

snow

Options

iterative_dia_size

dia size for the iterative motion estimation

VAAPI encoders

Wrappers for hardware encoders accessible via VAAPI.

These encoders only accept input in VAAPI hardware surfaces. If you have input in software frames, use the **hwupload** filter to upload them to the GPU.

The following standard libavcodec options are used:

⊕ **g / gop_size**

⊕ **bf / max_b_frames**

⊕ **profile**

If not set, this will be determined automatically from the format of the input frames and the profiles supported by the driver.

⊕ **level**

⊕ **b / bit_rate**

⊕ **maxrate / rc_max_rate**

⊕ **bufsize / rc_buffer_size**

⊕ **rc_init_occupancy / rc_initial_buffer_occupancy**

⊕ **compression_level**

Speed / quality tradeoff: higher values are faster / worse quality.

⊕ **q / global_quality**

Size / quality tradeoff: higher values are smaller / worse quality.

⊕ **qmin**

⊕ **qmax**

⊕ **i_qfactor / i_quant_factor**

⊕ **i_qoffset / i_quant_offset**

⊕ **b_qfactor / b_quant_factor**

⊕ **b_qoffset / b_quant_offset**

⊕ **slices**

All encoders support the following options:

low_power

Some drivers/platforms offer a second encoder for some codecs intended to use less power than the default encoder; setting this option will attempt to use that encoder. Note that it may support a reduced feature set, so some other options may not be available in this mode.

idr_interval

Set the number of normal intra frames between full-refresh (IDR) frames in open-GOP mode. The intra frames are still IRAPs, but will not include global headers and may have non-decodable leading pictures.

b_depth

Set the B-frame reference depth. When set to one (the default), all B-frames will refer only to P- or I-frames. When set to greater values multiple layers of B-frames will be present, frames in each layer only referring to frames in higher layers.

async_depth

Maximum processing parallelism. Increase this to improve single channel performance. This option doesn't work if driver doesn't implement `vaSyncBuffer` function. Please make sure there are enough `hw_frames` allocated if a large number of `async_depth` is used.

max_frame_size

Set the allowed max size in bytes for each frame. If the frame size exceeds the limitation, encoder will adjust the QP value to control the frame size. Invalid in CQP rate control mode.

rc_mode

Set the rate control mode to use. A given driver may only support a subset of modes.

Possible modes:

auto Choose the mode automatically based on driver support and the other options. This is the default.

CQP

Constant-quality.

CBR

Constant-bitrate.

VBR

Variable-bitrate.

ICQ Intelligent constant-quality.

QVBR

Quality-defined variable-bitrate.

AVBR

Average variable bitrate.

Each encoder also has its own specific options:

h264_vaapi

profile sets the value of *profile_idc* and the *constraint_set*_flags*. **level** sets the value of *level_idc*.

coder

Set entropy encoder (default is *cabac*). Possible values:

ac

cabac

Use CABAC.

vlc

cavlc

Use CAVLC.

aud Include access unit delimiters in the stream (not included by default).

sei Set SEI message types to include. Some combination of the following values:

identifier

Include a *user_data_unregistered* message containing information about the encoder.

timing

Include picture timing parameters (*buffering_period* and *pic_timing* messages).

recovery_point

Include recovery points where appropriate (*recovery_point* messages).

hevc_vaapi

profile and **level** set the values of *general_profile_idc* and *general_level_idc* respectively.

aud Include access unit delimiters in the stream (not included by default).

tier Set *general_tier_flag*. This may affect the level chosen for the stream if it is not explicitly specified.

sei Set SEI message types to include. Some combination of the following values:

hdr Include HDR metadata if the input frames have it (*mastering_display_colour_volume* and *content_light_level* messages).

tiles Set the number of tiles to encode the input video with, as columns x rows. Larger numbers allow greater parallelism in both encoding and decoding, but may decrease coding efficiency.

mjpeg_vaapi

Only baseline DCT encoding is supported. The encoder always uses the standard quantisation and huffman tables - **global_quality** scales the standard quantisation table (range 1-100).

For YUV, 4:2:0, 4:2:2 and 4:4:4 subsampling modes are supported. RGB is also supported, and will create an RGB JPEG.

jfif Include JFIF header in each frame (not included by default).

huffman

Include standard huffman tables (on by default). Turning this off will save a few hundred bytes in each output frame, but may lose compatibility with some JPEG decoders which don't fully handle MJPEG.

mpeg2_vaapi

profile and **level** set the value of *profile_and_level_indication*.

vp8_vaapi

B-frames are not supported.

global_quality sets the *q_idx* used for non-key frames (range 0-127).

loop_filter_level

loop_filter_sharpness

Manually set the loop filter parameters.

vp9_vaapi

global_quality sets the *q_idx* used for P-frames (range 0-255).

loop_filter_level

loop_filter_sharpness

Manually set the loop filter parameters.

B-frames are supported, but the output stream is always in encode order rather than display order. If B-frames are enabled, it may be necessary to use the **vp9_raw_reorder** bitstream filter to modify the output stream to display frames in the correct order.

Only normal frames are produced - the **vp9_superframe** bitstream filter may be required to produce a stream usable with all decoders.

vbn

Vizrt Binary Image encoder.

This format is used by the broadcast vendor Vizrt for quick texture streaming. Advanced features of

the format such as LZW compression of texture data or generation of mipmaps are not supported.

Options

format string

Sets the texture compression used by the VBN file. Can be *dxt1*, *dxt5* or *raw*. Default is *dxt5*.

vc2

SMPTE VC-2 (previously BBC Dirac Pro). This codec was primarily aimed at professional broadcasting but since it supports yuv420, yuv422 and yuv444 at 8 (limited range or full range), 10 or 12 bits, this makes it suitable for other tasks which require low overhead and low compression (like screen recording).

Options

- b** Sets target video bitrate. Usually that's around 1:6 of the uncompressed video bitrate (e.g. for 1920x1080 50fps yuv422p10 that's around 400Mbps). Higher values (close to the uncompressed bitrate) turn on lossless compression mode.

field_order

Enables field coding when set (e.g. to *tt* - top field first) for interlaced inputs. Should increase compression with interlaced content as it splits the fields and encodes each separately.

wavelet_depth

Sets the total amount of wavelet transforms to apply, between 1 and 5 (default). Lower values reduce compression and quality. Less capable decoders may not be able to handle values of **wavelet_depth** over 3.

wavelet_type

Sets the transform type. Currently only *5_3* (LeGall) and *9_7* (Deslauriers-Dubuc) are implemented, with *9_7* being the one with better compression and thus is the default.

slice_width

slice_height

Sets the slice size for each slice. Larger values result in better compression. For compatibility with other more limited decoders use **slice_width** of 32 and **slice_height** of 8.

tolerance

Sets the undershoot tolerance of the rate control system in percent. This is to prevent an expensive search from being run.

qm Sets the quantization matrix preset to use by default or when **wavelet_depth** is set to 5

- *default* Uses the default quantization matrix from the specifications, extended with values for the fifth level. This provides a good balance between keeping detail and omitting artifacts.
- *flat* Use a completely zeroed out quantization matrix. This increases PSNR but might reduce perception. Use in bogus benchmarks.
- *color* Reduces detail but attempts to preserve color at extremely low bitrates.

SUBTITLES ENCODERS

dvds

This codec encodes the bitmap subtitle format that is used in DVDs. Typically they are stored in VOBSUB file pairs (*.idx + *.sub), and they can also be used in Matroska files.

Options

palette

Specify the global palette used by the bitmaps.

The format for this option is a string containing 16 24-bits hexadecimal numbers (without 0x prefix) separated by commas, for example "0d00ee, ee450d, 101010, eaeaea, 0ce60b, ec14ed, ebff0b, 0d617a, 7b7b7b, d1d1d1, 7b2a0e, 0d950c, 0f007b, cf0dec, cfa80c, 7c127b".

even_rows_fix

When set to 1, enable a work-around that makes the number of pixel rows even in all subtitles. This fixes a problem with some players that cut off the bottom row if the number is odd. The work-around just adds a fully transparent row if needed. The overhead is low, typically one byte per subtitle on average.

By default, this work-around is disabled.

BITSTREAM FILTERS

When you configure your FFmpeg build, all the supported bitstream filters are enabled by default. You can list all available ones using the configure option "--list-bsfs".

You can disable all the bitstream filters using the configure option "--disable-bsfs", and selectively enable any bitstream filter using the option "--enable-bsf=BSF", or you can disable a particular bitstream filter using the option "--disable-bsf=BSF".

The option "-bsfs" of the ff* tools will display the list of all the supported bitstream filters included in your build.

The ff* tools have a -bsf option applied per stream, taking a comma-separated list of filters, whose parameters follow the filter name after a '='.

```
ffmpeg -i INPUT -c:v copy -bsf:v filter1[=opt1=str1:opt2=str2][,filter2] OUTPUT
```

Below is a description of the currently available bitstream filters, with their parameters, if any.

aac_adtstoasc

Convert MPEG-2/4 AAC ADTS to an MPEG-4 Audio Specific Configuration bitstream.

This filter creates an MPEG-4 AudioSpecificConfig from an MPEG-2/4 ADTS header and removes the ADTS header.

This filter is required for example when copying an AAC stream from a raw ADTS AAC or an MPEG-TS container to MP4A-LATM, to an FLV file, or to MOV/MP4 files and related formats such as 3GP or M4A. Please note that it is auto-inserted for MP4A-LATM and MOV/MP4 and related formats.

av1_metadata

Modify metadata embedded in an AV1 stream.

td Insert or remove temporal delimiter OBUs in all temporal units of the stream.

insert

Insert a TD at the beginning of every TU which does not already have one.

remove

Remove the TD from the beginning of every TU which has one.

color primaries

transfer characteristics

matrix coefficients

Set the color description fields in the stream (see AV1 section 6.4.2).

color_range

Set the color range in the stream (see AV1 section 6.4.2; note that this cannot be set for streams using BT.709 primaries, sRGB transfer characteristic and identity (RGB) matrix coefficients).

tv Limited range.

pc Full range.

chroma_sample_position

Set the chroma sample location in the stream (see AV1 section 6.4.2). This can only be set for 4:2:0 streams.

vertical

Left position (matching the default in MPEG-2 and H.264).

colocated

Top-left position.

tick_rate

Set the tick rate (*time_scale / num_units_in_display_tick*) in the timing info in the sequence header.

num_ticks_per_picture

Set the number of ticks in each picture, to indicate that the stream has a fixed framerate. Ignored if **tick_rate** is not also set.

delete_padding

Deletes Padding OBUs.

chomp

Remove zero padding at the end of a packet.

dca_core

Extract the core from a DCA/DTS stream, dropping extensions such as DTS-HD.

dump_extra

Add extradata to the beginning of the filtered packets except when said packets already exactly begin with the extradata that is intended to be added.

freq The additional argument specifies which packets should be filtered. It accepts the values:

k

keyframe

add extradata to all key packets

e**all** add extradata to all packets

If not specified it is assumed **k**.

For example the following **ffmpeg** command forces a global header (thus disabling individual packet headers) in the H.264 packets generated by the "libx264" encoder, but corrects them by adding the header stored in extradata to the key packets:

```
ffmpeg -i INPUT -map 0 -flags:v +global_header -c:v libx264 -bsf:v dump_extra out.ts
```

dv_error_marker

Blocks in DV which are marked as damaged are replaced by blocks of the specified color.

color

The color to replace damaged blocks by

sta A 16 bit mask which specifies which of the 16 possible error status values are to be replaced by colored blocks. 0xFFFE is the default which replaces all non 0 error status values.

ok No error, no concealment

err Error, No concealment

res Reserved

notok

Error or concealment

notres

Not reserved

Aa, Ba, Ca, Ab, Bb, Cb, A, B, C, a, b, erri, erru

The specific error status code

see page 44-46 or section 5.5 of

<http://web.archive.org/web/20060927044735/http://www.smpte.org/smpte_store/standards/pdf/s314m.pdf>

eac3_core

Extract the core from a E-AC-3 stream, dropping extra channels.

extract_extradata

Extract the in-band extradata.

Certain codecs allow the long-term headers (e.g. MPEG-2 sequence headers, or H.264/HEVC (VPS/)SPS/PPS) to be transmitted either "in-band" (i.e. as a part of the bitstream containing the coded frames) or "out of band" (e.g. on the container level). This latter form is called "extradata" in FFmpeg terminology.

This bitstream filter detects the in-band headers and makes them available as extradata.

remove

When this option is enabled, the long-term headers are removed from the bitstream after extraction.

filter_units

Remove units with types in or not in a given set from the stream.

pass_types

List of unit types or ranges of unit types to pass through while removing all others. This is specified as a '|' -separated list of unit type values or ranges of values with '-'.

remove_types

Identical to **pass_types**, except the units in the given set removed and all others passed through.

Extradata is unchanged by this transformation, but note that if the stream contains inline parameter sets then the output may be unusable if they are removed.

For example, to remove all non-VCL NAL units from an H.264 stream:

```
ffmpeg -i INPUT -c:v copy -bsf:v 'filter_units=pass_types=1-5' OUTPUT
```

To remove all AUDs, SEI and filler from an H.265 stream:

```
ffmpeg -i INPUT -c:v copy -bsf:v 'filter_units=remove_types=35|38-40' OUTPUT
```

hapqa_extract

Extract Rgb or Alpha part of an HAPQA file, without recompression, in order to create an HAPQ or an HAPAlphaOnly file.

texture

Specifies the texture to keep.

color
alpha

Convert HAPQA to HAPQ

```
ffmpeg -i hapqa_inputfile.mov -c copy -bsf:v hapqa_extract=texture=color -tag:v HapY -metadata:s:v:0 encoder=
```

Convert HAPQA to HAPAlphaOnly

```
ffmpeg -i hapqa_inputfile.mov -c copy -bsf:v hapqa_extract=texture=alpha -tag:v HapA -metadata:s:v:0 encoder=
```

h264_metadata

Modify metadata embedded in an H.264 stream.

aud Insert or remove AUD NAL units in all access units of the stream.

pass
insert
remove

Default is pass.

sample_aspect_ratio

Set the sample aspect ratio of the stream in the VUI parameters. See H.264 table E-1.

overscan_appropriate_flag

Set whether the stream is suitable for display using overscan or not (see H.264 section E.2.1).

video_format

video_full_range_flag

Set the video format in the stream (see H.264 section E.2.1 and table E-2).

colour primaries

transfer characteristics

matrix coefficients

Set the colour description in the stream (see H.264 section E.2.1 and tables E-3, E-4 and E-5).

chroma_sample_loc_type

Set the chroma sample location in the stream (see H.264 section E.2.1 and figure E-1).

tick_rate

Set the tick rate ($\text{time_scale} / \text{num_units_in_tick}$) in the VUI parameters. This is the smallest time unit representable in the stream, and in many cases represents the field rate of the stream (double the frame rate).

fixed_frame_rate_flag

Set whether the stream has fixed framerate - typically this indicates that the framerate is exactly half the tick rate, but the exact meaning is dependent on interlacing and the picture structure (see H.264 section E.2.1 and table E-6).

zero_new_constraint_set_flags

Zero `constraint_set4_flag` and `constraint_set5_flag` in the SPS. These bits were reserved in a previous version of the H.264 spec, and thus some hardware decoders require these to be zero. The result of zeroing this is still a valid bitstream.

crop_left**crop_right****crop_top****crop_bottom**

Set the frame cropping offsets in the SPS. These values will replace the current ones if the stream is already cropped.

These fields are set in pixels. Note that some sizes may not be representable if the chroma is subsampled or the stream is interlaced (see H.264 section 7.4.2.1.1).

sei_user_data

Insert a string as SEI unregistered user data. The argument must be of the form *UUID+string*, where the UUID is as hex digits possibly separated by hyphens, and the string can be anything.

For example, **086f3693-b7b3-4f2c-9653-21492fccc5b8+hello** will insert the string ‘hello’ associated with the given UUID.

delete_filler

Deletes both filler NAL units and filler SEI messages.

display_orientation

Insert, extract or remove Display orientation SEI messages. See H.264 section D.1.27 and D.2.27 for syntax and semantics.

pass
insert
remove
extract

Default is pass.

Insert mode works in conjunction with "rotate" and "flip" options. Any pre-existing Display orientation messages will be removed in insert or remove mode. Extract mode attaches the display matrix to the packet as side data.

rotate

Set rotation in display orientation SEI (anticlockwise angle in degrees). Range is -360 to +360. Default is NaN.

flip Set flip in display orientation SEI.

horizontal
vertical

Default is unset.

level

Set the level in the SPS. Refer to H.264 section A.3 and tables A-1 to A-5.

The argument must be the name of a level (for example, **4.2**), a level_idc value (for example, **42**), or the special name **auto** indicating that the filter should attempt to guess the level from the input stream properties.

h264_mp4toannexb

Convert an H.264 bitstream from length prefixed mode to start code prefixed mode (as defined in the Annex B of the ITU-T H.264 specification).

This is required by some streaming formats, typically the MPEG-2 transport stream format (muxer "mpegts").

For example to remux an MP4 file containing an H.264 stream to mpegts format with **ffmpeg**, you can use the command:

```
ffmpeg -i INPUT.mp4 -codec copy -bsf:v h264_mp4toannexb OUTPUT.ts
```

Please note that this filter is auto-inserted for MPEG-TS (muxer "mpegts") and raw H.264 (muxer "h264") output formats.

h264_redundant_pps

This applies a specific fixup to some Blu-ray streams which contain redundant PPSs modifying irrelevant parameters of the stream which confuse other transformations which require correct extradata.

hevc_metadata

Modify metadata embedded in an HEVC stream.

aud Insert or remove AUD NAL units in all access units of the stream.

insert

remove

sample_aspect_ratio

Set the sample aspect ratio in the stream in the VUI parameters.

video_format

video_full_range_flag

Set the video format in the stream (see H.265 section E.3.1 and table E.2).

colour primaries

transfer characteristics

matrix coefficients

Set the colour description in the stream (see H.265 section E.3.1 and tables E.3, E.4 and E.5).

chroma_sample_loc_type

Set the chroma sample location in the stream (see H.265 section E.3.1 and figure E.1).

tick_rate

Set the tick rate in the VPS and VUI parameters ($\text{time_scale} / \text{num_units_in_tick}$). Combined with **num_ticks_poc_diff_one**, this can set a constant framerate in the stream. Note that it is likely to be overridden by container parameters when the stream is in a container.

num_ticks_poc_diff_one

Set **poc_proportional_to_timing_flag** in VPS and VUI and use this value to set **num_ticks_poc_diff_one_minus1** (see H.265 sections 7.4.3.1 and E.3.1). Ignored if **tick_rate** is not also set.

crop_left**crop_right****crop_top****crop_bottom**

Set the conformance window cropping offsets in the SPS. These values will replace the current ones if the stream is already cropped.

These fields are set in pixels. Note that some sizes may not be representable if the chroma is subsampled (H.265 section 7.4.3.2.1).

level

Set the level in the VPS and SPS. See H.265 section A.4 and tables A.6 and A.7.

The argument must be the name of a level (for example, **5.1**), a *general_level_idc* value (for example, **153** for level 5.1), or the special name **auto** indicating that the filter should attempt to guess the level from the input stream properties.

hevc_mp4toannexb

Convert an HEVC/H.265 bitstream from length prefixed mode to start code prefixed mode (as defined in the Annex B of the ITU-T H.265 specification).

This is required by some streaming formats, typically the MPEG-2 transport stream format (muxer "mpegts").

For example to remux an MP4 file containing an HEVC stream to mpegts format with **ffmpeg**, you can use the command:

```
ffmpeg -i INPUT.mp4 -codec copy -bsf:v hevc_mp4toannexb OUTPUT.ts
```

Please note that this filter is auto-inserted for MPEG-TS (muxer "mpegts") and raw HEVC/H.265 (muxer "h265" or "hevc") output formats.

imxdump

Modifies the bitstream to fit in MOV and to be usable by the Final Cut Pro decoder. This filter only applies to the mpeg2video codec, and is likely not needed for Final Cut Pro 7 and newer with the appropriate **-tag:v**.

For example, to remux 30 MB/sec NTSC IMX to MOV:

```
ffmpeg -i input.mxf -c copy -bsf:v imxdump -tag:v mx3n output.mov
```

mjpeg2jpeg

Convert MJPEG/AVI1 packets to full JPEG/JFIF packets.

MJPEG is a video codec wherein each video frame is essentially a JPEG image. The individual frames can be extracted without loss, e.g. by

```
ffmpeg -i ../some_mjpeg.avi -c:v copy frames_%d.jpg
```

Unfortunately, these chunks are incomplete JPEG images, because they lack the DHT segment required for decoding. Quoting from <http://www.digitalpreservation.gov/formats/fdd/fdd000063.shtml>:

Avery Lee, writing in the rec.video.desktop newsgroup in 2001, commented that "MJPEG, or at least the MJPEG in AVIs having the MJPG fourcc, is restricted JPEG with a fixed -- and *omitted* -- Huffman table. The JPEG must be YCbCr colorspace, it must be 4:2:2, and it must use basic Huffman encoding, not arithmetic or progressive. . . . You can indeed extract the MJPEG frames and decode them with a regular JPEG decoder, but you have to prepend the DHT segment to them, or else the decoder won't have any idea how to decompress the data. The exact table necessary is given in the OpenDML spec."

This bitstream filter patches the header of frames extracted from an MJPEG stream (carrying the AVI1 header ID and lacking a DHT segment) to produce fully qualified JPEG images.

```
ffmpeg -i mjpeg-movie.avi -c:v copy -bsf:v mjpeg2jpeg frame_%d.jpg
exiftran -i -9 frame*.jpg
ffmpeg -i frame_%d.jpg -c:v copy rotated.avi
```

mjpegadump

Add an MJPEG A header to the bitstream, to enable decoding by Quicktime.

mov2textsub

Extract a representable text file from MOV subtitles, stripping the metadata header from each subtitle packet.

See also the **text2movsub** filter.

mp3decomp

Decompress non-standard compressed MP3 audio headers.

mpeg2_metadata

Modify metadata embedded in an MPEG-2 stream.

display_aspect_ratio

Set the display aspect ratio in the stream.

The following fixed values are supported:

4/3

16/9

221/100

Any other value will result in square pixels being signalled instead (see H.262 section 6.3.3 and table 6-3).

frame_rate

Set the frame rate in the stream. This is constructed from a table of known values combined with a small multiplier and divisor - if the supplied value is not exactly representable, the nearest representable value will be used instead (see H.262 section 6.3.3 and table 6-4).

video_format

Set the video format in the stream (see H.262 section 6.3.6 and table 6-6).

colour_primaries**transfer_characteristics****matrix_coefficients**

Set the colour description in the stream (see H.262 section 6.3.6 and tables 6-7, 6-8 and 6-9).

mpeg4_unpack_bframes

Unpack DivX-style packed B-frames.

DivX-style packed B-frames are not valid MPEG-4 and were only a workaround for the broken Video for Windows subsystem. They use more space, can cause minor AV sync issues, require more CPU power to decode (unless the player has some decoded picture queue to compensate the 2,0,2,0 frame per packet style) and cause trouble if copied into a standard container like mp4 or mpeg-ps/ts, because MPEG-4 decoders may not be able to decode them, since they are not valid MPEG-4.

For example to fix an AVI file containing an MPEG-4 stream with DivX-style packed B-frames using **ffmpeg**, you can use the command:

```
ffmpeg -i INPUT.avi -codec copy -bsf:v mpeg4_unpack_bframes OUTPUT.avi
```

noise

Damages the contents of packets or simply drops them without damaging the container. Can be used for fuzzing or testing error resilience/concealment.

Parameters:

amount

Accepts an expression whose evaluation per-packet determines how often bytes in that packet will be modified. A value below 0 will result in a variable frequency. Default is 0 which results in no modification. However, if neither amount nor drop is specified, amount will be set to *-1*. See below for accepted variables.

drop

Accepts an expression evaluated per-packet whose value determines whether that packet is dropped. Evaluation to a positive value results in the packet being dropped. Evaluation to a negative value results in a variable chance of it being dropped, roughly inverse in proportion to the magnitude of the value. Default is 0 which results in no drops. See below for accepted variables.

dropamount

Accepts a non-negative integer, which assigns a variable chance of it being dropped, roughly inverse in proportion to the value. Default is 0 which results in no drops. This option is kept for backwards compatibility and is equivalent to setting drop to a negative value with the same magnitude i.e. "dropamount=4" is the same as "drop=-4". Ignored if drop is also specified.

Both "amount" and "drop" accept expressions containing the following variables:

n The index of the packet, starting from zero.

tb The timebase for packet timestamps.

pts Packet presentation timestamp.

dts Packet decoding timestamp.

nopts

Constant representing AV_NOPTS_VALUE.

startpts

First non-AV_NOPTS_VALUE PTS seen in the stream.

startdts

First non-AV_NOPTS_VALUE DTS seen in the stream.

duration

d Packet duration, in timebase units.

pos Packet position in input; may be -1 when unknown or not set.

size Packet size, in bytes.

key Whether packet is marked as a keyframe.

state

A pseudo random integer, primarily derived from the content of packet payload.

Examples

Apply modification to every byte but don't drop any packets.

```
ffmpeg -i INPUT -c copy -bsf noise=1 output.mkv
```

Drop every video packet not marked as a keyframe after timestamp 30s but do not modify any of the remaining packets.

```
ffmpeg -i INPUT -c copy -bsf:v noise=drop='gt(t,30)*not(key)' output.mkv
```

Drop one second of audio every 10 seconds and add some random noise to the rest.

```
ffmpeg -i INPUT -c copy -bsf:a noise=amount=-1:drop='between(mod(t,10)\,9\,10)' output.mkv
```

null

This bitstream filter passes the packets through unchanged.

pcm_rechunk

Repacketize PCM audio to a fixed number of samples per packet or a fixed packet rate per second. This is similar to the **asetnsamples audio filter** but works on audio packets instead of audio frames.

nb_out_samples, n

Set the number of samples per each output audio packet. The number is intended as the number of samples *per each channel*. Default value is 1024.

pad, p

If set to 1, the filter will pad the last audio packet with silence, so that it will contain the same number of samples (or roughly the same number of samples, see **frame_rate**) as the previous ones. Default value is 1.

frame_rate, r

This option makes the filter output a fixed number of packets per second instead of a fixed number of samples per packet. If the audio sample rate is not divisible by the frame rate then the number of samples will not be constant but will vary slightly so that each packet will start as close to the frame boundary as possible. Using this option has precedence over **nb_out_samples**.

You can generate the well known 1602-1601-1602-1601-1602 pattern of 48kHz audio for NTSC frame rate using the **frame_rate** option.

```
ffmpeg -f lavfi -i sine=r=48000:d=1 -c pcm_s16le -bsf pcm_rechunk=r=30000/1001 -f framecrc -
```

pgs_frame_merge

Merge a sequence of PGS Subtitle segments ending with an "end of display set" segment into a single packet.

This is required by some containers that support PGS subtitles (muxer "matroska").

prores_metadata

Modify color property metadata embedded in prores stream.

color primaries

Set the color primaries. Available values are:

auto Keep the same color primaries property (default).

unknown

bt709

bt470bg

BT601 625

smpte170m

BT601 525

bt2020

smpte431

DCI P3

smpte432

P3 D65

transfer_characteristics

Set the color transfer. Available values are:

auto Keep the same transfer characteristics property (default).

unknown

bt709

BT 601, BT 709, BT 2020

smpte2084

SMPTE ST 2084

arib-std-b67

ARIB STD-B67

matrix_coefficients

Set the matrix coefficient. Available values are:

auto Keep the same colorspace property (default).

unknown

bt709

smpte170m

BT 601

bt2020nc

Set Rec709 colorspace for each frame of the file

```
ffmpeg -i INPUT -c copy -bsf:v prores_metadata=color primaries=bt709:color_trc=bt709:colorspace=bt709 output
```

Set Hybrid Log-Gamma parameters for each frame of the file

```
ffmpeg -i INPUT -c copy -bsf:v prores_metadata=color primaries=bt2020:color_trc=arib-std-b67:colorspace=bt2020nc output
```

remove_extra

Remove extradata from packets.

It accepts the following parameter:

freq Set which frame types to remove extradata from.

k Remove extradata from non-keyframes only.

keyframe

Remove extradata from keyframes only.

e, all

Remove extradata from all frames.

setts

Set PTS and DTS in packets.

It accepts the following parameters:

ts**pts**

dts Set expressions for PTS, DTS or both.

duration

Set expression for duration.

time_base

Set output time base.

The expressions are evaluated through the eval API and can contain the following constants:

N The count of the input packet. Starting from 0.

TS The demux timestamp in input in case of "ts" or "dts" option or presentation timestamp in case of "pts" option.

POS

The original position in the file of the packet, or undefined if undefined for the current packet

DTS

The demux timestamp in input.

PTS The presentation timestamp in input.

DURATION

The duration in input.

STARTDTS

The DTS of the first packet.

STARTPTS

The PTS of the first packet.

PREV_INDTS

The previous input DTS.

PREV_INPTS

The previous input PTS.

PREV_INDURATION

The previous input duration.

PREV_OUTDTS

The previous output DTS.

PREV_OUTPTS

The previous output PTS.

PREV_OUTDURATION

The previous output duration.

NEXT_DTS

The next input DTS.

NEXTPTS

The next input PTS.

NEXT_DURATION

The next input duration.

TB The timebase of stream packet belongs.

TB_OUT

The output timebase.

SR The sample rate of stream packet belongs.

NOPTS

The AV_NOPTS_VALUE constant.

text2movsub

Convert text subtitles to MOV subtitles (as used by the "mov_text" codec) with metadata headers.

See also the **mov2textsub** filter.

trace_headers

Log trace output containing all syntax elements in the coded stream headers (everything above the level of individual coded blocks). This can be useful for debugging low-level stream issues.

Supports AV1, H.264, H.265, (M)JPEG, MPEG-2 and VP9, but depending on the build only a subset of these may be available.

truehd_core

Extract the core from a TrueHD stream, dropping ATMOS data.

vp9_metadata

Modify metadata embedded in a VP9 stream.

color_space

Set the color space value in the frame header. Note that any frame set to RGB will be implicitly set to PC range and that RGB is incompatible with profiles 0 and 2.

unknown

bt601

bt709

smpte170

smpte240

bt2020

rgb

color_range

Set the color range value in the frame header. Note that any value imposed by the color space will take precedence over this value.

tv

pc

vp9_superframe

Merge VP9 invisible (alt-ref) frames back into VP9 superframes. This fixes merging of split/segmented VP9 streams where the alt-ref frame was split from its visible counterpart.

vp9_superframe_split

Split VP9 superframes into single frames.

vp9_raw_reorder

Given a VP9 stream with correct timestamps but possibly out of order, insert additional show-existing-frame packets to correct the ordering.

FORMAT OPTIONS

The libavformat library provides some generic global options, which can be set on all the muxers and demuxers. In addition each muxer or demuxer may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the "AVFormatContext" options or using the *libavutil/opt.h* API for programmatic use.

The list of supported options follows:

avioflags *flags (input/output)*

Possible values:

direct

Reduce buffering.

probesize *integer (input)*

Set probing size in bytes, i.e. the size of the data to analyze to get stream information. A higher value will enable detecting more information in case it is dispersed into the stream, but will increase latency. Must be an integer not lesser than 32. It is 5000000 by default.

max_probe_packets *integer (input)*

Set the maximum number of buffered packets when probing a codec. Default is 2500 packets.

packetsize *integer (output)*

Set packet size.

fflags *flags*

Set format flags. Some are implemented for a limited number of formats.

Possible values for input files:

discardcorrupt

Discard corrupted packets.

fastseek

Enable fast, but inaccurate seeks for some formats.

genpts

Generate missing PTS if DTS is present.

igndts

Ignore DTS if PTS is set. Inert when nofillin is set.

ignidx

Ignore index.

nobuffer

Reduce the latency introduced by buffering during initial input streams analysis.

nofillin

Do not fill in missing values in packet fields that can be exactly calculated.

noparse

Disable AVParsers, this needs "+nofillin" too.

sortdts

Try to interleave output packets by DTS. At present, available only for AVIs with an index.

Possible values for output files:

autobsf

Automatically apply bitstream filters as required by the output format. Enabled by default.

bitexact

Only write platform-, build- and time-independent data. This ensures that file and data checksums are reproducible and match between platforms. Its primary use is for regression testing.

flush_packets

Write out packets immediately.

shortest

Stop muxing at the end of the shortest stream. It may be needed to increase `max_interleave_delta` to avoid flushing the longer streams before EOF.

seek2any *integer (input)*

Allow seeking to non-keyframes on demuxer level when supported if set to 1. Default is 0.

analyzeduration *integer (input)*

Specify how many microseconds are analyzed to probe the input. A higher value will enable detecting more accurate information, but will increase latency. It defaults to 5,000,000 microseconds = 5 seconds.

cryptokey *hexadecimal string (input)*

Set decryption key.

indexmem *integer (input)*

Set max memory used for timestamp index (per stream).

rtbufsize *integer (input)*

Set max memory used for buffering real-time frames.

fdebug *flags (input/output)*

Print specific debug info.

Possible values:

ts**max_delay** *integer (input/output)*

Set maximum muxing or demuxing delay in microseconds.

fpsprobesize *integer (input)*

Set number of frames used to probe fps.

audio_preload *integer (output)*

Set microseconds by which audio packets should be interleaved earlier.

chunk_duration *integer (output)*

Set microseconds for each chunk.

chunk_size *integer (output)*

Set size in bytes for each chunk.

err_detect, f_err_detect *flags (input)*

Set error detection flags. "f_err_detect" is deprecated and should be used only via the **ffmpeg** tool.

Possible values:

crccheck

Verify embedded CRCs.

bitstream

Detect bitstream specification deviations.

buffer

Detect improper bitstream length.

explode

Abort decoding on minor error detection.

careful

Consider things that violate the spec and have not been seen in the wild as errors.

compliant

Consider all spec non compliancies as errors.

aggressive

Consider things that a sane encoder should not do as an error.

max_interleave_delta *integer (output)*

Set maximum buffering duration for interleaving. The duration is expressed in microseconds, and defaults to 10000000 (10 seconds).

To ensure all the streams are interleaved correctly, libavformat will wait until it has at least one

packet for each stream before actually writing any packets to the output file. When some streams are "sparse" (i.e. there are large gaps between successive packets), this can result in excessive buffering.

This field specifies the maximum difference between the timestamps of the first and the last packet in the muxing queue, above which libavformat will output a packet regardless of whether it has queued a packet for all the streams.

If set to 0, libavformat will continue buffering packets until it has a packet for each stream, regardless of the maximum timestamp difference between the buffered packets.

use_wallclock_as_timestamps *integer (input)*

Use wallclock as timestamps if set to 1. Default is 0.

avoid_negative_ts *integer (output)*

Possible values:

make_non_negative

Shift timestamps to make them non-negative. Also note that this affects only leading negative timestamps, and not non-monotonic negative timestamps.

make_zero

Shift timestamps so that the first timestamp is 0.

auto (default)

Enables shifting when required by the target format.

disabled

Disables shifting of timestamp.

When shifting is enabled, all output timestamps are shifted by the same amount. Audio, video, and subtitles desynching and relative timestamp differences are preserved compared to how they would have been without shifting.

skip_initial_bytes *integer (input)*

Set number of bytes to skip before reading header and frames if set to 1. Default is 0.

correct_ts_overflow *integer (input)*

Correct single timestamp overflows if set to 1. Default is 1.

flush_packets *integer (output)*

Flush the underlying I/O stream after each packet. Default is -1 (auto), which means that the underlying protocol will decide, 1 enables it, and has the effect of reducing the latency, 0 disables it and may increase IO throughput in some cases.

output_ts_offset *offset (output)*

Set the output time offset.

offset must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1) manual**.

The offset is added by the muxer to the output timestamps.

Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in *offset*. Default value is 0 (meaning that no offset is applied).

format_whitelist *list (input)*

"," separated list of allowed demuxers. By default all are allowed.

dump_separator *string (input)*

Separator used to separate the fields printed on the command line about the Stream parameters.

For example, to separate the fields with newlines and indentation:

```
ffprobe -dump_separator "  
    " -i ~/videos/matrixbench_mpeg2.mpg
```

max_streams *integer (input)*

Specifies the maximum number of streams. This can be used to reject files that would require too many resources due to a large number of streams.

skip_estimate_duration_from_pts *bool (input)*

Skip estimation of input duration when calculated using PTS. At present, applicable for MPEG-PS and MPEG-TS.

strict, f_strict *integer (input/output)*

Specify how strictly to follow the standards. "f_strict" is deprecated and should be used only via the **ffmpeg** tool.

Possible values:

very

strictly conform to an older more strict version of the spec or reference software

strict

strictly conform to all the things in the spec no matter what consequences

normal**unofficial**

allow unofficial extensions

experimental

allow non standardized experimental things, experimental (unfinished/work in progress/not well tested) decoders and encoders. Note: experimental decoders can pose a security risk, do not use this for decoding untrusted input.

Format stream specifiers

Format stream specifiers allow selection of one or more streams that match specific properties.

The exact semantics of stream specifiers is defined by the "avformat_match_stream_specifier()" function declared in the *libavformat/avformat.h* header and documented in the **Stream specifiers section in the ffmpeg(1) manual**.

DEMUXERS

Demuxers are configured elements in FFmpeg that can read the multimedia streams from a particular type of file.

When you configure your FFmpeg build, all the supported demuxers are enabled by default. You can list all available ones using the configure option "--list-demuxers".

You can disable all the demuxers using the configure option "--disable-demuxers", and selectively enable a single demuxer with the option "--enable-demuxer=*DEMUXER*", or disable it with the option "--disable-demuxer=*DEMUXER*".

The option "-demuxers" of the ff* tools will display the list of enabled demuxers. Use "-formats" to view a combined list of enabled demuxers and muxers.

The description of some of the currently available demuxers follows.

aa

Audible Format 2, 3, and 4 demuxer.

This demuxer is used to demux Audible Format 2, 3, and 4 (.aa) files.

aac

Raw Audio Data Transport Stream AAC demuxer.

This demuxer is used to demux an ADTS input containing a single AAC stream alongwith any ID3v1/2 or APE tags in it.

apng

Animated Portable Network Graphics demuxer.

This demuxer is used to demux APNG files. All headers, but the PNG signature, up to (but not including) the first fcTL chunk are transmitted as extradata. Frames are then split as being all the chunks between two fcTL ones, or between the last fcTL and IEND chunks.

-ignore_loop *bool*

Ignore the loop variable in the file if set. Default is enabled.

-max_fps *int*

Maximum framerate in frames per second. Default of 0 imposes no limit.

-default_fps *int*

Default framerate in frames per second when none is specified in the file (0 meaning as fast as possible). Default is 15.

asf

Advanced Systems Format demuxer.

This demuxer is used to demux ASF files and MMS network streams.

-no_resync_search *bool*

Do not try to resynchronize by looking for a certain optional start code.

concat

Virtual concatenation script demuxer.

This demuxer reads a list of files and other directives from a text file and demuxes them one after the other, as if all their packets had been muxed together.

The timestamps in the files are adjusted so that the first file starts at 0 and each next file starts where

the previous one finishes. Note that it is done globally and may cause gaps if all streams do not have exactly the same length.

All files must have the same streams (same codecs, same time base, etc.).

The duration of each file is used to adjust the timestamps of the next file: if the duration is incorrect (because it was computed using the bit-rate or because the file is truncated, for example), it can cause artifacts. The "duration" directive can be used to override the duration stored in each file.

Syntax

The script is a text file in extended-ASCII, with one directive per line. Empty lines, leading spaces and lines starting with '#' are ignored. The following directive is recognized:

"file path"

Path to a file to read; special characters and spaces must be escaped with backslash or single quotes.

All subsequent file-related directives apply to that file.

"ffconcat version 1.0"

Identify the script type and version.

To make FFmpeg recognize the format automatically, this directive must appear exactly as is (no extra space or byte-order-mark) on the very first line of the script.

"duration dur"

Duration of the file. This information can be specified from the file; specifying it here may be more efficient or help if the information from the file is not available or accurate.

If the duration is set for all files, then it is possible to seek in the whole concatenated video.

"inpoint timestamp"

In point of the file. When the demuxer opens the file it instantly seeks to the specified timestamp. Seeking is done so that all streams can be presented successfully at In point.

This directive works best with intra frame codecs, because for non-intra frame ones you will usually get extra packets before the actual In point and the decoded content will most likely contain frames before In point too.

For each file, packets before the file In point will have timestamps less than the calculated start timestamp of the file (negative in case of the first file), and the duration of the files (if not specified by the "duration" directive) will be reduced based on their specified In point.

Because of potential packets before the specified In point, packet timestamps may overlap between two concatenated files.

"outpoint timestamp"

Out point of the file. When the demuxer reaches the specified decoding timestamp in any of the streams, it handles it as an end of file condition and skips the current and all the remaining packets from all streams.

Out point is exclusive, which means that the demuxer will not output packets with a decoding timestamp greater or equal to Out point.

This directive works best with intra frame codecs and formats where all streams are tightly interleaved. For non-intra frame codecs you will usually get additional packets with presentation timestamp after Out point therefore the decoded content will most likely contain frames after Out point too. If your streams are not tightly interleaved you may not get all the packets from all streams before Out point and you may only will be able to decode the earliest stream until Out point.

The duration of the files (if not specified by the "duration" directive) will be reduced based on their specified Out point.

"file_packet_metadata key=value"

Metadata of the packets of the file. The specified metadata will be set for each file packet. You can specify this directive multiple times to add multiple metadata entries. This directive is deprecated, use "file_packet_meta" instead.

"file_packet_meta key value"

Metadata of the packets of the file. The specified metadata will be set for each file packet. You can specify this directive multiple times to add multiple metadata entries.

"option key value"

Option to access, open and probe the file. Can be present multiple times.

"stream"

Introduce a stream in the virtual file. All subsequent stream-related directives apply to the last introduced stream. Some streams properties must be set in order to allow identifying the matching

streams in the subfiles. If no streams are defined in the script, the streams from the first file are copied.

"exact_stream_id id"

Set the id of the stream. If this directive is given, the string with the corresponding id in the subfiles will be used. This is especially useful for MPEG-PS (VOB) files, where the order of the streams is not reliable.

"stream_meta key value"

Metadata for the stream. Can be present multiple times.

"stream_codec value"

Codec for the stream.

"stream_extradata hex_string"

Extradata for the string, encoded in hexadecimal.

"chapter id start end"

Add a chapter. *id* is an unique identifier, possibly small and consecutive.

Options

This demuxer accepts the following option:

safe If set to 1, reject unsafe file paths and directives. A file path is considered safe if it does not contain a protocol specification and is relative and all components only contain characters from the portable character set (letters, digits, period, underscore and hyphen) and have no period at the beginning of a component.

If set to 0, any file name is accepted.

The default is 1.

auto_convert

If set to 1, try to perform automatic conversions on packet data to make the streams concatenable. The default is 1.

Currently, the only conversion is adding the h264_mp4toannexb bitstream filter to H.264 streams in MP4 format. This is necessary in particular if there are resolution changes.

segment_time_metadata

If set to 1, every packet will contain the *lavf.concat.start_time* and the *lavf.concat.duration* packet metadata values which are the *start_time* and the *duration* of the respective file segments in the concatenated output expressed in microseconds. The duration metadata is only set if it is known based on the concat file. The default is 0.

Examples

- ⊕ Use absolute filenames and include some comments:

```
# my first filename
file /mnt/share/file-1.wav
# my second filename including whitespace
file '/mnt/share/file 2.wav'
# my third filename including whitespace plus single quote
file '/mnt/share/file 3\'\''.wav'
```

- ⊕ Allow for input format auto-probing, use safe filenames and set the duration of the first file:

```
ffconcat version 1.0

file file-1.wav
duration 20.0

file subdir/file-2.wav
```

dash

Dynamic Adaptive Streaming over HTTP demuxer.

This demuxer presents all AVStreams found in the manifest. By setting the discard flags on AVStreams the caller can decide which streams to actually receive. Each stream mirrors the "id" and "bandwidth" properties from the "<Representation>" as metadata keys named "id" and "variant_bitrate" respectively.

Options

This demuxer accepts the following option:

cenc_decryption_key

16-byte key, in hex, to decrypt files encrypted using ISO Common Encryption (CENC/AES-128

CTR; ISO/IEC 23001-7).

ea

Electronic Arts Multimedia format demuxer.

This format is used by various Electronic Arts games.

*Options***merge_alpha** *bool*

Normally the VP6 alpha channel (if exists) is returned as a secondary video stream, by setting this option you can make the demuxer return a single video stream which contains the alpha channel in addition to the ordinary video.

imf

Interoperable Master Format demuxer.

This demuxer presents audio and video streams found in an IMF Composition.

flv, live_flv, kux

Adobe Flash Video Format demuxer.

This demuxer is used to demux FLV files and RTMP network streams. In case of live network streams, if you force format, you may use `live_flv` option instead of `flv` to survive timestamp discontinuities. KUX is a flv variant used on the Youku platform.

```
ffmpeg -f flv -i myfile.flv ...
```

```
ffmpeg -f live_flv -i rtmp://<any.server>/anything/key ....
```

-flv_metadata *bool*

Allocate the streams according to the onMetaData array content.

-flv_ignore_prevtag *bool*

Ignore the size of previous tag value.

-flv_full_metadata *bool*

Output all context of the onMetadata.

gif

Animated GIF demuxer.

It accepts the following options:

min_delay

Set the minimum valid delay between frames in hundredths of seconds. Range is 0 to 6000.
Default value is 2.

max_gif_delay

Set the maximum valid delay between frames in hundredth of seconds. Range is 0 to 65535.
Default value is 65535 (nearly eleven minutes), the maximum value allowed by the specification.

default_delay

Set the default delay between frames in hundredths of seconds. Range is 0 to 6000. Default value is 10.

ignore_loop

GIF files can contain information to loop a certain number of times (or infinitely). If **ignore_loop** is set to 1, then the loop setting from the input will be ignored and looping will not occur. If set to 0, then looping will occur and will cycle the number of times according to the GIF. Default value is 1.

For example, with the overlay filter, place an infinitely looping GIF over another video:

```
ffmpeg -i input.mp4 -ignore_loop 0 -i input.gif -filter_complex overlay=shortest=1 out.mkv
```

Note that in the above example the shortest option for overlay filter is used to end the output video at the length of the shortest input file, which in this case is *input.mp4* as the GIF in this example loops infinitely.

hls

HLS demuxer

Apple HTTP Live Streaming demuxer.

This demuxer presents all AVStreams from all variant streams. The id field is set to the bitrate variant index number. By setting the discard flags on AVStreams (by pressing 'a' or 'v' in ffplay), the caller can decide which variant streams to actually receive. The total bitrate of the variant that the stream belongs to is available in a metadata key named "variant_bitrate".

It accepts the following options:

live_start_index

segment index to start live streams at (negative values are from the end).

prefer_x_start

prefer to use #EXT-X-START if it's in playlist instead of live_start_index.

allowed_extensions

',' separated list of file extensions that hls is allowed to access.

max_reload

Maximum number of times a insufficient list is attempted to be reloaded. Default value is 1000.

m3u8_hold_counters

The maximum number of times to load m3u8 when it refreshes without new segments. Default value is 1000.

http_persistent

Use persistent HTTP connections. Applicable only for HTTP streams. Enabled by default.

http_multiple

Use multiple HTTP connections for downloading HTTP segments. Enabled by default for HTTP/1.1 servers.

http_seekable

Use HTTP partial requests for downloading HTTP segments. 0 = disable, 1 = enable, -1 = auto, Default is auto.

seg_format_options

Set options for the demuxer of media segments using a list of key=value pairs separated by ":".

seg_max_retry

Maximum number of times to reload a segment on error, useful when segment skip on network error is not desired. Default value is 0.

image2

Image file demuxer.

This demuxer reads from a list of image files specified by a pattern. The syntax and meaning of the pattern is specified by the option *pattern_type*.

The pattern may contain a suffix which is used to automatically determine the format of the images contained in the files.

The size, the pixel format, and the format of each image must be the same for all the files in the sequence.

This demuxer accepts the following options:

framerate

Set the frame rate for the video stream. It defaults to 25.

loop

If set to 1, loop over the input. Default value is 0.

pattern_type

Select the pattern type used to interpret the provided filename.

pattern_type accepts one of the following values.

none

Disable pattern matching, therefore the video will only contain the specified image. You should use this option if you do not want to create sequences from multiple images and your filenames may contain special pattern characters.

sequence

Select a sequence pattern type, used to specify a sequence of files indexed by sequential numbers.

A sequence pattern may contain the string "%d" or "%0Nd", which specifies the position of the characters representing a sequential number in each filename matched by the pattern. If the form "%d0Nd" is used, the string representing the number in each filename is 0-padded and *N* is the total number of 0-padded digits representing the number. The literal character '%' can be specified in the pattern with the string "%%".

If the sequence pattern contains "%d" or "%0Nd", the first filename of the file list specified by the pattern must contain a number inclusively contained between *start_number* and *start_number+start_number_range-1*, and all the following numbers must be sequential.

For example the pattern "img-%03d.bmp" will match a sequence of filenames of the form *img-001.bmp*, *img-002.bmp*, ..., *img-010.bmp*, etc.; the pattern "i%%m%%g-%d.jpg" will

match a sequence of filenames of the form *i%m%g-1.jpg, i%m%g-2.jpg, ..., i%m%g-10.jpg*, etc.

Note that the pattern must not necessarily contain "%d" or "%0Nd", for example to convert a single image file *img.jpeg* you can employ the command:

```
ffmpeg -i img.jpeg img.png
```

glob

Select a glob wildcard pattern type.

The pattern is interpreted like a "glob()" pattern. This is only selectable if libavformat was compiled with globbing support.

glob_sequence (*deprecated, will be removed*)

Select a mixed glob wildcard/sequence pattern.

If your version of libavformat was compiled with globbing support, and the provided pattern contains at least one glob meta character among "%*?[]{}" that is preceded by an unescaped "%", the pattern is interpreted like a "glob()" pattern, otherwise it is interpreted like a sequence pattern.

All glob special characters "%*?[]{}" must be prefixed with "%". To escape a literal "%" you shall use "%%".

For example the pattern "foo-%*.jpeg" will match all the filenames prefixed by "foo-" and terminating with ".jpeg", and "foo-%????.jpeg" will match all the filenames prefixed with "foo-", followed by a sequence of three characters, and terminating with ".jpeg".

This pattern type is deprecated in favor of *glob* and *sequence*.

Default value is *glob_sequence*.

pixel_format

Set the pixel format of the images to read. If not specified the pixel format is guessed from the first image file in the sequence.

start_number

Set the index of the file matched by the image file pattern to start to read from. Default value is 0.

start_number_range

Set the index interval range to check when looking for the first image file in the sequence, starting from *start_number*. Default value is 5.

ts_from_file

If set to 1, will set frame timestamp to modification time of image file. Note that monotony of timestamps is not provided: images go in the same order as without this option. Default value is 0. If set to 2, will set frame timestamp to the modification time of the image file in nanosecond precision.

video_size

Set the video size of the images to read. If not specified the video size is guessed from the first image file in the sequence.

export_path_metadata

If set to 1, will add two extra fields to the metadata found in input, making them also available for other filters (see *drawtext* filter for examples). Default value is 0. The extra fields are described below:

lavf.image2dec.source_path

Corresponds to the full path to the input file being read.

lavf.image2dec.source_basename

Corresponds to the name of the file being read.

Examples

- ⊕ Use **ffmpeg** for creating a video from the images in the file sequence *img-001.jpeg*, *img-002.jpeg*, ..., assuming an input frame rate of 10 frames per second:

```
ffmpeg -framerate 10 -i 'img-%03d.jpeg' out.mkv
```

- ⊕ As above, but start by reading from a file with index 100 in the sequence:

```
ffmpeg -framerate 10 -start_number 100 -i 'img-%03d.jpeg' out.mkv
```

- ⊕ Read images matching the `"*.png"` glob pattern, that is all the files terminating with the `".png"` suffix:

```
ffmpeg -framerate 10 -pattern_type glob -i "*.png" out.mkv
```

libgme

The Game Music Emu library is a collection of video game music file emulators.

See <<https://bitbucket.org/mpyne/game-music-emu/overview>> for more information.

It accepts the following options:

track_index

Set the index of which track to demux. The demuxer can only export one track. Track indexes start at 0. Default is to pick the first track. Number of tracks is exported as *tracks* metadata entry.

sample_rate

Set the sampling rate of the exported track. Range is 1000 to 999999. Default is 44100.

max_size (*bytes*)

The demuxer buffers the entire file into memory. Adjust this value to set the maximum buffer size, which in turn, acts as a ceiling for the size of files that can be read. Default is 50 MiB.

libmodplug

ModPlug based module demuxer

See <<https://github.com/Konstanty/libmodplug>>

It will export one 2-channel 16-bit 44.1 kHz audio stream. Optionally, a "pal8" 16-color video stream can be exported with or without printed metadata.

It accepts the following options:

noise_reduction

Apply a simple low-pass filter. Can be 1 (on) or 0 (off). Default is 0.

reverb_depth

Set amount of reverb. Range 0-100. Default is 0.

reverb_delay

Set delay in ms, clamped to 40-250 ms. Default is 0.

bass_amount

Apply bass expansion a.k.a. XBass or megabass. Range is 0 (quiet) to 100 (loud). Default is 0.

bass_range

Set cutoff i.e. upper-bound for bass frequencies. Range is 10-100 Hz. Default is 0.

surround_depth

Apply a Dolby Pro-Logic surround effect. Range is 0 (quiet) to 100 (heavy). Default is 0.

surround_delay

Set surround delay in ms, clamped to 5-40 ms. Default is 0.

max_size

The demuxer buffers the entire file into memory. Adjust this value to set the maximum buffer size, which in turn, acts as a ceiling for the size of files that can be read. Range is 0 to 100 MiB. 0 removes buffer size limit (not recommended). Default is 5 MiB.

video_stream_expr

String which is evaluated using the eval API to assign colors to the generated video stream. Variables which can be used are "x", "y", "w", "h", "t", "speed", "tempo", "order", "pattern" and "row".

video_stream

Generate video stream. Can be 1 (on) or 0 (off). Default is 0.

video_stream_w

Set video frame width in 'chars' where one char indicates 8 pixels. Range is 20-512. Default is 30.

video_stream_h

Set video frame height in 'chars' where one char indicates 8 pixels. Range is 20-512. Default is 30.

video_stream_ptxt

Print metadata on video stream. Includes "speed", "tempo", "order", "pattern", "row" and "ts" (time in ms). Can be 1 (on) or 0 (off). Default is 1.

libopenmpt

libopenmpt based module demuxer

See <<https://lib.openmpt.org/libopenmpt/>> for more information.

Some files have multiple subsongs (tracks) this can be set with the **subsong** option.

It accepts the following options:

subsong

Set the subsong index. This can be either 'all', 'auto', or the index of the subsong. Subsong indexes start at 0. The default is 'auto'.

The default value is to let libopenmpt choose.

layout

Set the channel layout. Valid values are 1, 2, and 4 channel layouts. The default value is STEREO.

sample_rate

Set the sample rate for libopenmpt to output. Range is from 1000 to INT_MAX. The value default is 48000.

mov/mp4/3gp

Demuxer for Quicktime File Format & ISO/IEC Base Media File Format (ISO/IEC 14496-12 or MPEG-4 Part 12, ISO/IEC 15444-12 or JPEG 2000 Part 12).

Registered extensions: mov, mp4, m4a, 3gp, 3g2, mj2, psp, m4b, ism, ismv, isma, f4v

Options

This demuxer accepts the following options:

enable_drefs

Enable loading of external tracks, disabled by default. Enabling this can theoretically leak information in some use cases.

use_absolute_path

Allows loading of external tracks via absolute paths, disabled by default. Enabling this poses a security risk. It should only be enabled if the source is known to be non-malicious.

seek_streams_individually

When seeking, identify the closest point in each stream individually and demux packets in that stream from identified point. This can lead to a different sequence of packets compared to demuxing linearly from the beginning. Default is true.

ignore_editlist

Ignore any edit list atoms. The demuxer, by default, modifies the stream index to reflect the timeline described by the edit list. Default is false.

advanced_editlist

Modify the stream index to reflect the timeline described by the edit list. "ignore_editlist" must be set to false for this option to be effective. If both "ignore_editlist" and this option are set to false, then only the start of the stream index is modified to reflect initial dwell time or starting timestamp described by the edit list. Default is true.

ignore_chapters

Don't parse chapters. This includes GoPro 'HiLight' tags/moments. Note that chapters are only parsed when input is seekable. Default is false.

use_mfra_for

For seekable fragmented input, set fragment's starting timestamp from media fragment random access box, if present.

Following options are available:

auto Auto-detect whether to set mfra timestamps as PTS or DTS (*default*)

dts Set mfra timestamps as DTS

pts Set mfra timestamps as PTS

0 Don't use mfra box to set timestamps

use_tfdt

For fragmented input, set fragment's starting timestamp to "baseMediaDecodeTime" from the "tfdt" box. Default is enabled, which will prefer to use the "tfdt" box to set DTS. Disable to use the "earliest_presentation_time" from the "sidc" box. In either case, the timestamp from the "mfra" box will be used if it's available and "use_mfra_for" is set to pts or dts.

export_all

Export unrecognized boxes within the *udta* box as metadata entries. The first four characters of the box type are set as the key. Default is false.

export_xmp

Export entire contents of *XMP_* box and *uuid* box as a string with key "xmp". Note that if "export_all" is set and this option isn't, the contents of *XMP_* box are still exported but with key

"XMP_". Default is false.

activation_bytes

4-byte key required to decrypt Audible AAX and AAX+ files. See Audible AAX subsection below.

audible_fixed_key

Fixed key used for handling Audible AAX/AAX+ files. It has been pre-set so should not be necessary to specify.

decryption_key

16-byte key, in hex, to decrypt files encrypted using ISO Common Encryption (CENC/AES-128 CTR; ISO/IEC 23001-7).

max_stts_delta

Very high sample deltas written in a trak's stts box may occasionally be intended but usually they are written in error or used to store a negative value for dts correction when treated as signed 32-bit integers. This option lets the user set an upper limit, beyond which the delta is clamped to 1. Values greater than the limit if negative when cast to int32 are used to adjust onward dts.

Unit is the track time scale. Range is 0 to UINT_MAX. Default is "UINT_MAX - 48000*10" which allows upto a 10 second dts correction for 48 kHz audio streams while accommodating 99.9% of "uint32" range.

Audible AAX

Audible AAX files are encrypted M4B files, and they can be decrypted by specifying a 4 byte activation secret.

```
ffmpeg -activation_bytes 1CEB00DA -i test.aax -vn -c:a copy output.mp4
```

mpegts

MPEG-2 transport stream demuxer.

This demuxer accepts the following options:

resync_size

Set size limit for looking up a new synchronization. Default value is 65536.

skip_unknown_pmt

Skip PMTs for programs not defined in the PAT. Default value is 0.

fix_teletext_pts

Override teletext packet PTS and DTS values with the timestamps calculated from the PCR of the first program which the teletext stream is part of and is not discarded. Default value is 1, set this option to 0 if you want your teletext packet PTS and DTS values untouched.

ts_packetsize

Output option carrying the raw packet size in bytes. Show the detected raw packet size, cannot be set by the user.

scan_all_pmts

Scan and combine all PMTs. The value is an integer with value from -1 to 1 (-1 means automatic setting, 1 means enabled, 0 means disabled). Default value is -1.

merge_pmt_versions

Re-use existing streams when a PMT's version is updated and elementary streams move to different PIDs. Default value is 0.

max_packet_size

Set maximum size, in bytes, of packet emitted by the demuxer. Payloads above this size are split across multiple packets. Range is 1 to INT_MAX/2. Default is 204800 bytes.

mpjpeg

MJPEG encapsulated in multi-part MIME demuxer.

This demuxer allows reading of MJPEG, where each frame is represented as a part of multipart/x-mixed-replace stream.

strict_mime_boundary

Default implementation applies a relaxed standard to multi-part MIME boundary detection, to prevent regression with numerous existing endpoints not generating a proper MIME MJPEG stream. Turning this option on by setting it to 1 will result in a stricter check of the boundary value.

rawvideo

Raw video demuxer.

This demuxer allows one to read raw video data. Since there is no header specifying the assumed video parameters, the user must specify them in order to be able to decode the data correctly.

This demuxer accepts the following options:

framerate

Set input video frame rate. Default value is 25.

pixel_format

Set the input video pixel format. Default value is "yuv420p".

video_size

Set the input video size. This value must be specified explicitly.

For example to read a rawvideo file *input.raw* with **ffplay**, assuming a pixel format of "rgb24", a video size of "320x240", and a frame rate of 10 images per second, use the command:

```
ffplay -f rawvideo -pixel_format rgb24 -video_size 320x240 -framerate 10 input.raw
```

sbg

SBaGen script demuxer.

This demuxer reads the script language used by SBaGen <<http://uazu.net/sbagen/>> to generate binaural beats sessions. A SBG script looks like that:

```
-SE
a: 300-2.5/3 440+4.5/0
b: 300-2.5/0 440+4.5/3
off: -
NOW    == a
+0:07:00 == b
+0:14:00 == a
+0:21:00 == b
+0:30:00 off
```

A SBG script can mix absolute and relative timestamps. If the script uses either only absolute timestamps (including the script start time) or only relative ones, then its layout is fixed, and the conversion is straightforward. On the other hand, if the script mixes both kind of timestamps, then the *NOW* reference for relative timestamps will be taken from the current time of day at the time the script is read, and the script layout will be frozen according to that reference. That means that if the script is directly played, the actual times will match the absolute timestamps up to the sound controller's clock accuracy, but if the user somehow pauses the playback or seeks, all times will be shifted accordingly.

tedcaptions

JSON captions used for `<http://www.ted.com/>`.

TED does not provide links to the captions, but they can be guessed from the page. The file *tools/bookmarklets.html* from the FFmpeg source tree contains a bookmarklet to expose them.

This demuxer accepts the following option:

start_time

Set the start time of the TED talk, in milliseconds. The default is 15000 (15s). It is used to sync the captions with the downloadable videos, because they include a 15s intro.

Example: convert the captions to a format most players understand:

```
ffmpeg -i http://www.ted.com/talks/subtitles/id/1/lang/en talk1-en.srt
```

vapoursynth

Vapoursynth wrapper.

Due to security concerns, Vapoursynth scripts will not be autodetected so the input format has to be forced. For ff* CLI tools, add `"-f vapoursynth"` before the input `"-i yoursript.vpy"`.

This demuxer accepts the following option:

max_script_size

The demuxer buffers the entire script into memory. Adjust this value to set the maximum buffer size, which in turn, acts as a ceiling for the size of scripts that can be read. Default is 1 MiB.

MUXERS

Muxers are configured elements in FFmpeg which allow writing multimedia streams to a particular type of file.

When you configure your FFmpeg build, all the supported muxers are enabled by default. You can list all available muxers using the configure option `"--list-muxers"`.

You can disable all the muxers with the configure option `"--disable-muxers"` and selectively enable / disable single muxers with the options `"--enable-muxer=MUXER"` / `"--disable-muxer=MUXER"`.

The option `"-muxers"` of the ff* tools will display the list of enabled muxers. Use `"-formats"` to view a combined list of enabled demuxers and muxers.

A description of some of the currently available muxers follows.

a64

A64 muxer for Commodore 64 video. Accepts a single "a64_multi" or "a64_multi5" codec video stream.

adts

Audio Data Transport Stream muxer. It accepts a single AAC stream.

Options

It accepts the following options:

write_id3v2 *bool*

Enable to write ID3v2.4 tags at the start of the stream. Default is disabled.

write_apetag *bool*

Enable to write APE tags at the end of the stream. Default is disabled.

write_mpeg2 *bool*

Enable to set MPEG version bit in the ADTS frame header to 1 which indicates MPEG-2. Default is 0, which indicates MPEG-4.

aiff

Audio Interchange File Format muxer.

Options

It accepts the following options:

write_id3v2

Enable ID3v2 tags writing when set to 1. Default is 0 (disabled).

id3v2_version

Select ID3v2 version to write. Currently only version 3 and 4 (aka. ID3v2.3 and ID3v2.4) are supported. The default is version 4.

alp

Muxer for audio of High Voltage Software's Lego Racers game. It accepts a single ADPCM_IMA_ALP stream with no more than 2 channels nor a sample rate greater than 44100 Hz.

Extensions: tun, pcm

Options

It accepts the following options:

type *type*

Set file type.

tun Set file type as music. Must have a sample rate of 22050 Hz.

pcm Set file type as sfx.

auto Set file type as per output file extension. ".pcm" results in type "pcm" else type "tun" is set.
(*default*)

asf

Advanced Systems Format muxer.

Note that Windows Media Audio (wma) and Windows Media Video (wmv) use this muxer too.

Options

It accepts the following options:

packet_size

Set the muxer packet size. By tuning this setting you may reduce data fragmentation or muxer overhead depending on your source. Default value is 3200, minimum is 100, maximum is 64k.

avi

Audio Video Interleaved muxer.

Options

It accepts the following options:

reserve_index_space

Reserve the specified amount of bytes for the OpenDML master index of each stream within the file header. By default additional master indexes are embedded within the data packets if there is no space left in the first master index and are linked together as a chain of indexes. This index

structure can cause problems for some use cases, e.g. third-party software strictly relying on the OpenDML index specification or when file seeking is slow. Reserving enough index space in the file header avoids these problems.

The required index space depends on the output file size and should be about 16 bytes per gigabyte. When this option is omitted or set to zero the necessary index space is guessed.

write_channel_mask

Write the channel layout mask into the audio stream header.

This option is enabled by default. Disabling the channel mask can be useful in specific scenarios, e.g. when merging multiple audio streams into one for compatibility with software that only supports a single audio stream in AVI (see **the "amerge" section in the ffmpeg-filters manual**).

flipped_raw_rgb

If set to true, store positive height for raw RGB bitmaps, which indicates bitmap is stored bottom-up. Note that this option does not flip the bitmap which has to be done manually beforehand, e.g. by using the `vflip` filter. Default is *false* and indicates bitmap is stored top down.

chromaprint

Chromaprint fingerprinter.

This muxer feeds audio data to the Chromaprint library, which generates a fingerprint for the provided audio data. See <<https://acoustid.org/chromaprint>>

It takes a single signed native-endian 16-bit raw audio stream of at most 2 channels.

Options

silence_threshold

Threshold for detecting silence. Range is from -1 to 32767, where -1 disables silence detection. Silence detection can only be used with version 3 of the algorithm. Silence detection must be disabled for use with the AcoustID service. Default is -1.

algorithm

Version of algorithm to fingerprint with. Range is 0 to 4. Version 3 enables silence detection. Default is 1.

fp_format

Format to output the fingerprint as. Accepts the following options:

raw Binary raw fingerprint

compressed

Binary compressed fingerprint

base64

Base64 compressed fingerprint (*default*)

crc

CRC (Cyclic Redundancy Check) testing format.

This muxer computes and prints the Adler-32 CRC of all the input audio and video frames. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the CRC.

The output of the muxer consists of a single line of the form: `CRC=0xCRC`, where *CRC* is a hexadecimal number 0-padded to 8 digits containing the CRC for all the decoded input frames.

See also the **framecrc** muxer.

Examples

For example to compute the CRC of the input, and store it in the file *out.crc*:

```
ffmpeg -i INPUT -f crc out.crc
```

You can print the CRC to stdout with the command:

```
ffmpeg -i INPUT -f crc -
```

You can select the output format of each frame with **ffmpeg** by specifying the audio and video codec and format. For example to compute the CRC of the input audio converted to PCM unsigned 8-bit and the input video converted to MPEG-2 video, use the command:

```
ffmpeg -i INPUT -c:a pcm_u8 -c:v mpeg2video -f crc -
```

dash

Dynamic Adaptive Streaming over HTTP (DASH) muxer that creates segments and manifest files according to the MPEG-DASH standard ISO/IEC 23009-1:2014.

For more information see:

- ⊕ ISO DASH Specification:
<http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip>
- ⊕ WebM DASH Specification:
<<https://sites.google.com/a/webmproject.org/wiki/adaptive-streaming/webm-dash-specification>>

It creates a MPD manifest file and segment files for each stream.

The segment filename might contain pre-defined identifiers used with SegmentTemplate as defined in section 5.3.9.4.4 of the standard. Available identifiers are "\$RepresentationID\$", "\$Number\$", "\$Bandwidth\$" and "\$Time\$". In addition to the standard identifiers, an ffmpeg-specific "\$ext\$" identifier is also supported. When specified ffmpeg will replace \$ext\$ in the file name with muxing format's extensions such as mp4, webm etc.,

```
ffmpeg -re -i <input> -map 0 -map 0 -c:a libfdk_aac -c:v libx264 \
-b:v:0 800k -b:v:1 300k -s:v:1 320x170 -profile:v:1 baseline \
-profile:v:0 main -bf 1 -keyint_min 120 -g 120 -sc_threshold 0 \
-b_strategy 0 -ar:a:1 22050 -use_timeline 1 -use_template 1 \
-window_size 5 -adaptation_sets "id=0,streams=v id=1,streams=a" \
-f dash /path/to/out.mpd
```

seg_duration *duration*

Set the segment length in seconds (fractional value can be set). The value is treated as average segment duration when *use_template* is enabled and *use_timeline* is disabled and as minimum segment duration for all the other use cases.

frag_duration *duration*

Set the length in seconds of fragments within segments (fractional value can be set).

frag_type *type*

Set the type of interval for fragmentation.

window_size *size*

Set the maximum number of segments kept in the manifest.

extra_window_size *size*

Set the maximum number of segments kept outside of the manifest before removing from disk.

remove_at_exit *remove*

Enable (1) or disable (0) removal of all segments when finished.

use_template *template*

Enable (1) or disable (0) use of SegmentTemplate instead of SegmentList.

use_timeline *timeline*

Enable (1) or disable (0) use of SegmentTimeline in SegmentTemplate.

single_file *single_file*

Enable (1) or disable (0) storing all segments in one file, accessed using byte ranges.

single_file_name *file_name*

DASH-templated name to be used for baseURL. Implies *single_file* set to "1". In the template, "\$ext\$" is replaced with the file name extension specific for the segment format.

init_seg_name *init_name*

DASH-templated name to be used for the initialization segment. Default is "init-stream\$RepresentationID\$. \$ext\$". "\$ext\$" is replaced with the file name extension specific for the segment format.

media_seg_name *segment_name*

DASH-templated name to be used for the media segments. Default is "chunk-stream\$RepresentationID\$-\$Number%05d\$. \$ext\$". "\$ext\$" is replaced with the file name extension specific for the segment format.

utc_timing_url *utc_url*

URL of the page that will return the UTC timestamp in ISO format. Example:
"https://time.akamai.com/?iso"

method *method*

Use the given HTTP method to create output files. Generally set to PUT or POST.

http_user_agent *user_agent*

Override User-Agent field in HTTP header. Applicable only for HTTP output.

http_persistent *http_persistent*

Use persistent HTTP connections. Applicable only for HTTP output.

hls_playlist *hls_playlist*

Generate HLS playlist files as well. The master playlist is generated with the filename *hls_master_name*. One media playlist file is generated for each stream with filenames *media_0.m3u8*, *media_1.m3u8*, etc.

hls_master_name *file_name*

HLS master playlist name. Default is "master.m3u8".

streaming *streaming*

Enable (1) or disable (0) chunk streaming mode of output. In chunk streaming mode, each frame will be a moof fragment which forms a chunk.

adaptation_sets *adaptation_sets*

Assign streams to AdaptationSets. Syntax is "id=x,streams=a,b,c id=y,streams=d,e" with x and y being the IDs of the adaptation sets and a,b,c,d and e are the indices of the mapped streams.

To map all video (or audio) streams to an AdaptationSet, "v" (or "a") can be used as stream identifier instead of IDs.

When no assignment is defined, this defaults to an AdaptationSet for each stream.

Optional syntax is

"id=x,seg_duration=x,frag_duration=x,frag_type=type,descriptor=descriptor_string,streams=a,b,c id=y,seg_duration=y,frag_type=type,streams=d,e" and so on, descriptor is useful to the scheme defined by ISO/IEC 23009-1:2014/Amd.2:2015. For example, -adaptation_sets "id=0,descriptor=<SupplementalProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="\0,0,0,1,1,2,2"/>,streams=v". Please note that descriptor string should be a self-closing xml tag. *seg_duration*, *frag_duration* and *frag_type* override the global option values for each adaptation set. For example, -adaptation_sets "id=0,seg_duration=2,frag_duration=1,frag_type=duration,streams=v id=1,seg_duration=2,frag_type=none,streams=a" *type_id* marks an adaptation set as containing streams meant to be used for Trick Mode for the referenced adaptation set. For example, -adaptation_sets "id=0,seg_duration=2,frag_type=none,streams=0 id=1,seg_duration=10,frag_type=none,trick_id=0,streams=1"

timeout *timeout*

Set timeout for socket I/O operations. Applicable only for HTTP output.

index_correction *index_correction*

Enable (1) or Disable (0) segment index correction logic. Applicable only when *use_template* is enabled and *use_timeline* is disabled.

When enabled, the logic monitors the flow of segment indexes. If a stream's segment index value is not at the expected real time position, then the logic corrects that index value.

Typically this logic is needed in live streaming use cases. The network bandwidth fluctuations are common during long run streaming. Each fluctuation can cause the segment indexes fall behind the expected real time position.

format_options *options_list*

Set container format (mp4/webm) options using a ":" separated list of key=value parameters. Values containing ":" special characters must be escaped.

global_sidx *global_sidx*

Write global SIDX atom. Applicable only for single file, mp4 output, non-streaming mode.

dash_segment_type *dash_segment_type*

Possible values:

auto If this flag is set, the dash segment files format will be selected based on the stream codec. This is the default mode.

mp4

If this flag is set, the dash segment files will be in in ISOBMFF format.

webm

If this flag is set, the dash segment files will be in in WebM format.

ignore_io_errors *ignore_io_errors*

Ignore IO errors during open and write. Useful for long-duration runs with network output.

lhls *lhls*

Enable Low-latency HLS(LHLS). Adds #EXT-X-PREFETCH tag with current segment's URI. hls.js player folks are trying to standardize an open LHLS spec. The draft spec is available in <https://github.com/video-dev/hlsjs-rfcs/blob/lhls-spec/proposals/0001-lhls.md> This option tries to comply with the above open spec. It enables *streaming* and *hls_playlist* options automatically. This is an experimental feature.

Note: This is not Apple's version LHLS. See

<<https://datatracker.ietf.org/doc/html/draft-pantos-hls-rfc8216bis>>

ldash *ldash*

Enable Low-latency Dash by constraining the presence and values of some elements.

master_m3u8_publish_rate *master_m3u8_publish_rate*

Publish master playlist repeatedly every after specified number of segment intervals.

write_prft *write_prft*

Write Producer Reference Time elements on supported streams. This also enables writing prft boxes in the underlying muxer. Applicable only when the *utc_url* option is enabled. It's set to auto by default, in which case the muxer will attempt to enable it only in modes that require it.

mpd_profile *mpd_profile*

Set one or more manifest profiles.

http_opts *http_opts*

A :-separated list of key=value options to pass to the underlying HTTP protocol. Applicable only for HTTP output.

target_latency *target_latency*

Set an intended target latency in seconds (fractional value can be set) for serving. Applicable only when *streaming* and *write_prft* options are enabled. This is an informative fields clients can use to measure the latency of the service.

min_playback_rate *min_playback_rate*

Set the minimum playback rate indicated as appropriate for the purposes of automatically adjusting playback latency and buffer occupancy during normal playback by clients.

max_playback_rate *max_playback_rate*

Set the maximum playback rate indicated as appropriate for the purposes of automatically adjusting playback latency and buffer occupancy during normal playback by clients.

update_period *update_period*

Set the mpd update period ,for dynamic content.
The unit is second.

fifo

The fifo pseudo-muxer allows the separation of encoding and muxing by using first-in-first-out queue and running the actual muxer in a separate thread. This is especially useful in combination with the **tee** muxer and can be used to send data to several destinations with different reliability/writing speed/latency.

API users should be aware that callback functions (`interrupt_callback`, `io_open` and `io_close`) used within its `AVFormatContext` must be thread-safe.

The behavior of the fifo muxer if the queue fills up or if the output fails is selectable,

- ⊕ output can be transparently restarted with configurable delay between retries based on real time or time of the processed stream.
- ⊕ encoding can be blocked during temporary failure, or continue transparently dropping packets in case fifo queue fills up.

fifo_format

Specify the format name. Useful if it cannot be guessed from the output name suffix.

queue_size

Specify size of the queue (number of packets). Default value is 60.

format_opts

Specify format options for the underlying muxer. Muxer options can be specified as a list of *key=value* pairs separated by `'`.

drop_pkts_on_overflow *bool*

If set to 1 (true), in case the fifo queue fills up, packets will be dropped rather than blocking the encoder. This makes it possible to continue streaming without delaying the input, at the cost of omitting part of the stream. By default this option is set to 0 (false), so in such cases the encoder will be blocked until the muxer processes some of the packets and none of them is lost.

attempt_recovery *bool*

If failure occurs, attempt to recover the output. This is especially useful when used with network output, since it makes it possible to restart streaming transparently. By default this option is set to 0 (false).

max_recovery_attempts

Sets maximum number of successive unsuccessful recovery attempts after which the output fails permanently. By default this option is set to 0 (unlimited).

recovery_wait_time *duration*

Waiting time before the next recovery attempt after previous unsuccessful recovery attempt. Default value is 5 seconds.

recovery_wait_streamtime *bool*

If set to 0 (false), the real time is used when waiting for the recovery attempt (i.e. the recovery will be attempted after at least `recovery_wait_time` seconds). If set to 1 (true), the time of the processed stream is taken into account instead (i.e. the recovery will be attempted after at least `recovery_wait_time` seconds of the stream is omitted). By default, this option is set to 0 (false).

recover_any_error *bool*

If set to 1 (true), recovery will be attempted regardless of type of the error causing the failure. By default this option is set to 0 (false) and in case of certain (usually permanent) errors the recovery is not attempted even when `attempt_recovery` is set to 1.

restart_with_keyframe *bool*

Specify whether to wait for the keyframe after recovering from queue overflow or failure. This option is set to 0 (false) by default.

timeshift *duration*

Buffer the specified amount of packets and delay writing the output. Note that `queue_size` must be big enough to store the packets for timeshift. At the end of the input the fifo buffer is flushed at realtime speed.

Examples

- ⊕ Stream something to rtmp server, continue processing the stream at real-time rate even in case of temporary failure (network outage) and attempt to recover streaming every second indefinitely.

```
ffmpeg -re -i ... -c:v libx264 -c:a aac -f fifo -fifo_format flv -map 0:v -map 0:a
      -drop_pkts_on_overflow 1 -attempt_recovery 1 -recovery_wait_time 1 rtmp://example.com/live/stream_name
```

flv

Adobe Flash Video Format muxer.

This muxer accepts the following options:

flvflags *flags*

Possible values:

aac_seq_header_detect

Place AAC sequence header based on audio stream data.

no_sequence_end

Disable sequence end tag.

no_metadata

Disable metadata tag.

no_duration_filesize

Disable duration and filesize in metadata when they are equal to zero at the end of stream.
(Be used to non-seekable living stream).

add_keyframe_index

Used to facilitate seeking; particularly for HTTP pseudo streaming.

framecrc

Per-packet CRC (Cyclic Redundancy Check) testing format.

This muxer computes and prints the Adler-32 CRC for each audio and video packet. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the CRC.

The output of the muxer consists of a line for each audio and video packet of the form:

```
<stream_index>, <packet_dts>, <packet_pts>, <packet_duration>, <packet_size>, 0x<CRC>
```

CRC is a hexadecimal number 0-padded to 8 digits containing the CRC of the packet.

Examples

For example to compute the CRC of the audio and video frames in *INPUT*, converted to raw audio and video packets, and store it in the file *out.crc*:

```
ffmpeg -i INPUT -f framecrc out.crc
```

To print the information to stdout, use the command:

```
ffmpeg -i INPUT -f framecrc -
```

With **ffmpeg**, you can select the output format to which the audio and video frames are encoded before computing the CRC for each packet by specifying the audio and video codec. For example, to compute the CRC of each decoded input audio frame converted to PCM unsigned 8-bit and of each decoded input video frame converted to MPEG-2 video, use the command:


```
ffmpeg -i INPUT -c:a pcm_u8 -c:v mpeg2video -f framecrc -
```

See also the **crc** muxer.

framehash

Per-packet hash testing format.

This muxer computes and prints a cryptographic hash for each audio and video packet. This can be used for packet-by-packet equality checks without having to individually do a binary comparison on each.

By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash, but the output of explicit conversions to other codecs can also be used. It uses the SHA-256 cryptographic hash function by default, but supports several other algorithms.

The output of the muxer consists of a line for each audio and video packet of the form:

```
<stream_index>, <packet_dts>, <packet_pts>, <packet_duration>, <packet_size>, <hash>
```

hash is a hexadecimal number representing the computed hash for the packet.

hash algorithm

Use the cryptographic hash function specified by the string *algorithm*. Supported values include "MD5", "murmur3", "RIPEMD128", "RIPEMD160", "RIPEMD256", "RIPEMD320", "SHA160", "SHA224", "SHA256" (default), "SHA512/224", "SHA512/256", "SHA384", "SHA512", "CRC32" and "adler32".

Examples

To compute the SHA-256 hash of the audio and video frames in *INPUT*, converted to raw audio and video packets, and store it in the file *out.sha256*:

```
ffmpeg -i INPUT -f framehash out.sha256
```

To print the information to stdout, using the MD5 hash function, use the command:

```
ffmpeg -i INPUT -f framehash -hash md5 -
```

See also the **hash** muxer.

framemd5

Per-packet MD5 testing format.

This is a variant of the **framehash** muxer. Unlike that muxer, it defaults to using the MD5 hash function.

Examples

To compute the MD5 hash of the audio and video frames in *INPUT*, converted to raw audio and video packets, and store it in the file *out.md5*:

```
ffmpeg -i INPUT -f framemd5 out.md5
```

To print the information to stdout, use the command:

```
ffmpeg -i INPUT -f framemd5 -
```

See also the **framehash** and **md5** muxers.

gif

Animated GIF muxer.

It accepts the following options:

loop

Set the number of times to loop the output. Use "-1" for no loop, 0 for looping indefinitely (default).

final_delay

Force the delay (expressed in centiseconds) after the last frame. Each frame ends with a delay until the next frame. The default is "-1", which is a special value to tell the muxer to re-use the previous delay. In case of a loop, you might want to customize this value to mark a pause for instance.

For example, to encode a gif looping 10 times, with a 5 seconds delay between the loops:

```
ffmpeg -i INPUT -loop 10 -final_delay 500 out.gif
```

Note 1: if you wish to extract the frames into separate GIF files, you need to force the **image2** muxer:

```
ffmpeg -i INPUT -c:v gif -f image2 "out%d.gif"
```

Note 2: the GIF format has a very large time base: the delay between two frames can therefore not be smaller than one centi second.

hash

Hash testing format.

This muxer computes and prints a cryptographic hash of all the input audio and video frames. This can be used for equality checks without having to do a complete binary comparison.

By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash, but the output of explicit conversions to other codecs can also be used.

Timestamps are ignored. It uses the SHA-256 cryptographic hash function by default, but supports several other algorithms.

The output of the muxer consists of a single line of the form: *algo=hash*, where *algo* is a short string representing the hash function used, and *hash* is a hexadecimal number representing the computed hash.

hash algorithm

Use the cryptographic hash function specified by the string *algorithm*. Supported values include "MD5", "murmur3", "RIPEMD128", "RIPEMD160", "RIPEMD256", "RIPEMD320", "SHA160", "SHA224", "SHA256" (default), "SHA512/224", "SHA512/256", "SHA384", "SHA512", "CRC32" and "adler32".

Examples

To compute the SHA-256 hash of the input converted to raw audio and video, and store it in the file *out.sha256*:

```
ffmpeg -i INPUT -f hash out.sha256
```

To print an MD5 hash to stdout use the command:

```
ffmpeg -i INPUT -f hash -hash md5 -
```

See also the **framehash** muxer.

hls

Apple HTTP Live Streaming muxer that segments MPEG-TS according to the HTTP Live Streaming (HLS) specification.

It creates a playlist file, and one or more segment files. The output filename specifies the playlist filename.

By default, the muxer creates a file for each segment produced. These files have the same name as the playlist, followed by a sequential number and a .ts extension.

Make sure to require a closed GOP when encoding and to set the GOP size to fit your segment time constraint.

For example, to convert an input file with **ffmpeg**:

```
ffmpeg -i in.mkv -c:v h264 -flags +cgop -g 30 -hls_time 1 out.m3u8
```

This example will produce the playlist, *out.m3u8*, and segment files: *out0.ts*, *out1.ts*, *out2.ts*, etc.

See also the **segment** muxer, which provides a more generic and flexible implementation of a segmenter, and can be used to perform HLS segmentation.

Options

This muxer supports the following options:

hls_init_time *duration*

Set the initial target segment length. Default value is *0*.

duration must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1) manual**.

Segment will be cut on the next key frame after this time has passed on the first m3u8 list. After the initial playlist is filled **ffmpeg** will cut segments at duration equal to "hls_time"

hls_time *duration*

Set the target segment length. Default value is *2*.

duration must be a time duration specification, see **the Time duration section in the ffmpeg-utils(1) manual**. Segment will be cut on the next key frame after this time has passed.

hls_list_size *size*

Set the maximum number of playlist entries. If set to 0 the list file will contain all the segments. Default value is 5.

hls_delete_threshold *size*

Set the number of unreferenced segments to keep on disk before "hls_flags delete_segments" deletes them. Increase this to allow continue clients to download segments which were recently referenced in the playlist. Default value is 1, meaning segments older than "hls_list_size+1" will be deleted.

hls_start_number_source

Start the playlist sequence number ("#EXT-X-MEDIA-SEQUENCE") according to the specified source. Unless "hls_flags single_file" is set, it also specifies source of starting sequence numbers of segment and subtitle filenames. In any case, if "hls_flags append_list" is set and read playlist sequence number is greater than the specified start sequence number, then that value will be used as start value.

It accepts the following values:

generic (default)

Set the starting sequence numbers according to *start_number* option value.

epoch

The start number will be the seconds since epoch (1970-01-01 00:00:00)

epoch_us

The start number will be the microseconds since epoch (1970-01-01 00:00:00)

datetime

The start number will be based on the current date/time as YYYYmmddHHMMSS. e.g. 20161231235759.

start_number *number*

Start the playlist sequence number ("#EXT-X-MEDIA-SEQUENCE") from the specified *number* when *hls_start_number_source* value is *generic*. (This is the default case.) Unless "hls_flags single_file" is set, it also specifies starting sequence numbers of segment and subtitle filenames. Default value is 0.

hls_allow_cache *allowcache*

Explicitly set whether the client MAY (1) or MUST NOT (0) cache media segments.

hls_base_url *baseurl*

Append *baseurl* to every entry in the playlist. Useful to generate playlists with absolute paths.

Note that the playlist sequence number must be unique for each segment and it is not to be confused with the segment filename sequence number which can be cyclic, for example if the **wrap** option is specified.

hls_segment_filename *filename*

Set the segment filename. Unless "hls_flags single_file" is set, *filename* is used as a string format with the segment number:

```
ffmpeg -i in.nut -hls_segment_filename 'file%03d.ts' out.m3u8
```

This example will produce the playlist, *out.m3u8*, and segment files: *file000.ts*, *file001.ts*, *file002.ts*, etc.

filename may contain full path or relative path specification, but only the file name part without any path info will be contained in the m3u8 segment list. Should a relative path be specified, the path of the created segment files will be relative to the current working directory. When `strftime_mkdir` is set, the whole expanded value of *filename* will be written into the m3u8 segment list.

When "var_stream_map" is set with two or more variant streams, the *filename* pattern must contain the string "%v", this string specifies the position of variant stream index in the generated segment file names.

```
ffmpeg -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k -b:a:1 32k \
  -map 0:v -map 0:a -map 0:v -map 0:a -f hls -var_stream_map "v:0,a:0 v:1,a:1" \
  -hls_segment_filename 'file_%v_%03d.ts' out_%v.m3u8
```

This example will produce the playlists segment file sets: *file_0_000.ts*, *file_0_001.ts*, *file_0_002.ts*, etc. and *file_1_000.ts*, *file_1_001.ts*, *file_1_002.ts*, etc.

The string "%v" may be present in the filename or in the last directory name containing the file, but only in one of them. (Additionally, %v may appear multiple times in the last sub-directory or filename.) If the string %v is present in the directory name, then sub-directories are created after expanding the directory name pattern. This enables creation of segments corresponding to different variant streams in subdirectories.

```
ffmpeg -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k -b:a:1 32k \
```

```
-map 0:v -map 0:a -map 0:v -map 0:a -f hls -var_stream_map "v:0,a:0 v:1,a:1" \
-hls_segment_filename 'vs%v/file_%03d.ts' vs%v/out.m3u8
```

This example will produce the playlists segment file sets: *vs0/file_000.ts*, *vs0/file_001.ts*, *vs0/file_002.ts*, etc. and *vs1/file_000.ts*, *vs1/file_001.ts*, *vs1/file_002.ts*, etc.

strftime

Use **strftime()** on *filename* to expand the segment filename with localtime. The segment number is also available in this mode, but to use it, you need to specify `second_level_segment_index` `hls_flag` and `%%d` will be the specifier.

```
ffmpeg -i in.nut -strftime 1 -hls_segment_filename 'file-%Y%m%d-%s.ts' out.m3u8
```

This example will produce the playlist, *out.m3u8*, and segment files: *file-20160215-1455569023.ts*, *file-20160215-1455569024.ts*, etc. Note: On some systems/environments, the `%s` specifier is not available. See "strftime()" documentation.

```
ffmpeg -i in.nut -strftime 1 -hls_flags second_level_segment_index -hls_segment_filename 'file-%Y%m%d-
```

This example will produce the playlist, *out.m3u8*, and segment files: *file-20160215-0001.ts*, *file-20160215-0002.ts*, etc.

strftime_mkdir

Used together with `-strftime_mkdir`, it will create all subdirectories which is expanded in *filename*.

```
ffmpeg -i in.nut -strftime 1 -strftime_mkdir 1 -hls_segment_filename '%Y%m%d/file-%Y%m%d-%s.ts' out.
```

This example will create a directory 201560215 (if it does not exist), and then produce the playlist, *out.m3u8*, and segment files: *20160215/file-20160215-1455569023.ts*, *20160215/file-20160215-1455569024.ts*, etc.

```
ffmpeg -i in.nut -strftime 1 -strftime_mkdir 1 -hls_segment_filename '%Y/%m/%d/file-%Y%m%d-%s.ts' ou
```

This example will create a directory hierarchy 2016/02/15 (if any of them do not exist), and then produce the playlist, *out.m3u8*, and segment files: *2016/02/15/file-20160215-1455569023.ts*, *2016/02/15/file-20160215-1455569024.ts*, etc.

hls_segment_options options_list

Set output format options using a `:`-separated list of `key=value` parameters. Values containing `:"`

special characters must be escaped.

hls_key_info_file *key_info_file*

Use the information in *key_info_file* for segment encryption. The first line of *key_info_file* specifies the key URI written to the playlist. The key URL is used to access the encryption key during playback. The second line specifies the path to the key file used to obtain the key during the encryption process. The key file is read as a single packed array of 16 octets in binary format. The optional third line specifies the initialization vector (IV) as a hexadecimal string to be used instead of the segment sequence number (default) for encryption. Changes to *key_info_file* will result in segment encryption with the new key/IV and an entry in the playlist for the new key URI/IV if "hls_flags periodic_rekey" is enabled.

Key info file format:

```
<key URI>
<key file path>
<IV> (optional)
```

Example key URIs:

```
http://server/file.key
/path/to/file.key
file.key
```

Example key file paths:

```
file.key
/path/to/file.key
```

Example IV:

```
0123456789ABCDEF0123456789ABCDEF
```

Key info file example:

```
http://server/file.key
/path/to/file.key
0123456789ABCDEF0123456789ABCDEF
```

Example shell script:


```
#!/bin/sh
BASE_URL=${1:-'.'}
openssl rand 16 > file.key
echo $BASE_URL/file.key > file.keyinfo
echo file.key >> file.keyinfo
echo $(openssl rand -hex 16) >> file.keyinfo
ffmpeg -f lavfi -re -i testsrc -c:v h264 -hls_flags delete_segments \
-hls_key_info_file file.keyinfo out.m3u8
```

-hls_enc *enc*

Enable (1) or disable (0) the AES128 encryption. When enabled every segment generated is encrypted and the encryption key is saved as *playlist name.key*.

-hls_enc_key *key*

16-octet key to encrypt the segments, by default it is randomly generated.

-hls_enc_key_url *keyurl*

If set, *keyurl* is prepended instead of *baseurl* to the key filename in the playlist.

-hls_enc_iv *iv*

16-octet initialization vector for every segment instead of the autogenerated ones.

hls_segment_type *flags*

Possible values:

mpegts

Output segment files in MPEG-2 Transport Stream format. This is compatible with all HLS versions.

fmp4

Output segment files in fragmented MP4 format, similar to MPEG-DASH. fmp4 files may be used in HLS version 7 and above.

hls_fmp4_init_filename *filename*

Set filename to the fragment files header file, default filename is *init.mp4*.

Use "-strftime 1" on *filename* to expand the segment filename with localtime.

```
ffmpeg -i in.nut -hls_segment_type fmp4 -strftime 1 -hls_fmp4_init_filename "%s_init.mp4" out.m3u8
```

This will produce init like this *1602678741_init.mp4*

hls_fmp4_init_resend

Resend init file after m3u8 file refresh every time, default is *0*.

When "var_stream_map" is set with two or more variant streams, the *filename* pattern must contain the string "%v", this string specifies the position of variant stream index in the generated init file names. The string "%v" may be present in the filename or in the last directory name containing the file. If the string is present in the directory name, then sub-directories are created after expanding the directory name pattern. This enables creation of init files corresponding to different variant streams in subdirectories.

hls_flags flags

Possible values:

single_file

If this flag is set, the muxer will store all segments in a single MPEG-TS file, and will use byte ranges in the playlist. HLS playlists generated with this way will have the version number 4. For example:

```
ffmpeg -i in.nut -hls_flags single_file out.m3u8
```

Will produce the playlist, *out.m3u8*, and a single segment file, *out.ts*.

delete_segments

Segment files removed from the playlist are deleted after a period of time equal to the duration of the segment plus the duration of the playlist.

append_list

Append new segments into the end of old segment list, and remove the "#EXT-X-ENDLIST" from the old segment list.

round_durations

Round the duration info in the playlist file segment info to integer values, instead of using floating point. If there are no other features requiring higher HLS versions be used, then this will allow ffmpeg to output a HLS version 2 m3u8.

discont_start

Add the "#EXT-X-DISCONTINUITY" tag to the playlist, before the first segment's information.

omit_endlist

Do not append the "EXT-X-ENDLIST" tag at the end of the playlist.

periodic_rekey

The file specified by "hls_key_info_file" will be checked periodically and detect updates to the encryption info. Be sure to replace this file atomically, including the file containing the AES encryption key.

independent_segments

Add the "#EXT-X-INDEPENDENT-SEGMENTS" to playlists that has video segments and when all the segments of that playlist are guaranteed to start with a Key frame.

iframes_only

Add the "#EXT-X-I-FRAMES-ONLY" to playlists that has video segments and can play only I-frames in the "#EXT-X-BYTERANGE" mode.

split_by_time

Allow segments to start on frames other than keyframes. This improves behavior on some players when the time between keyframes is inconsistent, but may make things worse on others, and can cause some oddities during seeking. This flag should be used with the "hls_time" option.

program_date_time

Generate "EXT-X-PROGRAM-DATE-TIME" tags.

second_level_segment_index

Makes it possible to use segment indexes as %%d in hls_segment_filename expression besides date/time values when strftime is on. To get fixed width numbers with trailing zeroes, %%0xd format is available where x is the required width.

second_level_segment_size

Makes it possible to use segment sizes (counted in bytes) as %%s in hls_segment_filename expression besides date/time values when strftime is on. To get fixed width numbers with trailing zeroes, %%0xs format is available where x is the required width.

second_level_segment_duration

Makes it possible to use segment duration (calculated in microseconds) as %%t in hls_segment_filename expression besides date/time values when strftime is on. To get fixed width numbers with trailing zeroes, %%0xt format is available where x is the required width.

```
ffmpeg -i sample.mpeg \
  -f hls -hls_time 3 -hls_list_size 5 \
  -hls_flags second_level_segment_index+second_level_segment_size+second_level_segment_duration
  -strftime 1 -strftime_mkdir 1 -hls_segment_filename "segment_%Y%m%d%H%M%S_%04d_%0
```

This will produce segments like this:

```
segment_20170102194334_0003_00122200_0000003000000.ts,
segment_20170102194334_0004_00120072_0000003000000.ts etc.
```

temp_file

Write segment data to filename.tmp and rename to filename only once the segment is complete. A webserver serving up segments can be configured to reject requests to *.tmp to prevent access to in-progress segments before they have been added to the m3u8 playlist. This flag also affects how m3u8 playlist files are created. If this flag is set, all playlist files will be written into a temporary file and renamed after they are complete, similarly as segments are handled. But playlists with "file" protocol and with type ("hls_playlist_type") other than "vod" are always written into a temporary file regardless of this flag. Master playlist files ("master_pl_name"), if any, with "file" protocol, are always written into a temporary file regardless of this flag if "master_pl_publish_rate" value is other than zero.

hls_playlist_type event

Emit "#EXT-X-PLAYLIST-TYPE:EVENT" in the m3u8 header. Forces **hls_list_size** to 0; the playlist can only be appended to.

hls_playlist_type vod

Emit "#EXT-X-PLAYLIST-TYPE:VOD" in the m3u8 header. Forces **hls_list_size** to 0; the playlist must not change.

method

Use the given HTTP method to create the hls files.

```
ffmpeg -re -i in.ts -f hls -method PUT http://example.com/live/out.m3u8
```

This example will upload all the mpegts segment files to the HTTP server using the HTTP PUT method, and update the m3u8 files every "refresh" times using the same method. Note that the HTTP server must support the given method for uploading files.

http_user_agent

Override User-Agent field in HTTP header. Applicable only for HTTP output.

var_stream_map

Map string which specifies how to group the audio, video and subtitle streams into different variant streams. The variant stream groups are separated by space. Expected string format is like this "a:0,v:0 a:1,v:1". Here a:, v:, s: are the keys to specify audio, video and subtitle streams respectively. Allowed values are 0 to 9 (limited just based on practical usage).

When there are two or more variant streams, the output filename pattern must contain the string "%v", this string specifies the position of variant stream index in the output media playlist filenames. The string "%v" may be present in the filename or in the last directory name containing the file. If the string is present in the directory name, then sub-directories are created after expanding the directory name pattern. This enables creation of variant streams in subdirectories.

```
ffmpeg -re -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k -b:a:1 32k \
  -map 0:v -map 0:a -map 0:v -map 0:a -f hls -var_stream_map "v:0,a:0 v:1,a:1" \
  http://example.com/live/out_%v.m3u8
```

This example creates two hls variant streams. The first variant stream will contain video stream of bitrate 1000k and audio stream of bitrate 64k and the second variant stream will contain video stream of bitrate 256k and audio stream of bitrate 32k. Here, two media playlist with file names out_0.m3u8 and out_1.m3u8 will be created. If you want something meaningful text instead of indexes in result names, you may specify names for each or some of the variants as in the following example.

```
ffmpeg -re -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k -b:a:1 32k \
  -map 0:v -map 0:a -map 0:v -map 0:a -f hls -var_stream_map "v:0,a:0,name:my_hd v:1,a:1,name:my_sd" \
  http://example.com/live/out_%v.m3u8
```

This example creates two hls variant streams as in the previous one. But here, the two media playlist with file names out_my_hd.m3u8 and out_my_sd.m3u8 will be created.

```
ffmpeg -re -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k \
  -map 0:v -map 0:a -map 0:v -f hls -var_stream_map "v:0 a:0 v:1" \
  http://example.com/live/out_%v.m3u8
```

This example creates three hls variant streams. The first variant stream will be a video only stream with video bitrate 1000k, the second variant stream will be an audio only stream with bitrate 64k and the third variant stream will be a video only stream with bitrate 256k. Here, three media playlist with file names out_0.m3u8, out_1.m3u8 and out_2.m3u8 will be created.

```
ffmpeg -re -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k -b:a:1 32k \
```

```
-map 0:v -map 0:a -map 0:v -map 0:a -f hls -var_stream_map "v:0,a:0 v:1,a:1" \
http://example.com/live/vs_%v/out.m3u8
```

This example creates the variant streams in subdirectories. Here, the first media playlist is created at http://example.com/live/vs_0/out.m3u8 and the second one at http://example.com/live/vs_1/out.m3u8.

```
ffmpeg -re -i in.ts -b:a:0 32k -b:a:1 64k -b:v:0 1000k -b:v:1 3000k \
-map 0:a -map 0:a -map 0:v -map 0:v -f hls \
-var_stream_map "a:0,agroup:aud_low a:1,agroup:aud_high v:0,agroup:aud_low v:1,agroup:aud_high" \
-master_pl_name master.m3u8 \
http://example.com/live/out_%v.m3u8
```

This example creates two audio only and two video only variant streams. In addition to the #EXT-X-STREAM-INF tag for each variant stream in the master playlist, #EXT-X-MEDIA tag is also added for the two audio only variant streams and they are mapped to the two video only variant streams with audio group names 'aud_low' and 'aud_high'.

By default, a single hls variant containing all the encoded streams is created.

```
ffmpeg -re -i in.ts -b:a:0 32k -b:a:1 64k -b:v:0 1000k \
-map 0:a -map 0:a -map 0:v -f hls \
-var_stream_map "a:0,agroup:aud_low,default:yes a:1,agroup:aud_low v:0,agroup:aud_low" \
-master_pl_name master.m3u8 \
http://example.com/live/out_%v.m3u8
```

This example creates two audio only and one video only variant streams. In addition to the #EXT-X-STREAM-INF tag for each variant stream in the master playlist, #EXT-X-MEDIA tag is also added for the two audio only variant streams and they are mapped to the one video only variant streams with audio group name 'aud_low', and the audio group have default stat is NO or YES.

By default, a single hls variant containing all the encoded streams is created.

```
ffmpeg -re -i in.ts -b:a:0 32k -b:a:1 64k -b:v:0 1000k \
-map 0:a -map 0:a -map 0:v -f hls \
-var_stream_map "a:0,agroup:aud_low,default:yes,language:ENG a:1,agroup:aud_low,language:CHN v:0,a" \
-master_pl_name master.m3u8 \
http://example.com/live/out_%v.m3u8
```

This example creates two audio only and one video only variant streams. In addition to the #EXT-X-STREAM-INF tag for each variant stream in the master playlist, #EXT-X-MEDIA tag is also added for the two audio only variant streams and they are mapped to the one video only variant streams with audio group name 'aud_low', and the audio group have default stat is NO or YES, and one audio have and language is named ENG, the other audio language is named CHN.

By default, a single hls variant containing all the encoded streams is created.

```
ffmpeg -y -i input_with_subtitle.mkv \
-b:v:0 5250k -c:v h264 -pix_fmt yuv420p -profile:v main -level 4.1 \
-b:a:0 256k \
-c:s webvtt -c:a mp2 -ar 48000 -ac 2 -map 0:v -map 0:a:0 -map 0:s:0 \
-f hls -var_stream_map "v:0,a:0,s:0,sgroup:subtitle" \
-master_pl_name master.m3u8 -t 300 -hls_time 10 -hls_init_time 4 -hls_list_size \
10 -master_pl_publish_rate 10 -hls_flags \
delete_segments+discont_start+split_by_time ./tmp/video.m3u8
```

This example adds "#EXT-X-MEDIA" tag with "TYPE=SUBTITLES" in the master playlist with webvtt subtitle group name 'subtitle'. Please make sure the input file has one text subtitle stream at least.

cc_stream_map

Map string which specifies different closed captions groups and their attributes. The closed captions stream groups are separated by space. Expected string format is like this "ccgroup:<group name>,instreamid:<INSTREAM-ID>,language:<language code>". 'ccgroup' and 'instreamid' are mandatory attributes. 'language' is an optional attribute. The closed captions groups configured using this option are mapped to different variant streams by providing the same 'ccgroup' name in the "var_stream_map" string. If "var_stream_map" is not set, then the first available ccgroup in "cc_stream_map" is mapped to the output variant stream. The examples for these two use cases are given below.

```
ffmpeg -re -i in.ts -b:v 1000k -b:a 64k -a53cc 1 -f hls \
-cc_stream_map "ccgroup:cc,instreamid:CC1,language:en" \
-master_pl_name master.m3u8 \
http://example.com/live/out.m3u8
```

This example adds "#EXT-X-MEDIA" tag with "TYPE=CLOSED-CAPTIONS" in the master playlist with group name 'cc', language 'en' (english) and INSTREAM-ID 'CC1'. Also, it adds "CLOSED-CAPTIONS" attribute with group name 'cc' for the output variant stream.

```
ffmpeg -re -i in.ts -b:v:0 1000k -b:v:1 256k -b:a:0 64k -b:a:1 32k \
-a53cc:0 1 -a53cc:1 1 \
-map 0:v -map 0:a -map 0:v -map 0:a -f hls \
-cc_stream_map "ccgroup:cc,instreamid:CC1,language:en ccgroup:cc,instreamid:CC2,language:sp" \
-var_stream_map "v:0,a:0,ccgroup:cc v:1,a:1,ccgroup:cc" \
-master_pl_name master.m3u8 \
http://example.com/live/out_%v.m3u8
```

This example adds two "#EXT-X-MEDIA" tags with "TYPE=CLOSED-CAPTIONS" in the master playlist for the INSTREAM-IDs 'CC1' and 'CC2'. Also, it adds "CLOSED-CAPTIONS" attribute with group name 'cc' for the two output variant streams.

master_pl_name

Create HLS master playlist with the given name.

```
ffmpeg -re -i in.ts -f hls -master_pl_name master.m3u8 http://example.com/live/out.m3u8
```

This example creates HLS master playlist with name master.m3u8 and it is published at <http://example.com/live/>

master_pl_publish_rate

Publish master play list repeatedly every after specified number of segment intervals.

```
ffmpeg -re -i in.ts -f hls -master_pl_name master.m3u8 \
-hls_time 2 -master_pl_publish_rate 30 http://example.com/live/out.m3u8
```

This example creates HLS master playlist with name master.m3u8 and keep publishing it repeatedly every after 30 segments i.e. every after 60s.

http_persistent

Use persistent HTTP connections. Applicable only for HTTP output.

timeout

Set timeout for socket I/O operations. Applicable only for HTTP output.

-ignore_io_errors

Ignore IO errors during open, write and delete. Useful for long-duration runs with network output.

headers

Set custom HTTP headers, can override built in default headers. Applicable only for HTTP output.

ico

ICO file muxer.

Microsoft's icon file format (ICO) has some strict limitations that should be noted:

- ⊕ Size cannot exceed 256 pixels in any dimension
- ⊕ Only BMP and PNG images can be stored
- ⊕ If a BMP image is used, it must be one of the following pixel formats:

BMP Bit Depth	FFmpeg Pixel Format
1bit	pal8
4bit	pal8
8bit	pal8
16bit	rgb555le
24bit	bgr24
32bit	bgra

- ⊕ If a BMP image is used, it must use the BITMAPINFOHEADER DIB header
- ⊕ If a PNG image is used, it must use the rgba pixel format

image2

Image file muxer.

The image file muxer writes video frames to image files.

The output filenames are specified by a pattern, which can be used to produce sequentially numbered series of files. The pattern may contain the string "%d" or "%0Nd", this string specifies the position of the characters representing a numbering in the filenames. If the form "%0Nd" is used, the string representing the number in each filename is 0-padded to *N* digits. The literal character '%' can be specified in the pattern with the string "%%".

If the pattern contains "%d" or "%0Nd", the first filename of the file list specified will contain the number 1, all the following numbers will be sequential.

The pattern may contain a suffix which is used to automatically determine the format of the image files to write.

For example the pattern "img-%03d.bmp" will specify a sequence of filenames of the form *img-001.bmp*, *img-002.bmp*, ..., *img-010.bmp*, etc. The pattern "img%%-%d.jpg" will specify a sequence of filenames of the form *img%-1.jpg*, *img%-2.jpg*, ..., *img%-10.jpg*, etc.

The image muxer supports the .Y.U.V image file format. This format is special in that that each image frame consists of three files, for each of the YUV420P components. To read or write this image file format, specify the name of the '.Y' file. The muxer will automatically open the '.U' and '.V' files as required.

Options

frame_pts

If set to 1, expand the filename with pts from pkt->pts. Default value is 0.

start_number

Start the sequence from the specified number. Default value is 1.

update

If set to 1, the filename will always be interpreted as just a filename, not a pattern, and the corresponding file will be continuously overwritten with new images. Default value is 0.

strftime

If set to 1, expand the filename with date and time information from "strftime()". Default value is 0.

atomic_writing

Write output to a temporary file, which is renamed to target filename once writing is completed. Default is disabled.

protocol_opts *options_list*

Set protocol options as a :-separated list of key=value parameters. Values containing the ":" special character must be escaped.

Examples

The following example shows how to use **ffmpeg** for creating a sequence of files *img-001.jpeg*, *img-002.jpeg*, ..., taking one image every second from the input video:

```
ffmpeg -i in.avi -vsync cfr -r 1 -f image2 'img-%03d.jpeg'
```

Note that with **ffmpeg**, if the format is not specified with the "-f" option and the output filename specifies an image file format, the image2 muxer is automatically selected, so the previous command can be written as:

```
ffmpeg -i in.avi -vsync cfr -r 1 'img-%03d.jpeg'
```

Note also that the pattern must not necessarily contain "%d" or "%0Nd", for example to create a single image file *img.jpeg* from the start of the input video you can employ the command:

```
ffmpeg -i in.avi -f image2 -frames:v 1 img.jpeg
```

The **strftime** option allows you to expand the filename with date and time information. Check the documentation of the "strftime()" function for the syntax.

For example to generate image files from the "strftime()" "%Y-%m-%d_%H-%M-%S" pattern, the following **ffmpeg** command can be used:

```
ffmpeg -f v4l2 -r 1 -i /dev/video0 -f image2 -strftime 1 "%Y-%m-%d_%H-%M-%S.jpg"
```

You can set the file name with current frame's PTS:

```
ffmpeg -f v4l2 -r 1 -i /dev/video0 -copyts -f image2 -frame_pts true %d.jpg"
```

A more complex example is to publish contents of your desktop directly to a WebDAV server every second:

```
ffmpeg -f x11grab -framerate 1 -i :0.0 -q:v 6 -update 1 -protocol_opts method=PUT http://example.com/desktop.jp
```

matroska

Matroska container muxer.

This muxer implements the matroska and webm container specs.

Metadata

The recognized metadata settings in this muxer are:

title Set title name provided to a single track. This gets mapped to the FileDescription element for a stream written as attachment.

language

Specify the language of the track in the Matroska languages form.

The language can be either the 3 letters bibliographic ISO-639-2 (ISO 639-2/B) form (like "fre" for French), or a language code mixed with a country code for specialities in languages (like "fre-ca" for Canadian French).

stereo_mode

Set stereo 3D video layout of two views in a single video track.

The following values are recognized:

mono

video is not stereo

left_right

Both views are arranged side by side, Left-eye view is on the left

bottom_top

Both views are arranged in top-bottom orientation, Left-eye view is at bottom

top_bottom

Both views are arranged in top-bottom orientation, Left-eye view is on top

checkerboard_rl

Each view is arranged in a checkerboard interleaved pattern, Left-eye view being first

checkerboard_lr

Each view is arranged in a checkerboard interleaved pattern, Right-eye view being first

row_interleaved_rl

Each view is constituted by a row based interleaving, Right-eye view is first row

row_interleaved_lr

Each view is constituted by a row based interleaving, Left-eye view is first row

col_interleaved_rl

Both views are arranged in a column based interleaving manner, Right-eye view is first column

col_interleaved_lr

Both views are arranged in a column based interleaving manner, Left-eye view is first column

anaglyph_cyan_red

All frames are in anaglyph format viewable through red-cyan filters

right_left

Both views are arranged side by side, Right-eye view is on the left

anaglyph_green_magenta

All frames are in anaglyph format viewable through green-magenta filters

block_lr

Both eyes laced in one Block, Left-eye view is first

block_rl

Both eyes laced in one Block, Right-eye view is first

For example a 3D WebM clip can be created using the following command line:

```
ffmpeg -i sample_left_right_clip.mpg -an -c:v libvpx -metadata stereo_mode=left_right -y stereo_clip.webm
```

Options

This muxer supports the following options:

reserve_index_space

By default, this muxer writes the index for seeking (called cues in Matroska terms) at the end of the file, because it cannot know in advance how much space to leave for the index at the beginning of the file. However for some use cases -- e.g. streaming where seeking is possible but slow -- it is useful to put the index at the beginning of the file.

If this option is set to a non-zero value, the muxer will reserve a given amount of space in the file header and then try to write the cues there when the muxing finishes. If the reserved space does not suffice, no Cues will be written, the file will be finalized and writing the trailer will return an error. A safe size for most use cases should be about 50kB per hour of video.

Note that cues are only written if the output is seekable and this option will have no effect if it is not.

cues_to_front

If set, the muxer will write the index at the beginning of the file by shifting the main data if necessary. This can be combined with `reserve_index_space` in which case the data is only shifted if the initially reserved space turns out to be insufficient.

This option is ignored if the output is unseekable.

default_mode

This option controls how the `FlagDefault` of the output tracks will be set. It influences which tracks players should play by default. The default mode is **passthrough**.

infer

Every track with disposition default will have the `FlagDefault` set. Additionally, for each type of track (audio, video or subtitle), if no track with disposition default of this type exists, then the first track of this type will be marked as default (if existing). This ensures that the default flag is set in a sensible way even if the input originated from containers that lack the concept of default tracks.

infer_no_subs

This mode is the same as `infer` except that if no subtitle track with disposition default exists, no subtitle track will be marked as default.

passthrough

In this mode the `FlagDefault` is set if and only if the `AV_DISPOSITION_DEFAULT` flag is set in the disposition of the corresponding stream.

flipped_raw_rgb

If set to true, store positive height for raw RGB bitmaps, which indicates bitmap is stored bottom-up. Note that this option does not flip the bitmap which has to be done manually beforehand, e.g. by using the `vflip` filter. Default is *false* and indicates bitmap is stored top down.

md5

MD5 testing format.

This is a variant of the **hash** muxer. Unlike that muxer, it defaults to using the MD5 hash function.

Examples

To compute the MD5 hash of the input converted to raw audio and video, and store it in the file *out.md5*:

```
ffmpeg -i INPUT -f md5 out.md5
```

You can print the MD5 to stdout with the command:

```
ffmpeg -i INPUT -f md5 -
```

See also the **hash** and **framemd5** muxers.

mov, mp4, ismv

MOV/MP4/ISMV (Smooth Streaming) muxer.

The mov/mp4/ismv muxer supports fragmentation. Normally, a MOV/MP4 file has all the metadata about all packets stored in one location (written at the end of the file, it can be moved to the start for better playback by adding *faststart* to the *movflags*, or using the **qt-faststart** tool). A fragmented file consists of a number of fragments, where packets and metadata about these packets are stored together. Writing a fragmented file has the advantage that the file is decodable even if the writing is interrupted (while a normal MOV/MP4 is undecodable if it is not properly finished), and it requires less memory when writing very long files (since writing normal MOV/MP4 files stores info about every single packet in memory until the file is closed). The downside is that it is less compatible with other applications.

Options

Fragmentation is enabled by setting one of the AVOptions that define how to cut the file into fragments:

-moov_size *bytes*

Reserves space for the moov atom at the beginning of the file instead of placing the moov atom at the end. If the space reserved is insufficient, muxing will fail.

-movflags frag_keyframe

Start a new fragment at each video keyframe.

-frag_duration *duration*

Create fragments that are *duration* microseconds long.

-frag_size *size*

Create fragments that contain up to *size* bytes of payload data.

-movflags frag_custom

Allow the caller to manually choose when to cut fragments, by calling "av_write_frame(ctx, NULL)" to write a fragment with the packets written so far. (This is only useful with other applications integrating libavformat, not from **ffmpeg**.)

-min_frag_duration *duration*

Don't create fragments that are shorter than *duration* microseconds long.

If more than one condition is specified, fragments are cut when one of the specified conditions is fulfilled. The exception to this is "-min_frag_duration", which has to be fulfilled for any of the other conditions to apply.

Additionally, the way the output file is written can be adjusted through a few other options:

-movflags empty_moov

Write an initial moov atom directly at the start of the file, without describing any samples in it. Generally, an mdat/moov pair is written at the start of the file, as a normal MOV/MP4 file, containing only a short portion of the file. With this option set, there is no initial mdat atom, and the moov atom only describes the tracks but has a zero duration.

This option is implicitly set when writing ismv (Smooth Streaming) files.

-movflags separate_moof

Write a separate moof (movie fragment) atom for each track. Normally, packets for all tracks are written in a moof atom (which is slightly more efficient), but with this option set, the muxer writes one moof/mdat pair for each track, making it easier to separate tracks.

This option is implicitly set when writing ismv (Smooth Streaming) files.

-movflags skip_sidx

Skip writing of sidx atom. When bitrate overhead due to sidx atom is high, this option could be used for cases where sidx atom is not mandatory. When global_sidx flag is enabled, this option will be ignored.

-movflags faststart

Run a second pass moving the index (moov atom) to the beginning of the file. This operation can take a while, and will not work in various situations such as fragmented output, thus it is not enabled by default.

-movflags rtphint

Add RTP hinting tracks to the output file.

-movflags disable_chpl

Disable Nero chapter markers (chpl atom). Normally, both Nero chapters and a QuickTime chapter track are written to the file. With this option set, only the QuickTime chapter track will be written. Nero chapters can cause failures when the file is reprocessed with certain tagging programs, like mp3Tag 2.61a and iTunes 11.3, most likely other versions are affected as well.

-movflags omit_tfhd_offset

Do not write any absolute `base_data_offset` in `tfhd` atoms. This avoids tying fragments to absolute byte positions in the file/streams.

-movflags default_base_moof

Similarly to the `omit_tfhd_offset`, this flag avoids writing the absolute `base_data_offset` field in `tfhd` atoms, but does so by using the new `default-base-is-moof` flag instead. This flag is new from 14496-12:2012. This may make the fragments easier to parse in certain circumstances (avoiding basing track fragment location calculations on the implicit end of the previous track fragment).

-write_tmcd

Specify "on" to force writing a timecode track, "off" to disable it and "auto" to write a timecode track only for mov and mp4 output (default).

-movflags negative_cts_offsets

Enables utilization of version 1 of the CTTS box, in which the CTS offsets can be negative. This enables the initial sample to have DTS/CTS of zero, and reduces the need for edit lists for some cases such as video tracks with B-frames. Additionally, eases conformance with the DASH-IF interoperability guidelines.

This option is implicitly set when writing ismv (Smooth Streaming) files.

-write_btrt *bool*

Force or disable writing bitrate box inside `stsd` box of a track. The box contains decoding buffer size (in bytes), maximum bitrate and average bitrate for the track. The box will be skipped if none of these values can be computed. Default is "-1" or "auto", which will write the box only in MP4 mode.

-write_prft

Write producer time reference box (PRFT) with a specified time source for the NTP field in the PRFT box. Set value as **wallclock** to specify timesource as wallclock time and **pts** to specify timesource as input packets' PTS values.

Setting value to **pts** is applicable only for a live encoding use case, where PTS values are set as as

wallclock time at the source. For example, an encoding use case with decklink capture source where **video_pts** and **audio_pts** are set to **abs_wallclock**.

-empty_hdlr_name *bool*

Enable to skip writing the name inside a "hdlr" box. Default is "false".

-movie_timescale *scale*

Set the timescale written in the movie header box ("mvhd"). Range is 1 to INT_MAX. Default is 1000.

-video_track_timescale *scale*

Set the timescale used for video tracks. Range is 0 to INT_MAX. If set to 0, the timescale is automatically set based on the native stream time base. Default is 0.

Example

Smooth Streaming content can be pushed in real time to a publishing point on IIS with this muxer.

Example:

```
ffmpeg -re <<normal input/transcoding options>> -movflags isml+frag_keyframe -f ismv http://server/publishing
```

mp3

The MP3 muxer writes a raw MP3 stream with the following optional features:

- ⊕ An ID3v2 metadata header at the beginning (enabled by default). Versions 2.3 and 2.4 are supported, the "id3v2_version" private option controls which one is used (3 or 4). Setting "id3v2_version" to 0 disables the ID3v2 header completely.

The muxer supports writing attached pictures (APIC frames) to the ID3v2 header. The pictures are supplied to the muxer in form of a video stream with a single packet. There can be any number of those streams, each will correspond to a single APIC frame. The stream metadata tags *title* and *comment* map to APIC *description* and *picture type* respectively. See <http://id3.org/id3v2.4.0-frames> for allowed picture types.

Note that the APIC frames must be written at the beginning, so the muxer will buffer the audio frames until it gets all the pictures. It is therefore advised to provide the pictures as soon as possible to avoid excessive buffering.

- ⊕ A Xing/LAME frame right after the ID3v2 header (if present). It is enabled by default, but will be written only if the output is seekable. The "write_xing" private option can be used to disable it.

The frame contains various information that may be useful to the decoder, like the audio duration or encoder delay.

- ⊕ A legacy ID3v1 tag at the end of the file (disabled by default). It may be enabled with the "write_id3v1" private option, but as its capabilities are very limited, its usage is not recommended.

Examples:

Write an mp3 with an ID3v2.3 header and an ID3v1 footer:

```
ffmpeg -i INPUT -id3v2_version 3 -write_id3v1 1 out.mp3
```

To attach a picture to an mp3 file select both the audio and the picture stream with "map":

```
ffmpeg -i input.mp3 -i cover.png -c copy -map 0 -map 1
-metadatas:s:v title="Album cover" -metadatas:s:v comment="Cover (Front)" out.mp3
```

Write a "clean" MP3 without any extra features:

```
ffmpeg -i input.wav -write_xing 0 -id3v2_version 0 out.mp3
```

mpepts

MPEG transport stream muxer.

This muxer implements ISO 13818-1 and part of ETSI EN 300 468.

The recognized metadata settings in mpepts muxer are "service_provider" and "service_name". If they are not set the default for "service_provider" is **FFmpeg** and the default for "service_name" is **Service01**.

Options

The muxer options are:

mpepts_transport_stream_id *integer*

Set the **transport_stream_id**. This identifies a transponder in DVB. Default is 0x0001.

mpepts_original_network_id *integer*

Set the **original_network_id**. This is unique identifier of a network in DVB. Its main use is in the unique identification of a service through the path **Original_Network_ID, Transport_Stream_ID**.

Default is 0x0001.

mpegs_service_id *integer*

Set the **service_id**, also known as program in DVB. Default is 0x0001.

mpegs_service_type *integer*

Set the program **service_type**. Default is "digital_tv". Accepts the following options:

hex_value

Any hexadecimal value between 0x01 and 0xff as defined in ETSI 300 468.

digital_tv

Digital TV service.

digital_radio

Digital Radio service.

teletext

Teletext service.

advanced_codec_digital_radio

Advanced Codec Digital Radio service.

mpeg2_digital_hdtv

MPEG2 Digital HDTV service.

advanced_codec_digital_sdtv

Advanced Codec Digital SDTV service.

advanced_codec_digital_hdtv

Advanced Codec Digital HDTV service.

mpegs_pmt_start_pid *integer*

Set the first PID for PMTs. Default is 0x1000, minimum is 0x0020, maximum is 0x1ffa. This option has no effect in m2ts mode where the PMT PID is fixed 0x0100.

mpegs_start_pid *integer*

Set the first PID for elementary streams. Default is 0x0100, minimum is 0x0020, maximum is 0x1ffa. This option has no effect in m2ts mode where the elementary stream PIDs are fixed.

mpegts_m2ts_mode *boolean*

Enable m2ts mode if set to 1. Default value is "-1" which disables m2ts mode.

muxrate *integer*

Set a constant muxrate. Default is VBR.

pes_payload_size *integer*

Set minimum PES packet payload in bytes. Default is 2930.

mpegts_flags *flags*

Set mpegts flags. Accepts the following options:

resend_headers

Reemit PAT/PMT before writing the next packet.

latm

Use LATM packetization for AAC.

pat_pmt_at_frames

Reemit PAT and PMT at each video frame.

system_b

Conform to System B (DVB) instead of System A (ATSC).

initial_discontinuity

Mark the initial packet of each stream as discontinuity.

nit Emit NIT table.**omit_rai**

Disable writing of random access indicator.

mpegts_copyts *boolean*

Preserve original timestamps, if value is set to 1. Default value is "-1", which results in shifting timestamps so that they start from 0.

omit_video_pes_length *boolean*

Omit the PES packet length for video packets. Default is 1 (true).

per_period *integer*

Override the default PCR retransmission time in milliseconds. Default is "-1" which means that the PCR interval will be determined automatically: 20 ms is used for CBR streams, the highest multiple of the frame duration which is less than 100 ms is used for VBR streams.

pat_period *duration*

Maximum time in seconds between PAT/PMT tables. Default is 0.1.

sdt_period *duration*

Maximum time in seconds between SDT tables. Default is 0.5.

nit_period *duration*

Maximum time in seconds between NIT tables. Default is 0.5.

tables_version *integer*

Set PAT, PMT, SDT and NIT version (default 0, valid values are from 0 to 31, inclusively). This option allows updating stream structure so that standard consumer may detect the change. To do so, reopen output "AVFormatContext" (in case of API usage) or restart **ffmpeg** instance, cyclically changing **tables_version** value:

```
ffmpeg -i source1.ts -codec copy -f mpegts -tables_version 0 udp://1.1.1.1:1111
ffmpeg -i source2.ts -codec copy -f mpegts -tables_version 1 udp://1.1.1.1:1111
...
ffmpeg -i source3.ts -codec copy -f mpegts -tables_version 31 udp://1.1.1.1:1111
ffmpeg -i source1.ts -codec copy -f mpegts -tables_version 0 udp://1.1.1.1:1111
ffmpeg -i source2.ts -codec copy -f mpegts -tables_version 1 udp://1.1.1.1:1111
...
```

Example

```
ffmpeg -i file.mpg -c copy \
  -mpegts_original_network_id 0x1122 \
  -mpegts_transport_stream_id 0x3344 \
  -mpegts_service_id 0x5566 \
  -mpegts_pmt_start_pid 0x1500 \
  -mpegts_start_pid 0x150 \
  -metadata service_provider="Some provider" \
  -metadata service_name="Some Channel" \
  out.ts
```

mxf, **mx**f_d10, **mx**f_opatom

MXF muxer.

Options

The muxer options are:

store_user_comments *bool*

Set if user comments should be stored if available or never. IRT D-10 does not allow user comments. The default is thus to write them for mxf and mxf_opatom but not for mxf_d10

null

Null muxer.

This muxer does not generate any output file, it is mainly useful for testing or benchmarking purposes.

For example to benchmark decoding with **ffmpeg** you can use the command:

```
ffmpeg -benchmark -i INPUT -f null out.null
```

Note that the above command does not read or write the *out.null* file, but specifying the output file is required by the **ffmpeg** syntax.

Alternatively you can write the command as:

```
ffmpeg -benchmark -i INPUT -f null -
```

nut

-syncpoints *flags*

Change the syncpoint usage in nut:

default **use the normal low-overhead seeking aids.**

none **do not use the syncpoints at all, reducing the overhead but making the stream non-seekable;**

Use of this option is not recommended, as the resulting files are very damage sensitive and seeking is not possible. Also in general the overhead from syncpoints is negligible. Note, **-C<write_index> 0** can be used to disable all growing data tables, allowing to mux endless streams with limited memory and without these disadvantages.

timestamped **extend the syncpoint with a wallclock field.**

The *none* and *timestamped* flags are experimental.

-write_index *bool*

Write index at the end, the default is to write an index.

```
ffmpeg -i INPUT -f_strict experimental -syncpoints none - | processor
```

ogg

Ogg container muxer.

-page_duration *duration*

Preferred page duration, in microseconds. The muxer will attempt to create pages that are approximately *duration* microseconds long. This allows the user to compromise between seek granularity and container overhead. The default is 1 second. A value of 0 will fill all segments, making pages as large as possible. A value of 1 will effectively use 1 packet-per-page in most situations, giving a small seek granularity at the cost of additional container overhead.

-serial_offset *value*

Serial value from which to set the streams serial number. Setting it to different and sufficiently large values ensures that the produced ogg files can be safely chained.

raw muxers

Raw muxers accept a single stream matching the designated codec. They do not store timestamps or metadata. The recognized extension is the same as the muxer name unless indicated otherwise.

ac3

Dolby Digital, also known as AC-3, audio.

adx

CRI Middleware ADX audio.

This muxer will write out the total sample count near the start of the first packet when the output is seekable and the count can be stored in 32 bits.

aptx

aptX (Audio Processing Technology for Bluetooth) audio.

aptx_hd

aptX HD (Audio Processing Technology for Bluetooth) audio.

Extensions: aptxhd

avs2

AVS2-P2/IEEE1857.4 video.

Extensions: avs, avs2

cavsvideo

Chinese AVS (Audio Video Standard) video.

Extensions: cavs

codec2raw

Codec 2 audio.

No extension is registered so format name has to be supplied e.g. with the ffmpeg CLI tool "-f codec2raw".

data

Data muxer accepts a single stream with any codec of any type. The input stream has to be selected using the "-map" option with the ffmpeg CLI tool.

No extension is registered so format name has to be supplied e.g. with the ffmpeg CLI tool "-f data".

dirac

BBC Dirac video. The Dirac Pro codec is a subset and is standardized as SMPTE VC-2.

Extensions: drc, vc2

dnxhd

Avid DNxHD video. It is standardized as SMPTE VC-3. Accepts DNxHR streams.

Extensions: dnxhd, dnxhr

dts

DTS Coherent Acoustics (DCA) audio.

eac3

Dolby Digital Plus, also known as Enhanced AC-3, audio.

g722

ITU-T G.722 audio.

g723_1

ITU-T G.723.1 audio.

Extensions: tco, rco

g726

ITU-T G.726 big-endian ("left-justified") audio.

No extension is registered so format name has to be supplied e.g. with the ffmpeg CLI tool "-f g726".

g726le

ITU-T G.726 little-endian ("right-justified") audio.

No extension is registered so format name has to be supplied e.g. with the ffmpeg CLI tool "-f g726le".

gsm

Global System for Mobile Communications audio.

h261

ITU-T H.261 video.

h263

ITU-T H.263 / H.263-1996, H.263+ / H.263-1998 / H.263 version 2 video.

h264

ITU-T H.264 / MPEG-4 Part 10 AVC video. Bitstream shall be converted to Annex B syntax if it's in length-prefixed mode.

Extensions: h264, 264

hevc

ITU-T H.265 / MPEG-H Part 2 HEVC video. Bitstream shall be converted to Annex B syntax if it's in length-prefixed mode.

Extensions: hevc, h265, 265

m4v

MPEG-4 Part 2 video.

mjpeg

Motion JPEG video.

Extensions: mjpg, mjpeg

mlp

Meridian Lossless Packing, also known as Packed PCM, audio.

mp2

MPEG-1 Audio Layer II audio.

Extensions: mp2, m2a, mpa

mpeg1video

MPEG-1 Part 2 video.

Extensions: mpg, mpeg, m1v

mpeg2video

ITU-T H.262 / MPEG-2 Part 2 video.

Extensions: m2v

obu

AV1 low overhead Open Bitstream Units muxer. Temporal delimiter OBUs will be inserted in all temporal units of the stream.

rawvideo

Raw uncompressed video.

Extensions: yuv, rgb

sbc

Bluetooth SIG low-complexity subband codec audio.

Extensions: sbc, msbc

truehd

Dolby TrueHD audio.

Extensions: thd

vc1

SMPTE 421M / VC-1 video.

segment, stream_segment, ssegment

Basic stream segmenter.

This muxer outputs streams to a number of separate files of nearly fixed duration. Output filename pattern can be set in a fashion similar to **image2**, or by using a "strftime" template if the **strftime** option is enabled.

"stream_segment" is a variant of the muxer used to write to streaming output formats, i.e. which do not require global headers, and is recommended for outputting e.g. to MPEG transport stream segments. "ssegment" is a shorter alias for "stream_segment".

Every segment starts with a keyframe of the selected reference stream, which is set through the **reference_stream** option.

Note that if you want accurate splitting for a video file, you need to make the input key frames correspond to the exact splitting times expected by the segmenter, or the segment muxer will start the new segment with the key frame found next after the specified start time.

The segment muxer works best with a single constant frame rate video.

Optionally it can generate a list of the created segments, by setting the option *segment_list*. The list type is specified by the *segment_list_type* option. The entry filenames in the segment list are set by default to the basename of the corresponding segment files.

See also the **hls** muxer, which provides a more specific implementation for HLS segmentation.

Options

The segment muxer supports the following options:

increment_tc *I/O*

if set to 1, increment timecode between each segment. If this is selected, the input need to have a timecode in the first video stream. Default value is 0.

reference_stream *specifier*

Set the reference stream, as specified by the string *specifier*. If *specifier* is set to "auto", the reference is chosen automatically. Otherwise it must be a stream specifier (see the "Stream specifiers" chapter in the ffmpeg manual) which specifies the reference stream. The default value is "auto".

segment_format *format*

Override the inner container format, by default it is guessed by the filename extension.

segment_format_options *options_list*

Set output format options using a :-separated list of key=value parameters. Values containing the ":" special character must be escaped.

segment_list *name*

Generate also a listfile named *name*. If not specified no listfile is generated.

segment_list_flags *flags*

Set flags affecting the segment list generation.

It currently supports the following flags:

cache

Allow caching (only affects M3U8 list files).

live Allow live-friendly file generation.

segment_list_size *size*

Update the list file so that it contains at most *size* segments. If 0 the list file will contain all the segments. Default value is 0.

segment_list_entry_prefix *prefix*

Prepend *prefix* to each entry. Useful to generate absolute paths. By default no prefix is applied.

segment_list_type *type*

Select the listing format.

The following values are recognized:

flat Generate a flat list for the created segments, one segment per line.

csv, ext

Generate a list for the created segments, one segment per line, each line matching the format (comma-separated values):

<segment_filename>,<segment_start_time>,<segment_end_time>

segment_filename is the name of the output file generated by the muxer according to the

provided pattern. CSV escaping (according to RFC4180) is applied if required.

segment_start_time and *segment_end_time* specify the segment start and end time expressed in seconds.

A list file with the suffix ".csv" or ".ext" will auto-select this format.

ext is deprecated in favor of **csv**.

ffconcat

Generate an ffconcat file for the created segments. The resulting file can be read using the FFmpeg **concat** demuxer.

A list file with the suffix ".ffcat" or ".ffconcat" will auto-select this format.

m3u8

Generate an extended M3U8 file, version 3, compliant with <http://tools.ietf.org/id/draft-pantos-http-live-streaming>.

A list file with the suffix ".m3u8" will auto-select this format.

If not specified the type is guessed from the list file name suffix.

segment_time *time*

Set segment duration to *time*, the value must be a duration specification. Default value is "2". See also the **segment_times** option.

Note that splitting may not be accurate, unless you force the reference stream key-frames at the given time. See the introductory notice and the examples below.

min_seg_duration *time*

Set minimum segment duration to *time*, the value must be a duration specification. This prevents the muxer ending segments at a duration below this value. Only effective with "segment_time". Default value is "0".

segment_atclocktime *1/0*

If set to "1" split at regular clock time intervals starting from 00:00 o'clock. The *time* value specified in **segment_time** is used for setting the length of the splitting interval.

For example with **segment_time** set to "900" this makes it possible to create files at 12:00 o'clock,

12:15, 12:30, etc.

Default value is "0".

segment_clocktime_offset *duration*

Delay the segment splitting times with the specified duration when using **segment_atclocktime**.

For example with **segment_time** set to "900" and **segment_clocktime_offset** set to "300" this makes it possible to create files at 12:05, 12:20, 12:35, etc.

Default value is "0".

segment_clocktime_wrap_duration *duration*

Force the segmenter to only start a new segment if a packet reaches the muxer within the specified duration after the segmenting clock time. This way you can make the segmenter more resilient to backward local time jumps, such as leap seconds or transition to standard time from daylight savings time.

Default is the maximum possible duration which means starting a new segment regardless of the elapsed time since the last clock time.

segment_time_delta *delta*

Specify the accuracy time when selecting the start time for a segment, expressed as a duration specification. Default value is "0".

When delta is specified a key-frame will start a new segment if its PTS satisfies the relation:

$$\text{PTS} \geq \text{start_time} - \text{time_delta}$$

This option is useful when splitting video content, which is always split at GOP boundaries, in case a key frame is found just before the specified split time.

In particular may be used in combination with the *ffmpeg* option *force_key_frames*. The key frame times specified by *force_key_frames* may not be set accurately because of rounding issues, with the consequence that a key frame time may result set just before the specified time. For constant frame rate videos a value of $1/(2*\text{frame_rate})$ should address the worst case mismatch between the specified time and the time set by *force_key_frames*.

segment_times *times*

Specify a list of split points. *times* contains a list of comma separated duration specifications, in

increasing order. See also the **segment_time** option.

segment_frames *frames*

Specify a list of split video frame numbers. *frames* contains a list of comma separated integer numbers, in increasing order.

This option specifies to start a new segment whenever a reference stream key frame is found and the sequential number (starting from 0) of the frame is greater or equal to the next value in the list.

segment_wrap *limit*

Wrap around segment index once it reaches *limit*.

segment_start_number *number*

Set the sequence number of the first segment. Defaults to 0.

strftime *1/0*

Use the "strftime" function to define the name of the new segments to write. If this is selected, the output segment name must contain a "strftime" function template. Default value is 0.

break_non_keyframes *1/0*

If enabled, allow segments to start on frames other than keyframes. This improves behavior on some players when the time between keyframes is inconsistent, but may make things worse on others, and can cause some oddities during seeking. Defaults to 0.

reset_timestamps *1/0*

Reset timestamps at the beginning of each segment, so that each segment will start with near-zero timestamps. It is meant to ease the playback of the generated segments. May not work with some combinations of muxers/codecs. It is set to 0 by default.

initial_offset *offset*

Specify timestamp offset to apply to the output packet timestamps. The argument must be a time duration specification, and defaults to 0.

write_empty_segments *1/0*

If enabled, write an empty segment if there are no packets during the period a segment would usually span. Otherwise, the segment will be filled with the next packet written. Defaults to 0.

Make sure to require a closed GOP when encoding and to set the GOP size to fit your segment time constraint.

Examples

- ⊕ Remux the content of file *in.mkv* to a list of segments *out-000.nut*, *out-001.nut*, etc., and write the list of generated segments to *out.list*:

```
ffmpeg -i in.mkv -codec hevc -flags +cgop -g 60 -map 0 -f segment -segment_list out.list out%03d.nut
```

- ⊕ Segment input and set output format options for the output segments:

```
ffmpeg -i in.mkv -f segment -segment_time 10 -segment_format_options movflags=+faststart out%03d.mp4
```

- ⊕ Segment the input file according to the split points specified by the *segment_times* option:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.csv -segment_times 1,2,3,5,8,13,21 out%03d.nut
```

- ⊕ Use the **ffmpeg force_key_frames** option to force key frames in the input at the specified location, together with the segment option **segment_time_delta** to account for possible roundings operated when setting key frame times.

```
ffmpeg -i in.mkv -force_key_frames 1,2,3,5,8,13,21 -codec:v mpeg4 -codec:a pcm_s16le -map 0 \
-f segment -segment_list out.csv -segment_times 1,2,3,5,8,13,21 -segment_time_delta 0.05 out%03d.nut
```

In order to force key frames on the input file, transcoding is required.

- ⊕ Segment the input file by splitting the input file according to the frame numbers sequence specified with the **segment_frames** option:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.csv -segment_frames 100,200,300,500,800
```

- ⊕ Convert the *in.mkv* to TS segments using the "libx264" and "aac" encoders:

```
ffmpeg -i in.mkv -map 0 -codec:v libx264 -codec:a aac -f ssegment -segment_list out.list out%03d.ts
```

- ⊕ Segment the input file, and create an M3U8 live playlist (can be used as live HLS source):

```
ffmpeg -re -i in.mkv -codec copy -map 0 -f segment -segment_list playlist.m3u8 \
-segment_list_flags +live -segment_time 10 out%03d.mkv
```

smoothstreaming

Smooth Streaming muxer generates a set of files (Manifest, chunks) suitable for serving with

conventional web server.

window_size

Specify the number of fragments kept in the manifest. Default 0 (keep all).

extra_window_size

Specify the number of fragments kept outside of the manifest before removing from disk. Default 5.

lookahead_count

Specify the number of lookahead fragments. Default 2.

min_frag_duration

Specify the minimum fragment duration (in microseconds). Default 5000000.

remove_at_exit

Specify whether to remove all fragments when finished. Default 0 (do not remove).

streamhash

Per stream hash testing format.

This muxer computes and prints a cryptographic hash of all the input frames, on a per-stream basis. This can be used for equality checks without having to do a complete binary comparison.

By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash, but the output of explicit conversions to other codecs can also be used. Timestamps are ignored. It uses the SHA-256 cryptographic hash function by default, but supports several other algorithms.

The output of the muxer consists of one line per stream of the form: *streamindex,streamtype,algo=hash*, where *streamindex* is the index of the mapped stream, *streamtype* is a single character indicating the type of stream, *algo* is a short string representing the hash function used, and *hash* is a hexadecimal number representing the computed hash.

hash algorithm

Use the cryptographic hash function specified by the string *algorithm*. Supported values include "MD5", "murmur3", "RIPEMD128", "RIPEMD160", "RIPEMD256", "RIPEMD320", "SHA160", "SHA224", "SHA256" (default), "SHA512/224", "SHA512/256", "SHA384", "SHA512", "CRC32" and "adler32".

Examples

To compute the SHA-256 hash of the input converted to raw audio and video, and store it in the file *out.sha256*:

```
ffmpeg -i INPUT -f streamhash out.sha256
```

To print an MD5 hash to stdout use the command:

```
ffmpeg -i INPUT -f streamhash -hash md5 -
```

See also the **hash** and **framehash** muxers.

tee

The tee muxer can be used to write the same data to several outputs, such as files or streams. It can be used, for example, to stream a video over a network and save it to disk at the same time.

It is different from specifying several outputs to the **ffmpeg** command-line tool. With the tee muxer, the audio and video data will be encoded only once. With conventional multiple outputs, multiple encoding operations in parallel are initiated, which can be a very expensive process. The tee muxer is not useful when using the libavformat API directly because it is then possible to feed the same packets to several muxers directly.

Since the tee muxer does not represent any particular output format, ffmpeg cannot auto-select output streams. So all streams intended for output must be specified using "-map". See the examples below.

Some encoders may need different options depending on the output format; the auto-detection of this can not work with the tee muxer, so they need to be explicitly specified. The main example is the **global_header** flag.

The slave outputs are specified in the file name given to the muxer, separated by '|'. If any of the slave name contains the '|' separator, leading or trailing spaces or any special character, those must be escaped (see the "**Quoting and escaping**" section in the **ffmpeg-utils(1)** manual).

Options

use_fifo *bool*

If set to 1, slave outputs will be processed in separate threads using the **fifo** muxer. This allows to compensate for different speed/latency/reliability of outputs and setup transparent recovery. By default this feature is turned off.

fifo_options

Options to pass to fifo pseudo-muxer instances. See **fifo**.

Muxer options can be specified for each slave by prepending them as a list of *key=value* pairs separated by ':', between square brackets. If the options values contain a special character or the ':' separator, they must be escaped; note that this is a second level escaping.

The following special options are also recognized:

f Specify the format name. Required if it cannot be guessed from the output URL.

bsfs[*/spec*]

Specify a list of bitstream filters to apply to the specified output.

It is possible to specify to which streams a given bitstream filter applies, by appending a stream specifier to the option separated by "/". *spec* must be a stream specifier (see **Format stream specifiers**).

If the stream specifier is not specified, the bitstream filters will be applied to all streams in the output. This will cause that output operation to fail if the output contains streams to which the bitstream filter cannot be applied e.g. "h264_mp4toannexb" being applied to an output containing an audio stream.

Options for a bitstream filter must be specified in the form of "opt=value".

Several bitstream filters can be specified, separated by ",".

use_fifo *bool*

This allows to override tee muxer use_fifo option for individual slave muxer.

fifo_options

This allows to override tee muxer fifo_options for individual slave muxer. See **fifo**.

select

Select the streams that should be mapped to the slave output, specified by a stream specifier. If not specified, this defaults to all the mapped streams. This will cause that output operation to fail if the output format does not accept all mapped streams.

You may use multiple stream specifiers separated by commas (",") e.g.: "a:0,v"

onfail

Specify behaviour on output failure. This can be set to either "abort" (which is default) or "ignore". "abort" will cause whole process to fail in case of failure on this slave output. "ignore" will ignore failure on this output, so other outputs will continue without being affected.

Examples

- ⊕ Encode something and both archive it in a WebM file and stream it as MPEG-TS over UDP:

```
ffmpeg -i ... -c:v libx264 -c:a mp2 -f tee -map 0:v -map 0:a
"archive-20121107.mkv|[f=mpegts]udp://10.0.1.255:1234/"
```

- ⊕ As above, but continue streaming even if output to local file fails (for example local drive fills up):

```
ffmpeg -i ... -c:v libx264 -c:a mp2 -f tee -map 0:v -map 0:a
"[onfail=ignore]archive-20121107.mkv|[f=mpegts]udp://10.0.1.255:1234/"
```

- ⊕ Use **ffmpeg** to encode the input, and send the output to three different destinations. The "dump_extra" bitstream filter is used to add extradata information to all the output video keyframes packets, as requested by the MPEG-TS format. The select option is applied to *out.aac* in order to make it contain only audio packets.

```
ffmpeg -i ... -map 0 -flags +global_header -c:v libx264 -c:a aac
-f tee "[bsfs/v=dump_extra=freq=keyframe]out.ts|[movflags=+faststart]out.mp4|[select=a]out.aac"
```

- ⊕ As above, but select only stream "a:1" for the audio output. Note that a second level escaping must be performed, as ":" is a special character used to separate options.

```
ffmpeg -i ... -map 0 -flags +global_header -c:v libx264 -c:a aac
-f tee "[bsfs/v=dump_extra=freq=keyframe]out.ts|[movflags=+faststart]out.mp4|[select='a:1']out.aac"
```

webm_chunk

WebM Live Chunk Muxer.

This muxer writes out WebM headers and chunks as separate files which can be consumed by clients that support WebM Live streams via DASH.

Options

This muxer supports the following options:

chunk_start_index

Index of the first chunk (defaults to 0).

header

Filename of the header where the initialization data will be written.

audio_chunk_duration

Duration of each audio chunk in milliseconds (defaults to 5000).

Example

```
ffmpeg -f v4l2 -i /dev/video0 \  
-f alsactl -i hw:0 \  
-map 0:0 \  
-c:v libvpx-vp9 \  
-s 640x360 -keyint_min 30 -g 30 \  
-f webm_chunk \  
-header webm_live_video_360.hdr \  
-chunk_start_index 1 \  
webm_live_video_360_%d.chk \  
-map 1:0 \  
-c:a libvorbis \  
-b:a 128k \  
-f webm_chunk \  
-header webm_live_audio_128.hdr \  
-chunk_start_index 1 \  
-audio_chunk_duration 1000 \  
webm_live_audio_128_%d.chk
```

webm_dash_manifest

WebM DASH Manifest muxer.

This muxer implements the WebM DASH Manifest specification to generate the DASH manifest XML. It also supports manifest generation for DASH live streams.

For more information see:

- ⊕ [WebM DASH Specification](#):

<<https://sites.google.com/a/webmproject.org/wiki/adaptive-streaming/webm-dash-specification>>

⊕ ISO DASH Specification:

<http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip>

Options

This muxer supports the following options:

adaptation_sets

This option has the following syntax: "id=x,streams=a,b,c id=y,streams=d,e" where x and y are the unique identifiers of the adaptation sets and a,b,c,d and e are the indices of the corresponding audio and video streams. Any number of adaptation sets can be added using this option.

live Set this to 1 to create a live stream DASH Manifest. Default: 0.

chunk_start_index

Start index of the first chunk. This will go in the **startNumber** attribute of the **SegmentTemplate** element in the manifest. Default: 0.

chunk_duration_ms

Duration of each chunk in milliseconds. This will go in the **duration** attribute of the **SegmentTemplate** element in the manifest. Default: 1000.

utc_timing_url

URL of the page that will return the UTC timestamp in ISO format. This will go in the **value** attribute of the **UTCTiming** element in the manifest. Default: None.

time_shift_buffer_depth

Smallest time (in seconds) shifting buffer for which any Representation is guaranteed to be available. This will go in the **timeShiftBufferDepth** attribute of the **MPD** element. Default: 60.

minimum_update_period

Minimum update period (in seconds) of the manifest. This will go in the **minimumUpdatePeriod** attribute of the **MPD** element. Default: 0.

Example

```
ffmpeg -f webm_dash_manifest -i video1.webm \
-f webm_dash_manifest -i video2.webm \
```



```

-f webm_dash_manifest -i audio1.webm \
-f webm_dash_manifest -i audio2.webm \
-map 0 -map 1 -map 2 -map 3 \
-c copy \
-f webm_dash_manifest \
-adaptation_sets "id=0,streams=0,1 id=1,streams=2,3" \
manifest.xml

```

METADATA

FFmpeg is able to dump metadata from media files into a simple UTF-8-encoded INI-like text file and then load it back using the metadata muxer/demuxer.

The file format is as follows:

1. A file consists of a header and a number of metadata tags divided into sections, each on its own line.
2. The header is a **;FFMETADATA** string, followed by a version number (now 1).
3. Metadata tags are of the form **key=value**
4. Immediately after header follows global metadata
5. After global metadata there may be sections with per-stream/per-chapter metadata.
6. A section starts with the section name in uppercase (i.e. **STREAM** or **CHAPTER**) in brackets ([,]) and ends with next section or end of file.
7. At the beginning of a chapter section there may be an optional timebase to be used for start/end values. It must be in form **TIMEBASE=num/den**, where *num* and *den* are integers. If the timebase is missing then start/end times are assumed to be in nanoseconds.

Next a chapter section must contain chapter start and end times in form **START=num, END=num**, where *num* is a positive integer.

8. Empty lines and lines starting with **;** or **#** are ignored.
9. Metadata keys or values containing special characters (**=**, **;**, **#**, **** and a newline) must be escaped with a backslash ****.

10. Note that whitespace in metadata (e.g. **foo = bar**) is considered to be a part of the tag (in the example above key is **foo** , value is **bar**).

A ffmetadata file might look like this:

```
;FFMETADATA1
title=bike\shed
;this is a comment
artist=FFmpeg troll team

[CHAPTER]
TIMEBASE=1/1000
START=0
#chapter ends at 0:01:00
END=60000
title=chapter \#1
[STREAM]
title=multi\
line
```

By using the ffmetadata muxer and demuxer it is possible to extract metadata from an input file to an ffmetadata file, and then transcode the file into an output file with the edited ffmetadata file.

Extracting an ffmetadata file with *ffmpeg* goes as follows:

```
ffmpeg -i INPUT -f ffmetadata FFMETADATAFILE
```

Reinserting edited metadata information from the FFMETADATAFILE file can be done as:

```
ffmpeg -i INPUT -i FFMETADATAFILE -map_metadata 1 -codec copy OUTPUT
```

PROTOCOL OPTIONS

The libavformat library provides some generic global options, which can be set on all the protocols. In addition each protocol may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the "AVFormatContext" options or using the *libavutil/opt.h* API for programmatic use.

The list of supported options follows:

protocol_whitelist *list (input)*

Set a ","-separated list of allowed protocols. "ALL" matches all protocols. Protocols prefixed by "-" are disabled. All protocols are allowed by default but protocols used by an another protocol (nested protocols) are restricted to a per protocol subset.

PROTOCOLS

Protocols are configured elements in FFmpeg that enable access to resources that require specific protocols.

When you configure your FFmpeg build, all the supported protocols are enabled by default. You can list all available ones using the configure option "--list-protocols".

You can disable all the protocols using the configure option "--disable-protocols", and selectively enable a protocol using the option "--enable-protocol=*PROTOCOL*", or you can disable a particular protocol using the option "--disable-protocol=*PROTOCOL*".

The option "-protocols" of the ff* tools will display the list of supported protocols.

All protocols accept the following options:

rw_timeout

Maximum time to wait for (network) read/write operations to complete, in microseconds.

A description of the currently available protocols follows.

amqp

Advanced Message Queuing Protocol (AMQP) version 0-9-1 is a broker based publish-subscribe communication protocol.

FFmpeg must be compiled with --enable-librabbitmq to support AMQP. A separate AMQP broker must also be run. An example open-source AMQP broker is RabbitMQ.

After starting the broker, an FFmpeg client may stream data to the broker using the command:

```
ffmpeg -re -i input -f mpegts amqp://[[user]:[password]@]hostname[:port][/vhost]
```

Where hostname and port (default is 5672) is the address of the broker. The client may also set a user/password for authentication. The default for both fields is "guest". Name of virtual host on broker can be set with vhost. The default value is "".

Multiple subscribers may stream from the broker using the command:

```
ffplay amqp://[[user]:[password]@]hostname[:port][/vhost]
```

In RabbitMQ all data published to the broker flows through a specific exchange, and each subscribing client has an assigned queue/buffer. When a packet arrives at an exchange, it may be copied to a client's queue depending on the exchange and `routing_key` fields.

The following options are supported:

exchange

Sets the exchange to use on the broker. RabbitMQ has several predefined exchanges: "amq.direct" is the default exchange, where the publisher and subscriber must have a matching `routing_key`; "amq.fanout" is the same as a broadcast operation (i.e. the data is forwarded to all queues on the fanout exchange independent of the `routing_key`); and "amq.topic" is similar to "amq.direct", but allows for more complex pattern matching (refer to the RabbitMQ documentation).

routing_key

Sets the routing key. The default value is "amqp". The routing key is used on the "amq.direct" and "amq.topic" exchanges to decide whether packets are written to the queue of a subscriber.

pkt_size

Maximum size of each packet sent/received to the broker. Default is 131072. Minimum is 4096 and max is any large value (representable by an int). When receiving packets, this sets an internal buffer size in Ffmpeg. It should be equal to or greater than the size of the published packets to the broker. Otherwise the received message may be truncated causing decoding errors.

connection_timeout

The timeout in seconds during the initial connection to the broker. The default value is `rw_timeout`, or 5 seconds if `rw_timeout` is not set.

delivery_mode *mode*

Sets the delivery mode of each message sent to broker. The following values are accepted:

persistent

Delivery mode set to "persistent" (2). This is the default value. Messages may be written to the broker's disk depending on its setup.

non-persistent

Delivery mode set to "non-persistent" (1). Messages will stay in broker's memory unless the

broker is under memory pressure.

async

Asynchronous data filling wrapper for input stream.

Fill data in a background thread, to decouple I/O operation from demux thread.

```
async:<URL>
async:http://host/resource
async:cache:http://host/resource
```

bluray

Read BluRay playlist.

The accepted options are:

angle

BluRay angle

chapter

Start chapter (1...N)

playlist

Playlist to read (BDMV/PLAYLIST/?????.mpls)

Examples:

Read longest playlist from BluRay mounted to /mnt/bluray:

```
bluray:/mnt/bluray
```

Read angle 2 of playlist 4 from BluRay mounted to /mnt/bluray, start from chapter 2:

```
-playlist 4 -angle 2 -chapter 2 bluray:/mnt/bluray
```

cache

Caching wrapper for input stream.

Cache the input stream to temporary file. It brings seeking capability to live streams.

The accepted options are:

read_ahead_limit

Amount in bytes that may be read ahead when seeking isn't supported. Range is -1 to INT_MAX. -1 for unlimited. Default is 65536.

URL Syntax is

cache:<URL>

concat

Physical concatenation protocol.

Read and seek from many resources in sequence as if they were a unique resource.

A URL accepted by this protocol has the syntax:

concat:<URL1>|<URL2>|...|<URLN>

where *URL1*, *URL2*, ..., *URLN* are the urls of the resource to be concatenated, each one possibly specifying a distinct protocol.

For example to read a sequence of files *split1.mpeg*, *split2.mpeg*, *split3.mpeg* with **ffplay** use the command:

```
ffplay concat:split1.mpeg\|split2.mpeg\|split3.mpeg
```

Note that you may need to escape the character "|" which is special for many shells.

concatf

Physical concatenation protocol using a line break delimited list of resources.

Read and seek from many resources in sequence as if they were a unique resource.

A URL accepted by this protocol has the syntax:

concatf:<URL>

where *URL* is the url containing a line break delimited list of resources to be concatenated, each one possibly specifying a distinct protocol. Special characters must be escaped with backslash or single

quotes. See **the "Quoting and escaping" section in the ffmpeg-utils(1) manual.**

For example to read a sequence of files *split1.mpeg*, *split2.mpeg*, *split3.mpeg* listed in separate lines within a file *split.txt* with **ffplay** use the command:

```
ffplay concatf:split.txt
```

Where *split.txt* contains the lines:

```
split1.mpeg
split2.mpeg
split3.mpeg
```

crypto

AES-encrypted stream reading protocol.

The accepted options are:

key Set the AES decryption key binary block from given hexadecimal representation.

iv Set the AES decryption initialization vector binary block from given hexadecimal representation.

Accepted URL formats:

```
crypto:<URL>
crypto+<URL>
```

data

Data in-line in the URI. See http://en.wikipedia.org/wiki/Data_URI_scheme.

For example, to convert a GIF file given inline with **ffmpeg**:

```
ffmpeg -i "data:image/gif;base64,R0lGODdhCAAIAAMIEAAAAAAAAA//8AAP//AP//////////////////ywAAAAACAAIAA
```

fd

File descriptor access protocol.

The accepted syntax is:

```
fd: -fd <file_descriptor>
```

If **fd** is not specified, by default the stdout file descriptor will be used for writing, stdin for reading. Unlike the pipe protocol, fd protocol has seek support if it corresponding to a regular file. fd protocol doesn't support pass file descriptor via URL for security.

This protocol accepts the following options:

blocksize

Set I/O operation maximum block size, in bytes. Default value is "INT_MAX", which results in not limiting the requested block size. Setting this value reasonably low improves user termination request reaction time, which is valuable if data transmission is slow.

fd Set file descriptor.

file

File access protocol.

Read from or write to a file.

A file URL can have the form:

file:<filename>

where *filename* is the path of the file to read.

An URL that does not have a protocol prefix will be assumed to be a file URL. Depending on the build, an URL that looks like a Windows path with the drive letter at the beginning will also be assumed to be a file URL (usually not the case in builds for unix-like systems).

For example to read from a file *input.mpeg* with **ffmpeg** use the command:

```
ffmpeg -i file:input.mpeg output.mpeg
```

This protocol accepts the following options:

truncate

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

blocksize

Set I/O operation maximum block size, in bytes. Default value is "INT_MAX", which results in not limiting the requested block size. Setting this value reasonably low improves user termination

request reaction time, which is valuable for files on slow medium.

follow

If set to 1, the protocol will retry reading at the end of the file, allowing reading files that still are being written. In order for this to terminate, you either need to use the `rw_timeout` option, or use the interrupt callback (for API users).

seekable

Controls if seekability is advertised on the file. 0 means non-seekable, -1 means auto (seekable for normal files, non-seekable for named pipes).

Many demuxers handle seekable and non-seekable resources differently, overriding this might speed up opening certain files at the cost of losing some features (e.g. accurate seeking).

ftp

FTP (File Transfer Protocol).

Read from or write to remote resources using FTP protocol.

Following syntax is required.

```
ftp://[user[:password]@]server[:port]/path/to/remote/resource.mpeg
```

This protocol accepts the following options.

timeout

Set timeout in microseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

ftp-user

Set a user to be used for authenticating to the FTP server. This is overridden by the user in the FTP URL.

ftp-password

Set a password to be used for authenticating to the FTP server. This is overridden by the password in the FTP URL, or by **ftp-anonymous-password** if no user is set.

ftp-anonymous-password

Password used when login as anonymous user. Typically an e-mail address should be used.

ftp-write-seekable

Control seekability of connection during encoding. If set to 1 the resource is supposed to be seekable, if set to 0 it is assumed not to be seekable. Default value is 0.

NOTE: Protocol can be used as output, but it is recommended to not do it, unless special care is taken (tests, customized server configuration etc.). Different FTP servers behave in different way during seek operation. ff* tools may produce incomplete content due to server limitations.

gopher

Gopher protocol.

gophers

Gophers protocol.

The Gopher protocol with TLS encapsulation.

hls

Read Apple HTTP Live Streaming compliant segmented stream as a uniform one. The M3U8 playlists describing the segments can be remote HTTP resources or local files, accessed using the standard file protocol. The nested protocol is declared by specifying "+proto" after the hls URI scheme name, where *proto* is either "file" or "http".

```
hls+http://host/path/to/remote/resource.m3u8
```

```
hls+file://path/to/local/resource.m3u8
```

Using this protocol is discouraged - the hls demuxer should work just as well (if not, please report the issues) and is more complete. To use the hls demuxer instead, simply use the direct URLs to the m3u8 files.

http

HTTP (Hyper Text Transfer Protocol).

This protocol accepts the following options:

seekable

Control seekability of connection. If set to 1 the resource is supposed to be seekable, if set to 0 it is assumed not to be seekable, if set to -1 it will try to autodetect if it is seekable. Default value is -1.

chunked_post

If set to 1 use chunked Transfer-Encoding for posts, default is 1.

content_type

Set a specific content type for the POST messages or for listen mode.

http_proxy

set HTTP proxy to tunnel through e.g. http://example.com:1234

headers

Set custom HTTP headers, can override built in default headers. The value must be a string encoding the headers.

multiple_requests

Use persistent connections if set to 1, default is 0.

post_data

Set custom HTTP post data.

referer

Set the Referer header. Include 'Referer: URL' header in HTTP request.

user_agent

Override the User-Agent header. If not specified the protocol will use a string describing the libavformat build. ("Lavf/<version>")

reconnect_at_eof

If set then eof is treated like an error and causes reconnection, this is useful for live / endless streams.

reconnect_streamed

If set then even streamed/non seekable streams will be reconnected on errors.

reconnect_on_network_error

Reconnect automatically in case of TCP/TLS errors during connect.

reconnect_on_http_error

A comma separated list of HTTP status codes to reconnect on. The list can include specific status codes (e.g. '503') or the strings '4xx' / '5xx'.

reconnect_delay_max

Sets the maximum delay in seconds after which to give up reconnecting

mime_type

Export the MIME type.

http_version

Exports the HTTP response version number. Usually "1.0" or "1.1".

icy If set to 1 request ICY (SHOUTcast) metadata from the server. If the server supports this, the metadata has to be retrieved by the application by reading the **icy_metadata_headers** and **icy_metadata_packet** options. The default is 1.

icy_metadata_headers

If the server supports ICY metadata, this contains the ICY-specific HTTP reply headers, separated by newline characters.

icy_metadata_packet

If the server supports ICY metadata, and **icy** was set to 1, this contains the last non-empty metadata packet sent by the server. It should be polled in regular intervals by applications interested in mid-stream metadata updates.

cookies

Set the cookies to be sent in future requests. The format of each cookie is the same as the value of a Set-Cookie HTTP response field. Multiple cookies can be delimited by a newline character.

offset

Set initial byte offset.

end_offset

Try to limit the request to bytes preceding this offset.

method

When used as a client option it sets the HTTP method for the request.

When used as a server option it sets the HTTP method that is going to be expected from the client(s). If the expected and the received HTTP method do not match the client will be given a Bad Request response. When unset the HTTP method is not checked for now. This will be replaced by autodetection in the future.

listen

If set to 1 enables experimental HTTP server. This can be used to send data when used as an output option, or read data from a client with HTTP POST when used as an input option. If set to 2 enables experimental multi-client HTTP server. This is not yet implemented in ffmpeg.c and thus must not be used as a command line option.

Server side (sending):

```
ffmpeg -i somefile.ogg -c copy -listen 1 -f ogg http://<server>:<port>
```

Client side (receiving):

```
ffmpeg -i http://<server>:<port> -c copy somefile.ogg
```

Client can also be done with wget:

```
wget http://<server>:<port> -O somefile.ogg
```

Server side (receiving):

```
ffmpeg -listen 1 -i http://<server>:<port> -c copy somefile.ogg
```

Client side (sending):

```
ffmpeg -i somefile.ogg -chunked_post 0 -c copy -f ogg http://<server>:<port>
```

Client can also be done with wget:

```
wget --post-file=somefile.ogg http://<server>:<port>
```

send_expect_100

Send an Expect: 100-continue header for POST. If set to 1 it will send, if set to 0 it won't, if set to -1 it will try to send if it is applicable. Default value is -1.

auth_type

Set HTTP authentication type. No option for Digest, since this method requires getting nonce parameters from the server first and can't be used straight away like Basic.

none

Choose the HTTP authentication type automatically. This is the default.

basic

Choose the HTTP basic authentication.

Basic authentication sends a Base64-encoded string that contains a user name and password for the client. Base64 is not a form of encryption and should be considered the same as sending the user name and password in clear text (Base64 is a reversible encoding). If a

resource needs to be protected, strongly consider using an authentication scheme other than basic authentication. HTTPS/TLS should be used with basic authentication. Without these additional security enhancements, basic authentication should not be used to protect sensitive or valuable information.

HTTP Cookies

Some HTTP requests will be denied unless cookie values are passed in with the request. The **cookies** option allows these cookies to be specified. At the very least, each cookie must specify a value along with a path and domain. HTTP requests that match both the domain and path will automatically include the cookie value in the HTTP Cookie header field. Multiple cookies can be delimited by a newline.

The required syntax to play a stream specifying a cookie is:

```
ffplay -cookies "nlqptid=nlid=tsn; path=/; domain=somedomain.com;" http://somedomain.com/somestream.m3u8
```

Icecast

Icecast protocol (stream to Icecast servers)

This protocol accepts the following options:

ice_genre

Set the stream genre.

ice_name

Set the stream name.

ice_description

Set the stream description.

ice_url

Set the stream website URL.

ice_public

Set if the stream should be public. The default is 0 (not public).

user_agent

Override the User-Agent header. If not specified a string of the form "Lavf/<version>" will be used.

password

Set the Icecast mountpoint password.

content_type

Set the stream content type. This must be set if it is different from audio/mpeg.

legacy_icecast

This enables support for Icecast versions < 2.4.0, that do not support the HTTP PUT method but the SOURCE method.

tls Establish a TLS (HTTPS) connection to Icecast.

```
icecast://[<username>[:<password>]@]<server>:<port>/<mountpoint>
```

ipfs

InterPlanetary File System (IPFS) protocol support. One can access files stored on the IPFS network through so-called gateways. These are http(s) endpoints. This protocol wraps the IPFS native protocols (ipfs:// and ipns://) to be sent to such a gateway. Users can (and should) host their own node which means this protocol will use one's local gateway to access files on the IPFS network.

This protocol accepts the following options:

gateway

Defines the gateway to use. When not set, the protocol will first try locating the local gateway by looking at \$IPFS_GATEWAY, \$IPFS_PATH and "\$HOME/.ipfs/", in that order.

One can use this protocol in 2 ways. Using IPFS:

```
ffplay ipfs://<hash>
```

Or the IPNS protocol (IPNS is mutable IPFS):

```
ffplay ipns://<hash>
```

mmst

MMS (Microsoft Media Server) protocol over TCP.

mmsh

MMS (Microsoft Media Server) protocol over HTTP.

The required syntax is:

```
mmsch://<server>[:<port>][/<app>][/<playpath>]
```

md5

MD5 output protocol.

Computes the MD5 hash of the data to be written, and on close writes this to the designated output or stdout if none is specified. It can be used to test muxers without writing an actual file.

Some examples follow.

```
# Write the MD5 hash of the encoded AVI file to the file output.avi.md5.
ffmpeg -i input.flv -f avi -y md5:output.avi.md5
```

```
# Write the MD5 hash of the encoded AVI file to stdout.
ffmpeg -i input.flv -f avi -y md5:
```

Note that some formats (typically MOV) require the output protocol to be seekable, so they will fail with the MD5 output protocol.

pipe

UNIX pipe access protocol.

Read and write from UNIX pipes.

The accepted syntax is:

```
pipe:[<number>]
```

If **fd** isn't specified, *number* is the number corresponding to the file descriptor of the pipe (e.g. 0 for stdin, 1 for stdout, 2 for stderr). If *number* is not specified, by default the stdout file descriptor will be used for writing, stdin for reading.

For example to read from stdin with **ffmpeg**:

```
cat test.wav | ffmpeg -i pipe:0
# ...this is the same as...
cat test.wav | ffmpeg -i pipe:
```


For writing to stdout with **ffmpeg**:

```
ffmpeg -i test.wav -f avi pipe:1 | cat > test.avi
# ...this is the same as...
ffmpeg -i test.wav -f avi pipe: | cat > test.avi
```

This protocol accepts the following options:

blocksize

Set I/O operation maximum block size, in bytes. Default value is "INT_MAX", which results in not limiting the requested block size. Setting this value reasonably low improves user termination request reaction time, which is valuable if data transmission is slow.

fd Set file descriptor.

Note that some formats (typically MOV), require the output protocol to be seekable, so they will fail with the pipe output protocol.

prompeg

Pro-MPEG Code of Practice #3 Release 2 FEC protocol.

The Pro-MPEG CoP#3 FEC is a 2D parity-check forward error correction mechanism for MPEG-2 Transport Streams sent over RTP.

This protocol must be used in conjunction with the "rtp_mpegts" muxer and the "rtp" protocol.

The required syntax is:

```
-f rtp_mpegts -fec prompeg=<option>=<val>... rtp://<hostname>:<port>
```

The destination UDP ports are "port + 2" for the column FEC stream and "port + 4" for the row FEC stream.

This protocol accepts the following options:

l=n The number of columns (4-20, LxD <= 100)

d=n The number of rows (4-20, LxD <= 100)

Example usage:

```
-f rtp_mpegts -fec prompeg=l=8:d=4 rtp://<hostname>:<port>
```

rist

Reliable Internet Streaming Transport protocol

The accepted options are:

rist_profile

Supported values:

simple

main

This one is default.

advanced

buffer_size

Set internal RIST buffer size in milliseconds for retransmission of data. Default value is 0 which means the librist default (1 sec). Maximum value is 30 seconds.

fifo_size

Size of the librist receiver output fifo in number of packets. This must be a power of 2. Defaults to 8192 (vs the librist default of 1024).

overrun_nonfatal=1/0

Survive in case of librist fifo buffer overrun. Default value is 0.

pkt_size

Set maximum packet size for sending data. 1316 by default.

log_level

Set loglevel for RIST logging messages. You only need to set this if you explicitly want to enable debug level messages or packet loss simulation, otherwise the regular loglevel is respected.

secret

Set override of encryption secret, by default is unset.

encryption

Set encryption type, by default is disabled. Acceptable values are 128 and 256.

rtmp

Real-Time Messaging Protocol.

The Real-Time Messaging Protocol (RTMP) is used for streaming multimedia content across a TCP/IP network.

The required syntax is:

```
rtmp://[<username>:<password>@]<server>[:<port>][/<app>][/<instance>][/<playpath>]
```

The accepted parameters are:

username

An optional username (mostly for publishing).

password

An optional password (mostly for publishing).

server

The address of the RTMP server.

port The number of the TCP port to use (by default is 1935).

app It is the name of the application to access. It usually corresponds to the path where the application is installed on the RTMP server (e.g. */ondemand/*, */flash/live/*, etc.). You can override the value parsed from the URI through the "rtmp_app" option, too.

playpath

It is the path or name of the resource to play with reference to the application specified in *app*, may be prefixed by "mp4:". You can override the value parsed from the URI through the "rtmp_playpath" option, too.

listen

Act as a server, listening for an incoming connection.

timeout

Maximum time to wait for the incoming connection. Implies listen.

Additionally, the following parameters can be set via command line options (or in code via "AVOption"s):

rtmp_app

Name of application to connect on the RTMP server. This option overrides the parameter specified in the URI.

rtmp_buffer

Set the client buffer time in milliseconds. The default is 3000.

rtmp_conn

Extra arbitrary AMF connection parameters, parsed from a string, e.g. like "B:1 S:authMe O:1 NN:code:1.23 NS:flag:ok O:0". Each value is prefixed by a single character denoting the type, B for Boolean, N for number, S for string, O for object, or Z for null, followed by a colon. For Booleans the data must be either 0 or 1 for FALSE or TRUE, respectively. Likewise for Objects the data must be 0 or 1 to end or begin an object, respectively. Data items in subobjects may be named, by prefixing the type with 'N' and specifying the name before the value (i.e. "NB:myFlag:1"). This option may be used multiple times to construct arbitrary AMF sequences.

rtmp_flashver

Version of the Flash plugin used to run the SWF player. The default is LNX 9,0,124,2. (When publishing, the default is FMLE/3.0 (compatible; <libavformat version>).)

rtmp_flush_interval

Number of packets flushed in the same request (RTMPT only). The default is 10.

rtmp_live

Specify that the media is a live stream. No resuming or seeking in live streams is possible. The default value is "any", which means the subscriber first tries to play the live stream specified in the playpath. If a live stream of that name is not found, it plays the recorded stream. The other possible values are "live" and "recorded".

rtmp_pageurl

URL of the web page in which the media was embedded. By default no value will be sent.

rtmp_playpath

Stream identifier to play or to publish. This option overrides the parameter specified in the URI.

rtmp_subscribe

Name of live stream to subscribe to. By default no value will be sent. It is only sent if the option is specified or if rtmp_live is set to live.

rtmp_swfhash

SHA256 hash of the decompressed SWF file (32 bytes).

rtmp_swfsize

Size of the decompressed SWF file, required for SWFVerification.

rtmp_swfurl

URL of the SWF player for the media. By default no value will be sent.

rtmp_swfverify

URL to player swf file, compute hash/size automatically.

rtmp_tcurl

URL of the target stream. Defaults to proto://host[:port]/app.

tcp_nodelay=1/0

Set TCP_NODELAY to disable Nagle's algorithm. Default value is 0.

Remark: Writing to the socket is currently not optimized to minimize system calls and reduces the efficiency / effect of TCP_NODELAY.

For example to read with **ffplay** a multimedia resource named "sample" from the application "vod" from an RTMP server "myserver":

```
ffplay rtmp://myserver/vod/sample
```

To publish to a password protected server, passing the playpath and app names separately:

```
ffmpeg -re -i <input> -f flv -rtmp_playpath some/long/path -rtmp_app long/app/name rtmp://username:password@
```

rtmpe

Encrypted Real-Time Messaging Protocol.

The Encrypted Real-Time Messaging Protocol (RTMPE) is used for streaming multimedia content within standard cryptographic primitives, consisting of Diffie-Hellman key exchange and HMACSHA256, generating a pair of RC4 keys.

rtmps

Real-Time Messaging Protocol over a secure SSL connection.

The Real-Time Messaging Protocol (RTMPS) is used for streaming multimedia content across an

encrypted connection.

rtmpt

Real-Time Messaging Protocol tunneled through HTTP.

The Real-Time Messaging Protocol tunneled through HTTP (RTMPT) is used for streaming multimedia content within HTTP requests to traverse firewalls.

rtmpte

Encrypted Real-Time Messaging Protocol tunneled through HTTP.

The Encrypted Real-Time Messaging Protocol tunneled through HTTP (RTMPTE) is used for streaming multimedia content within HTTP requests to traverse firewalls.

rtmpts

Real-Time Messaging Protocol tunneled through HTTPS.

The Real-Time Messaging Protocol tunneled through HTTPS (RTMPTS) is used for streaming multimedia content within HTTPS requests to traverse firewalls.

libsmbclient

libsmbclient permits one to manipulate CIFS/SMB network resources.

Following syntax is required.

```
smb://[[domain:]user[:password@]]server[/share[/path[/file]]]
```

This protocol accepts the following options.

timeout

Set timeout in milliseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

truncate

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

workgroup

Set the workgroup used for making connections. By default workgroup is not specified.

For more information see: <<http://www.samba.org/>>.

libssh

Secure File Transfer Protocol via libssh

Read from or write to remote resources using SFTP protocol.

Following syntax is required.

```
sftp://[user[:password]@]server[:port]/path/to/remote/resource.mpeg
```

This protocol accepts the following options.

timeout

Set timeout of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

truncate

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

private_key

Specify the path of the file containing private key to use during authorization. By default libssh searches for keys in the `~/.ssh/` directory.

Example: Play a file stored on remote server.

```
ffplay sftp://user:password@server_address:22/home/user/resource.mpeg
```

librtmp rtmp, rtmpe, rtmpe, rtmpt, rtmpte

Real-Time Messaging Protocol and its variants supported through librtmp.

Requires the presence of the librtmp headers and library during configuration. You need to explicitly configure the build with "--enable-librtmp". If enabled this will replace the native RTMP protocol.

This protocol provides most client functions and a few server functions needed to support RTMP, RTMP tunneled in HTTP (RTMPT), encrypted RTMP (RTMPE), RTMP over SSL/TLS (RTMPS) and tunneled variants of these encrypted types (RTMPTE, RTMPTS).

The required syntax is:

```
<rtmp_proto>://<server>[:<port>][/<app>][/<playpath>] <options>
```

where *rtmp_proto* is one of the strings "rtmp", "rtmpt", "rtmpe", "rtmps", "rtmpte", "rtmpts" corresponding to each RTMP variant, and *server*, *port*, *app* and *playpath* have the same meaning as specified for the RTMP native protocol. *options* contains a list of space-separated options of the form *key=val*.

See the librtmp manual page (man 3 librtmp) for more information.

For example, to stream a file in real-time to an RTMP server using **ffmpeg**:

```
ffmpeg -re -i myfile -f flv rtmp://myserver/live/mystream
```

To play the same stream using **ffplay**:

```
ffplay "rtmp://myserver/live/mystream live=1"
```

rtp

Real-time Transport Protocol.

The required syntax for an RTP URL is: `rtp://hostname[:port][?option=val...]`

port specifies the RTP port to use.

The following URL options are supported:

ttl=*n*

Set the TTL (Time-To-Live) value (for multicast only).

rtcpport=*n*

Set the remote RTCP port to *n*.

localrtpport=*n*

Set the local RTP port to *n*.

localrtcpport=*n*'

Set the local RTCP port to *n*.

pkt_size=*n*

Set max packet size (in bytes) to *n*.

buffer_size=*size*

Set the maximum UDP socket buffer size in bytes.

connect=0|1

Do a "connect()" on the UDP socket (if set to 1) or not (if set to 0).

sources=ip[,ip]

List allowed source IP addresses.

block=ip[,ip]

List disallowed (blocked) source IP addresses.

write_to_source=0|1

Send packets to the source address of the latest received packet (if set to 1) or to a default remote address (if set to 0).

localport=n

Set the local RTP port to *n*.

localaddr=addr

Local IP address of a network interface used for sending packets or joining multicast groups.

timeout=n

Set timeout (in microseconds) of socket I/O operations to *n*.

This is a deprecated option. Instead, **localrtpport** should be used.

Important notes:

1. If **rtcpport** is not set the RTCP port will be set to the RTP port value plus 1.
2. If **localrtpport** (the local RTP port) is not set any available port will be used for the local RTP and RTCP ports.
3. If **localrtcpport** (the local RTCP port) is not set it will be set to the local RTP port value plus 1.

rtsp

Real-Time Streaming Protocol.

RTSP is not technically a protocol handler in libavformat, it is a demuxer and muxer. The demuxer supports both normal RTSP (with data transferred over RTP; this is used by e.g. Apple and Microsoft)

and Real-RTSP (with data transferred over RDT).

The muxer can be used to send a stream using RTSP ANNOUNCE to a server supporting it (currently Darwin Streaming Server and Mischa Spiegelmock's <<https://github.com/revmischa/rtsp-server>>).

The required syntax for a RTSP url is:

```
rtsp://<hostname>[:<port>]/<path>
```

Options can be set on the **ffmpeg/ffplay** command line, or set in code via "AVOption"s or in "avformat_open_input".

Muxer

The following options are supported.

rtsp_transport

Set RTSP transport protocols.

It accepts the following values:

udp Use UDP as lower transport protocol.

tcp Use TCP (interleaving within the RTSP control channel) as lower transport protocol.

Default value is **0**.

rtsp_flags

Set RTSP flags.

The following values are accepted:

latm

Use MP4A-LATM packetization instead of MPEG4-GENERIC for AAC.

rfc2190

Use RFC 2190 packetization instead of RFC 4629 for H.263.

skip_rtcp

Don't send RTCP sender reports.

h264_mode0

Use mode 0 for H.264 in RTP.

send_bye

Send RTCP BYE packets when finishing.

Default value is **0**.

min_port

Set minimum local UDP port. Default value is 5000.

max_port

Set maximum local UDP port. Default value is 65000.

buffer_size

Set the maximum socket buffer size in bytes.

pkt_size

Set max send packet size (in bytes). Default value is 1472.

Demuxer

The following options are supported.

initial_pause

Do not start playing the stream immediately if set to 1. Default value is 0.

rtsp_transport

Set RTSP transport protocols.

It accepts the following values:

udp Use UDP as lower transport protocol.

tcp Use TCP (interleaving within the RTSP control channel) as lower transport protocol.

udp_multicast

Use UDP multicast as lower transport protocol.

http Use HTTP tunneling as lower transport protocol, which is useful for passing proxies.

https

Use HTTPs tunneling as lower transport protocol, which is useful for passing proxies and widely used for security consideration.

Multiple lower transport protocols may be specified, in that case they are tried one at a time (if the setup of one fails, the next one is tried). For the muxer, only the **tcp** and **udp** options are supported.

rtsp_flags

Set RTSP flags.

The following values are accepted:

filter_src

Accept packets only from negotiated peer address and port.

listen

Act as a server, listening for an incoming connection.

prefer_tcp

Try TCP for RTP transport first, if TCP is available as RTSP RTP transport.

satip_raw

Export raw MPEG-TS stream instead of demuxing. The flag will simply write out the raw stream, with the original PAT/PMT/PIDs intact.

Default value is **none**.

allowed_media_types

Set media types to accept from the server.

The following flags are accepted:

video

audio

data

subtitle

By default it accepts all media types.

min_port

Set minimum local UDP port. Default value is 5000.

max_port

Set maximum local UDP port. Default value is 65000.

listen_timeout

Set maximum timeout (in seconds) to establish an initial connection. Setting **listen_timeout** > 0 sets **rtsp_flags** to **listen**. Default is -1 which means an infinite timeout when **listen** mode is set.

reorder_queue_size

Set number of packets to buffer for handling of reordered packets.

timeout

Set socket TCP I/O timeout in microseconds.

user_agent

Override User-Agent header. If not specified, it defaults to the libavformat identifier string.

buffer_size

Set the maximum socket buffer size in bytes.

When receiving data over UDP, the demuxer tries to reorder received packets (since they may arrive out of order, or packets may get lost totally). This can be disabled by setting the maximum demuxing delay to zero (via the "max_delay" field of AVFormatContext).

When watching multi-bitrate Real-RTSP streams with **ffplay**, the streams to display can be chosen with "-vst" *n* and "-ast" *n* for video and audio respectively, and can be switched on the fly by pressing "v" and "a".

Examples

The following examples all make use of the **ffplay** and **ffmpeg** tools.

- ⊕ Watch a stream over UDP, with a max reordering delay of 0.5 seconds:

```
ffplay -max_delay 500000 -rtsp_transport udp rtsp://server/video.mp4
```

- ⊕ Watch a stream tunneled over HTTP:

```
ffplay -rtsp_transport http rtsp://server/video.mp4
```

- ⊕ Send a stream in realtime to a RTSP server, for others to watch:

```
ffmpeg -re -i <input> -f rtsp -muxdelay 0.1 rtsp://server/live.sdp
```

- ⊕ Receive a stream in realtime:

```
ffmpeg -rtsp_flags listen -i rtsp://ownaddress/live.sdp <output>
```

sap

Session Announcement Protocol (RFC 2974). This is not technically a protocol handler in libavformat, it is a muxer and demuxer. It is used for signalling of RTP streams, by announcing the SDP for the streams regularly on a separate port.

Muxer

The syntax for a SAP url given to the muxer is:

```
sap://<destination>[:<port>][?<options>]
```

The RTP packets are sent to *destination* on port *port*, or to port 5004 if no port is specified. *options* is a "&"-separated list. The following options are supported:

announce_addr=*address*

Specify the destination IP address for sending the announcements to. If omitted, the announcements are sent to the commonly used SAP announcement multicast address 224.2.127.254 (sap.mcast.net), or ff0e::2:7ffe if *destination* is an IPv6 address.

announce_port=*port*

Specify the port to send the announcements on, defaults to 9875 if not specified.

ttl=*ttl*

Specify the time to live value for the announcements and RTP packets, defaults to 255.

same_port=*0/1*

If set to 1, send all RTP streams on the same port pair. If zero (the default), all streams are sent on unique ports, with each stream on a port 2 numbers higher than the previous. VLC/Live555 requires this to be set to 1, to be able to receive the stream. The RTP stack in libavformat for receiving requires all streams to be sent on unique ports.

Example command lines follow.

To broadcast a stream on the local subnet, for watching in VLC:

```
ffmpeg -re -i <input> -f sap sap://224.0.0.255?same_port=1
```

Similarly, for watching in **ffplay**:

```
ffmpeg -re -i <input> -f sap sap://224.0.0.255
```

And for watching in **ffplay**, over IPv6:

```
ffmpeg -re -i <input> -f sap sap://[ff0e::1:2:3:4]
```

Demuxer

The syntax for a SAP url given to the demuxer is:

```
sap://[<address>][:<port>]
```

address is the multicast address to listen for announcements on, if omitted, the default 224.2.127.254 (sap.mcast.net) is used. *port* is the port that is listened on, 9875 if omitted.

The demuxers listens for announcements on the given address and port. Once an announcement is received, it tries to receive that particular stream.

Example command lines follow.

To play back the first stream announced on the normal SAP multicast address:

```
ffplay sap://
```

To play back the first stream announced on one the default IPv6 SAP multicast address:

```
ffplay sap://[ff0e::2:7ffe]
```

sctp

Stream Control Transmission Protocol.

The accepted URL syntax is:

```
sctp://<host>:<port>[?<options>]
```

The protocol accepts the following options:

listen

If set to any value, listen for an incoming connection. Outgoing connection is done by default.

max_streams

Set the maximum number of streams. By default no limit is set.

srt

Haivision Secure Reliable Transport Protocol via libsrt.

The supported syntax for a SRT URL is:

```
srt://<hostname>:<port>[?<options>]
```

options contains a list of &-separated options of the form *key=val*.

or

```
<options> srt://<hostname>:<port>
```

options contains a list of '-key val' options.

This protocol accepts the following options.

connect_timeout=milliseconds

Connection timeout; SRT cannot connect for RTT > 1500 msec (2 handshake exchanges) with the default connect timeout of 3 seconds. This option applies to the caller and rendezvous connection modes. The connect timeout is 10 times the value set for the rendezvous mode (which can be used as a workaround for this connection problem with earlier versions).

ffs=bytes

Flight Flag Size (Window Size), in bytes. FFS is actually an internal parameter and you should set it to not less than **recv_buffer_size** and **mss**. The default value is relatively large, therefore unless you set a very large receiver buffer, you do not need to change this option. Default value is 25600.

inputbw=bytes/seconds

Sender nominal input rate, in bytes per seconds. Used along with **oheadbw**, when **maxbw** is set to

relative (0), to calculate maximum sending rate when recovery packets are sent along with the main media stream: $\mathbf{inputbw} * (100 + \mathbf{overheadbw}) / 100$ if $\mathbf{inputbw}$ is not set while \mathbf{maxbw} is set to relative (0), the actual input rate is evaluated inside the library. Default value is 0.

iptos=*tos*

IP Type of Service. Applies to sender only. Default value is 0xB8.

ipttl=*tll*

IP Time To Live. Applies to sender only. Default value is 64.

latency=*microseconds*

Timestamp-based Packet Delivery Delay. Used to absorb bursts of missed packet retransmissions. This flag sets both **rcvlatency** and **peerlatency** to the same value. Note that prior to version 1.3.0 this is the only flag to set the latency, however this is effectively equivalent to setting **peerlatency**, when side is sender and **rcvlatency** when side is receiver, and the bidirectional stream sending is not supported.

listen_timeout=*microseconds*

Set socket listen timeout.

maxbw=*bytes/seconds*

Maximum sending bandwidth, in bytes per seconds. -1 infinite (CSRTCC limit is 30mbps) 0 relative to input rate (see **inputbw**) >0 absolute limit value Default value is 0 (relative)

mode=*caller/listener/rendezvous*

Connection mode. **caller** opens client connection. **listener** starts server to listen for incoming connections. **rendezvous** use Rendez-Vous connection mode. Default value is caller.

mss=*bytes*

Maximum Segment Size, in bytes. Used for buffer allocation and rate calculation using a packet counter assuming fully filled packets. The smallest MSS between the peers is used. This is 1500 by default in the overall internet. This is the maximum size of the UDP packet and can be only decreased, unless you have some unusual dedicated network settings. Default value is 1500.

nakreport=*1/0*

If set to 1, Receiver will send 'UMSG_LOSSREPORT' messages periodically until a lost packet is retransmitted or intentionally dropped. Default value is 1.

overheadbw=*percents*

Recovery bandwidth overhead above input rate, in percents. See **inputbw**. Default value is 25%.

passphrase=string

HaiCrypt Encryption/Decryption Passphrase string, length from 10 to 79 characters. The passphrase is the shared secret between the sender and the receiver. It is used to generate the Key Encrypting Key using PBKDF2 (Password-Based Key Derivation Function). It is used only if **pbkeylen** is non-zero. It is used on the receiver only if the received data is encrypted. The configured passphrase cannot be recovered (write-only).

enforced_encryption=1/0

If true, both connection parties must have the same password set (including empty, that is, with no encryption). If the password doesn't match or only one side is unencrypted, the connection is rejected. Default is true.

kmrefreshrate=packets

The number of packets to be transmitted after which the encryption key is switched to a new key. Default is -1. -1 means auto (0x1000000 in srt library). The range for this option is integers in the 0 - "INT_MAX".

kmpreannounce=packets

The interval between when a new encryption key is sent and when switchover occurs. This value also applies to the subsequent interval between when switchover occurs and when the old encryption key is decommissioned. Default is -1. -1 means auto (0x1000 in srt library). The range for this option is integers in the 0 - "INT_MAX".

snddropdelay=microseconds

The sender's extra delay before dropping packets. This delay is added to the default drop delay time interval value.

Special value -1: Do not drop packets on the sender at all.

payload_size=bytes

Sets the maximum declared size of a packet transferred during the single call to the sending function in Live mode. Use 0 if this value isn't used (which is default in file mode). Default is -1 (automatic), which typically means MPEG-TS; if you are going to use SRT to send any different kind of payload, such as, for example, wrapping a live stream in very small frames, then you can use a bigger maximum frame size, though not greater than 1456 bytes.

pkt_size=bytes

Alias for **payload_size**.

peerlatency=microseconds

The latency value (as described in **rcvlatency**) that is set by the sender side as a minimum value for the receiver.

pbkeylen=bytes

Sender encryption key length, in bytes. Only can be set to 0, 16, 24 and 32. Enable sender encryption if not 0. Not required on receiver (set to 0), key size obtained from sender in HaiCrypt handshake. Default value is 0.

rcvlatency=microseconds

The time that should elapse since the moment when the packet was sent and the moment when it's delivered to the receiver application in the receiving function. This time should be a buffer time large enough to cover the time spent for sending, unexpectedly extended RTT time, and the time needed to retransmit the lost UDP packet. The effective latency value will be the maximum of this options' value and the value of **peerlatency** set by the peer side. Before version 1.3.0 this option is only available as **latency**.

rcv_buffer_size=bytes

Set UDP receive buffer size, expressed in bytes.

send_buffer_size=bytes

Set UDP send buffer size, expressed in bytes.

timeout=microseconds

Set raise error timeouts for read, write and connect operations. Note that the SRT library has internal timeouts which can be controlled separately, the value set here is only a cap on those.

tlpktdrop=1/0

Too-late Packet Drop. When enabled on receiver, it skips missing packets that have not been delivered in time and delivers the following packets to the application when their time-to-play has come. It also sends a fake ACK to the sender. When enabled on sender and enabled on the receiving peer, the sender drops the older packets that have no chance of being delivered in time. It was automatically enabled in the sender if the receiver supports it.

sndbuf=bytes

Set send buffer size, expressed in bytes.

rcvbuf=bytes

Set receive buffer size, expressed in bytes.

Receive buffer must not be greater than **ffs**.

lossmaxttl=*packets*

The value up to which the Reorder Tolerance may grow. When Reorder Tolerance is > 0 , then packet loss report is delayed until that number of packets come in. Reorder Tolerance increases every time a "belated" packet has come, but it wasn't due to retransmission (that is, when UDP packets tend to come out of order), with the difference between the latest sequence and this packet's sequence, and not more than the value of this option. By default it's 0, which means that this mechanism is turned off, and the loss report is always sent immediately upon experiencing a "gap" in sequences.

minversion

The minimum SRT version that is required from the peer. A connection to a peer that does not satisfy the minimum version requirement will be rejected.

The version format in hex is 0xXXYYZZ for x.y.z in human readable form.

streamid=*string*

A string limited to 512 characters that can be set on the socket prior to connecting. This stream ID will be able to be retrieved by the listener side from the socket that is returned from `srt_accept` and was connected by a socket with that set stream ID. SRT does not enforce any special interpretation of the contents of this string. This option doesn't make sense in Rendezvous connection; the result might be that simply one side will override the value from the other side and it's the matter of luck which one would win

srt_streamid=*string*

Alias for **streamid** to avoid conflict with ffmpeg command line option.

smoother=*live|file*

The type of Smoother used for the transmission for that socket, which is responsible for the transmission and congestion control. The Smoother type must be exactly the same on both connecting parties, otherwise the connection is rejected.

messageapi=*1/0*

When set, this socket uses the Message API, otherwise it uses Buffer API. Note that in live mode (see **transtype**) there's only message API available. In File mode you can choose to use one of two modes:

Stream API (default, when this option is false). In this mode you may send as many data as you wish with one sending instruction, or even use dedicated functions that read directly from a file. The internal facility will take care of any speed and congestion control. When receiving, you can also receive as many data as desired, the data not extracted will be waiting for the next call. There

is no boundary between data portions in the Stream mode.

Message API. In this mode your single sending instruction passes exactly one piece of data that has boundaries (a message). Contrary to Live mode, this message may span across multiple UDP packets and the only size limitation is that it shall fit as a whole in the sending buffer. The receiver shall use as large buffer as necessary to receive the message, otherwise the message will not be given up. When the message is not complete (not all packets received or there was a packet loss) it will not be given up.

transtype=live|file

Sets the transmission type for the socket, in particular, setting this option sets multiple other parameters to their default values as required for a particular transmission type.

live: Set options as for live transmission. In this mode, you should send by one sending instruction only so many data that fit in one UDP packet, and limited to the value defined first in **payload_size** (1316 is default in this mode). There is no speed control in this mode, only the bandwidth control, if configured, in order to not exceed the bandwidth with the overhead transmission (retransmitted and control packets).

file: Set options as for non-live transmission. See **messageapi** for further explanations

linger=seconds

The number of seconds that the socket waits for unsend data when closing. Default is -1. -1 means auto (off with 0 seconds in live mode, on with 180 seconds in file mode). The range for this option is integers in the 0 - "INT_MAX".

tsbpd=1/0

When true, use Timestamp-based Packet Delivery mode. The default behavior depends on the transmission type: enabled in live mode, disabled in file mode.

For more information see: <<https://github.com/Haivision/srt>>.

srt

Secure Real-time Transport Protocol.

The accepted options are:

srtp_in_suite

srtp_out_suite

Select input and output encoding suites.

Supported values:

AES_CM_128_HMAC_SHA1_80
SRTP_AES128_CM_HMAC_SHA1_80
AES_CM_128_HMAC_SHA1_32
SRTP_AES128_CM_HMAC_SHA1_32

srtp_in_params

srtp_out_params

Set input and output encoding parameters, which are expressed by a base64-encoded representation of a binary block. The first 16 bytes of this binary block are used as master key, the following 14 bytes are used as master salt.

subfile

Virtually extract a segment of a file or another stream. The underlying stream must be seekable.

Accepted options:

start Start offset of the extracted segment, in bytes.

end End offset of the extracted segment, in bytes. If set to 0, extract till end of file.

Examples:

Extract a chapter from a DVD VOB file (start and end sectors obtained externally and multiplied by 2048):

```
subfile,,start,153391104,end,268142592,,:/media/dvd/VIDEO_TS/VTS_08_1.VOB
```

Play an AVI file directly from a TAR archive:

```
subfile,,start,183241728,end,366490624,,:archive.tar
```

Play a MPEG-TS file from start offset till end:

```
subfile,,start,32815239,end,0,,:video.ts
```

tee

Writes the output to multiple protocols. The individual outputs are separated by |

```
tee:file://path/to/local/this.avi|file://path/to/local/that.avi
```

tcp

Transmission Control Protocol.

The required syntax for a TCP url is:

```
tcp://<hostname>:<port>[?<options>]
```

options contains a list of &-separated options of the form *key=val*.

The list of supported options follows.

listen=2/1/0

Listen for an incoming connection. 0 disables listen, 1 enables listen in single client mode, 2 enables listen in multi-client mode. Default value is 0.

timeout=microseconds

Set raise error timeout, expressed in microseconds.

This option is only relevant in read mode: if no data arrived in more than this time interval, raise error.

listen_timeout=milliseconds

Set listen timeout, expressed in milliseconds.

recv_buffer_size=bytes

Set receive buffer size, expressed bytes.

send_buffer_size=bytes

Set send buffer size, expressed bytes.

tcp_nodelay=1/0

Set TCP_NODELAY to disable Nagle's algorithm. Default value is 0.

Remark: Writing to the socket is currently not optimized to minimize system calls and reduces the efficiency / effect of TCP_NODELAY.

tcp_mss=bytes

Set maximum segment size for outgoing TCP packets, expressed in bytes.

The following example shows how to setup a listening TCP connection with **ffmpeg**, which is then

accessed with **ffplay**:

```
ffmpeg -i <input> -f <format> tcp://<hostname>:<port>?listen  
ffplay tcp://<hostname>:<port>
```

tls

Transport Layer Security (TLS) / Secure Sockets Layer (SSL)

The required syntax for a TLS/SSL url is:

```
tls://<hostname>:<port>[?<options>]
```

The following parameters can be set via command line options (or in code via "AVOption"s):

ca_file, cafile=filename

A file containing certificate authority (CA) root certificates to treat as trusted. If the linked TLS library contains a default this might not need to be specified for verification to work, but not all libraries and setups have defaults built in. The file must be in OpenSSL PEM format.

tls_verify=1/0

If enabled, try to verify the peer that we are communicating with. Note, if using OpenSSL, this currently only makes sure that the peer certificate is signed by one of the root certificates in the CA database, but it does not validate that the certificate actually matches the host name we are trying to connect to. (With other backends, the host name is validated as well.)

This is disabled by default since it requires a CA database to be provided by the caller in many cases.

cert_file, cert=filename

A file containing a certificate to use in the handshake with the peer. (When operating as server, in listen mode, this is more often required by the peer, while client certificates only are mandated in certain setups.)

key_file, key=filename

A file containing the private key for the certificate.

listen=1/0

If enabled, listen for connections on the provided port, and assume the server role in the handshake instead of the client role.

http_proxy

The HTTP proxy to tunnel through, e.g. "http://example.com:1234". The proxy must support the CONNECT method.

Example command lines:

To create a TLS/SSL server that serves an input stream.

```
ffmpeg -i <input> -f <format> tls://<hostname>:<port>?listen&cert=<server.crt>&key=<server.key>
```

To play back a stream from the TLS/SSL server using **ffplay**:

```
ffplay tls://<hostname>:<port>
```

udp

User Datagram Protocol.

The required syntax for an UDP URL is:

```
udp://<hostname>:<port>[?<options>]
```

options contains a list of &-separated options of the form *key=val*.

In case threading is enabled on the system, a circular buffer is used to store the incoming data, which allows one to reduce loss of data due to UDP socket buffer overruns. The *fifo_size* and *overrun_nonfatal* options are related to this buffer.

The list of supported options follows.

buffer_size=size

Set the UDP maximum socket buffer size in bytes. This is used to set either the receive or send buffer size, depending on what the socket is used for. Default is 32 KB for output, 384 KB for input. See also *fifo_size*.

bitrate=bitrate

If set to nonzero, the output will have the specified constant bitrate if the input has enough packets to sustain it.

burst_bits=bits

When using *bitrate* this specifies the maximum number of bits in packet bursts.

localport=port

Override the local UDP port to bind with.

localaddr=addr

Local IP address of a network interface used for sending packets or joining multicast groups.

pkt_size=size

Set the size in bytes of UDP packets.

reuse=1/0

Explicitly allow or disallow reusing UDP sockets.

ttl=tvl

Set the time to live value (for multicast only).

connect=1/0

Initialize the UDP socket with "connect()". In this case, the destination address can't be changed with `ff_udp_set_remote_url` later. If the destination address isn't known at the start, this option can be specified in `ff_udp_set_remote_url`, too. This allows finding out the source address for the packets with `getsockname`, and makes writes return with `AVERROR(ECONNREFUSED)` if "destination unreachable" is received. For receiving, this gives the benefit of only receiving packets from the specified peer address/port.

sources=address[,address]

Only receive packets sent from the specified addresses. In case of multicast, also subscribe to multicast traffic coming from these addresses only.

block=address[,address]

Ignore packets sent from the specified addresses. In case of multicast, also exclude the source addresses in the multicast subscription.

fifo_size=units

Set the UDP receiving circular buffer size, expressed as a number of packets with size of 188 bytes. If not specified defaults to $7*4096$.

overrun_nonfatal=1/0

Survive in case of UDP receiving circular buffer overrun. Default value is 0.

timeout=microseconds

Set raise error timeout, expressed in microseconds.

This option is only relevant in read mode: if no data arrived in more than this time interval, raise error.

broadcast=1/0

Explicitly allow or disallow UDP broadcasting.

Note that broadcasting may not work properly on networks having a broadcast storm protection.

Examples

- ⊕ Use **ffmpeg** to stream over UDP to a remote endpoint:

```
ffmpeg -i <input> -f <format> udp://<hostname>:<port>
```

- ⊕ Use **ffmpeg** to stream in mpegts format over UDP using 188 sized UDP packets, using a large input buffer:

```
ffmpeg -i <input> -f mpegts udp://<hostname>:<port>?pkt_size=188&buffer_size=65535
```

- ⊕ Use **ffmpeg** to receive over UDP from a remote endpoint:

```
ffmpeg -i udp://[<multicast-address>]:<port> ...
```

unix

Unix local socket

The required syntax for a Unix socket URL is:

```
unix://<filepath>
```

The following parameters can be set via command line options (or in code via "AVOption"s):

timeout

Timeout in ms.

listen

Create the Unix socket in listening mode.

zmq

ZeroMQ asynchronous messaging using the libzmq library.

This library supports unicast streaming to multiple clients without relying on an external server.

The required syntax for streaming or connecting to a stream is:

```
zmq:tcp://ip-address:port
```

Example: Create a localhost stream on port 5555:

```
ffmpeg -re -i input -f mpegts zmq:tcp://127.0.0.1:5555
```

Multiple clients may connect to the stream using:

```
ffplay zmq:tcp://127.0.0.1:5555
```

Streaming to multiple clients is implemented using a ZeroMQ Pub-Sub pattern. The server side binds to a port and publishes data. Clients connect to the server (via IP address/port) and subscribe to the stream. The order in which the server and client start generally does not matter.

ffmpeg must be compiled with the `--enable-libzmq` option to support this protocol.

Options can be set on the **ffmpeg/ffplay** command line. The following options are supported:

pkt_size

Forces the maximum packet size for sending/receiving data. The default value is 131,072 bytes. On the server side, this sets the maximum size of sent packets via ZeroMQ. On the clients, it sets an internal buffer size for receiving packets. Note that `pkt_size` on the clients should be equal to or greater than `pkt_size` on the server. Otherwise the received message may be truncated causing decoding errors.

DEVICE OPTIONS

The `libavdevice` library provides the same interface as `libavformat`. Namely, an input device is considered like a demuxer, and an output device like a muxer, and the interface and generic device options are the same provided by `libavformat` (see the `ffmpeg-formats` manual).

In addition each input or output device may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the Ffmpeg tools, or by setting the value explicitly in the device "AVFormatContext" options or using the `libavutil/opt.h` API for programmatic use.

INPUT DEVICES

Input devices are configured elements in FFmpeg which enable accessing the data coming from a multimedia device attached to your system.

When you configure your FFmpeg build, all the supported input devices are enabled by default. You can list all available ones using the configure option "--list-indevs".

You can disable all the input devices using the configure option "--disable-indevs", and selectively enable an input device using the option "--enable-indev=*INDEV*", or you can disable a particular input device using the option "--disable-indev=*INDEV*".

The option "-devices" of the ff* tools will display the list of supported input devices.

A description of the currently available input devices follows.

alsa

ALSA (Advanced Linux Sound Architecture) input device.

To enable this input device during configuration you need libasound installed on your system.

This device allows capturing from an ALSA device. The name of the device to capture has to be an ALSA card identifier.

An ALSA identifier has the syntax:

```
hw:<CARD>[,<DEV>[,<SUBDEV>]]
```

where the *DEV* and *SUBDEV* components are optional.

The three arguments (in order: *CARD,DEV,SUBDEV*) specify card number or identifier, device number and subdevice number (-1 means any).

To see the list of cards currently recognized by your system check the files */proc/asound/cards* and */proc/asound/devices*.

For example to capture with **ffmpeg** from an ALSA device with card id 0, you may run the command:

```
ffmpeg -f alsa -i hw:0 alsaout.wav
```

For more information see: <<http://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html>>

*Options***sample_rate**

Set the sample rate in Hz. Default is 48000.

channels

Set the number of channels. Default is 2.

android_camera

Android camera input device.

This input devices uses the Android Camera2 NDK API which is available on devices with API level 24+. The availability of android_camera is autodetected during configuration.

This device allows capturing from all cameras on an Android device, which are integrated into the Camera2 NDK API.

The available cameras are enumerated internally and can be selected with the *camera_index* parameter. The input file string is discarded.

Generally the back facing camera has index 0 while the front facing camera has index 1.

*Options***video_size**

Set the video size given as a string such as 640x480 or hd720. Falls back to the first available configuration reported by Android if requested video size is not available or by default.

framerate

Set the video framerate. Falls back to the first available configuration reported by Android if requested framerate is not available or by default (-1).

camera_index

Set the index of the camera to use. Default is 0.

input_queue_size

Set the maximum number of frames to buffer. Default is 5.

avfoundation

AVFoundation input device.

AVFoundation is the currently recommended framework by Apple for streamgrabbing on OSX >= 10.7 as well as on iOS.

The input filename has to be given in the following syntax:

```
-i "[[VIDEO]:[AUDIO]]"
```

The first entry selects the video input while the latter selects the audio input. The stream has to be specified by the device name or the device index as shown by the device list. Alternatively, the video and/or audio input device can be chosen by index using the

```
B<-video_device_index E<lt>INDEXE<gt>>
```

and/or

```
B<-audio_device_index E<lt>INDEXE<gt>>
```

, overriding any device name or index given in the input filename.

All available devices can be enumerated by using **-list_devices true**, listing all device names and corresponding indices.

There are two device name aliases:

"default"

Select the AVFoundation default device of the corresponding type.

"none"

Do not record the corresponding media type. This is equivalent to specifying an empty device name or index.

Options

AVFoundation supports the following options:

-list_devices <TRUE|FALSE>

If set to true, a list of all available input devices is given showing all device names and indices.

-video_device_index <INDEX>

Specify the video device by its index. Overrides anything given in the input filename.

-audio_device_index <INDEX>

Specify the audio device by its index. Overrides anything given in the input filename.

-pixel_format <FORMAT>

Request the video device to use a specific pixel format. If the specified format is not supported, a list of available formats is given and the first one in this list is used instead. Available pixel formats are: "monob, rgb555be, rgb555le, rgb565be, rgb565le, rgb24, bgr24, 0rgb, bgr0, 0bgr, rgb0, bgr48be, uyvy422, yuva444p, yuva444p16le, yuv444p, yuv422p16, yuv422p10, yuv444p10, yuv420p, nv12, yuyv422, gray"

-framerate

Set the grabbing frame rate. Default is "ntsc", corresponding to a frame rate of "30000/1001".

-video_size

Set the video frame size.

-capture_cursor

Capture the mouse pointer. Default is 0.

-capture_mouse_clicks

Capture the screen mouse clicks. Default is 0.

-capture_raw_data

Capture the raw device data. Default is 0. Using this option may result in receiving the underlying data delivered to the AVFoundation framework. E.g. for muxed devices that sends raw DV data to the framework (like tape-based camcorders), setting this option to false results in extracted video frames captured in the designated pixel format only. Setting this option to true results in receiving the raw DV stream untouched.

Examples

- ⊕ Print the list of AVFoundation supported devices and exit:

```
$ ffmpeg -f avfoundation -list_devices true -i ""
```

- ⊕ Record video from video device 0 and audio from audio device 0 into out.avi:

```
$ ffmpeg -f avfoundation -i "0:0" out.avi
```


- ⊕ Record video from video device 2 and audio from audio device 1 into out.avi:

```
$ ffmpeg -f avfoundation -video_device_index 2 -i ":1" out.avi
```

- ⊕ Record video from the system default video device using the pixel format bgr0 and do not record any audio into out.avi:

```
$ ffmpeg -f avfoundation -pixel_format bgr0 -i "default:none" out.avi
```

- ⊕ Record raw DV data from a suitable input device and write the output into out.dv:

```
$ ffmpeg -f avfoundation -capture_raw_data true -i "zr100:none" out.dv
```

bktr

BSD video input device.

Options

framerate

Set the frame rate.

video_size

Set the video frame size. Default is "vga".

standard

Available values are:

pal

ntsc

secam

palm

palm

ntsej

decklink

The decklink input device provides capture capabilities for Blackmagic DeckLink devices.

To enable this input device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate "--extra-cflags" and "--extra-ldflags". On Windows, you need to run the IDL files through **widl**.

DeckLink is very picky about the formats it supports. Pixel format of the input can be set with **raw_format**. Framerate and video size must be determined for your device with **-list_formats 1**. Audio sample rate is always 48 kHz and the number of channels can be 2, 8 or 16. Note that all audio channels are bundled in one single audio track.

Options

list_devices

If set to **true**, print a list of devices and exit. Defaults to **false**. This option is deprecated, please use the "-sources" option of ffmpeg to list the available input devices.

list_formats

If set to **true**, print a list of supported formats and exit. Defaults to **false**.

format_code <FourCC>

This sets the input video format to the format given by the FourCC. To see the supported values of your device(s) use **list_formats**. Note that there is a FourCC '**pal**' that can also be used as **pal** (3 letters). Default behavior is autodetection of the input video format, if the hardware supports it.

raw_format

Set the pixel format of the captured video. Available values are:

auto This is the default which means 8-bit YUV 422 or 8-bit ARGB if format autodetection is used, 8-bit YUV 422 otherwise.

uyvy422

8-bit YUV 422.

yuv422p10

10-bit YUV 422.

argb

8-bit RGB.

bgra

8-bit RGB.

rgb10

10-bit RGB.

teletext_lines

If set to nonzero, an additional teletext stream will be captured from the vertical ancillary data. Both SD PAL (576i) and HD (1080i or 1080p) sources are supported. In case of HD sources, OP47 packets are decoded.

This option is a bitmask of the SD PAL VBI lines captured, specifically lines 6 to 22, and lines 318 to 335. Line 6 is the LSB in the mask. Selected lines which do not contain teletext information will be ignored. You can use the special **all** constant to select all possible lines, or **standard** to skip lines 6, 318 and 319, which are not compatible with all receivers.

For SD sources, ffmpeg needs to be compiled with "--enable-libzvbi". For HD sources, on older (pre-4K) DeckLink card models you have to capture in 10 bit mode.

channels

Defines number of audio channels to capture. Must be **2**, **8** or **16**. Defaults to **2**.

duplex_mode

Sets the decklink device duplex/profile mode. Must be **unset**, **half**, **full**, **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half** Defaults to **unset**.

Note: DeckLink SDK 11.0 have replaced the duplex property by a profile property. For the DeckLink Duo 2 and DeckLink Quad 2, a profile is shared between any 2 sub-devices that utilize the same connectors. For the DeckLink 8K Pro, a profile is shared between all 4 sub-devices. So DeckLink 8K Pro support four profiles.

Valid profile modes for DeckLink 8K Pro(with DeckLink SDK >= 11.0): **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half**

Valid profile modes for DeckLink Quad 2 and DeckLink Duo 2: **half**, **full**

timecode_format

Timecode type to include in the frame and video stream metadata. Must be **none**, **rp188vitr**, **rp188vitr2**, **rp188lrc**, **rp188hfr**, **rp188any**, **vitr**, **vitr2**, or **serial**. Defaults to **none** (not included).

In order to properly support 50/60 fps timecodes, the ordering of the queried timecode types for **rp188any** is HFR, VITC1, VITC2 and LTC for >30 fps content. Note that this is slightly different to the ordering used by the DeckLink API, which is HFR, VITC1, LTC, VITC2.

video_input

Sets the video input source. Must be **unset**, **sdi**, **hdmi**, **optical_sdi**, **component**, **composite** or

s_video. Defaults to **unset**.

audio_input

Sets the audio input source. Must be **unset**, **embedded**, **aes_ebu**, **analog**, **analog_xlr**, **analog_rca** or **microphone**. Defaults to **unset**.

video_pts

Sets the video packet timestamp source. Must be **video**, **audio**, **reference**, **wallclock** or **abs_wallclock**. Defaults to **video**.

audio_pts

Sets the audio packet timestamp source. Must be **video**, **audio**, **reference**, **wallclock** or **abs_wallclock**. Defaults to **audio**.

drawBars

If set to **true**, color bars are drawn in the event of a signal loss. Defaults to **true**.

queue_size

Sets maximum input buffer size in bytes. If the buffering reaches this value, incoming frames will be dropped. Defaults to **1073741824**.

audio_depth

Sets the audio sample bit depth. Must be **16** or **32**. Defaults to **16**.

decklink_copyts

If set to **true**, timestamps are forwarded as they are without removing the initial offset. Defaults to **false**.

timestamp_align

Capture start time alignment in seconds. If set to nonzero, input frames are dropped till the system timestamp aligns with configured value. Alignment difference of up to one frame duration is tolerated. This is useful for maintaining input synchronization across N different hardware devices deployed for 'N-way' redundancy. The system time of different hardware devices should be synchronized with protocols such as NTP or PTP, before using this option. Note that this method is not foolproof. In some border cases input synchronization may not happen due to thread scheduling jitters in the OS. Either sync could go wrong by 1 frame or in a rarer case **timestamp_align** seconds. Defaults to **0**.

wait_for_tc (*bool*)

Drop frames till a frame with timecode is received. Sometimes serial timecode isn't received with

the first input frame. If that happens, the stored stream timecode will be inaccurate. If this option is set to **true**, input frames are dropped till a frame with timecode is received. Option *timecode_format* must be specified. Defaults to **false**.

enable_klv(*bool*)

If set to **true**, extracts KLV data from VANC and outputs KLV packets. KLV VANC packets are joined based on MID and PSC fields and aggregated into one KLV packet. Defaults to **false**.

Examples

- ⊕ List input devices:

```
ffmpeg -sources decklink
```

- ⊕ List supported formats:

```
ffmpeg -f decklink -list_formats 1 -i 'Intensity Pro'
```

- ⊕ Capture video clip at 1080i50:

```
ffmpeg -format_code Hi50 -f decklink -i 'Intensity Pro' -c:a copy -c:v copy output.avi
```

- ⊕ Capture video clip at 1080i50 10 bit:

```
ffmpeg -raw_format yuv422p10 -format_code Hi50 -f decklink -i 'UltraStudio Mini Recorder' -c:a copy -c:v
```

- ⊕ Capture video clip at 1080i50 with 16 audio channels:

```
ffmpeg -channels 16 -format_code Hi50 -f decklink -i 'UltraStudio Mini Recorder' -c:a copy -c:v copy outpu
```

dshow

Windows DirectShow input device.

DirectShow support is enabled when FFmpeg is built with the mingw-w64 project. Currently only audio and video devices are supported.

Multiple devices may be opened as separate inputs, but they may also be opened on the same input, which should improve synchronism between them.

The input name should be in the format:

<TYPE>=<NAME>[:<TYPE>=<NAME>]

where *TYPE* can be either *audio* or *video*, and *NAME* is the device's name or alternative name..

Options

If no options are specified, the device's defaults are used. If the device does not support the requested options, it will fail to open.

video_size

Set the video size in the captured video.

framerate

Set the frame rate in the captured video.

sample_rate

Set the sample rate (in Hz) of the captured audio.

sample_size

Set the sample size (in bits) of the captured audio.

channels

Set the number of channels in the captured audio.

list_devices

If set to **true**, print a list of devices and exit.

list_options

If set to **true**, print a list of selected device's options and exit.

video_device_number

Set video device number for devices with the same name (starts at 0, defaults to 0).

audio_device_number

Set audio device number for devices with the same name (starts at 0, defaults to 0).

pixel_format

Select pixel format to be used by DirectShow. This may only be set when the video codec is not set or set to rawvideo.

audio_buffer_size

Set audio device buffer size in milliseconds (which can directly impact latency, depending on the device). Defaults to using the audio device's default buffer size (typically some multiple of 500ms). Setting this value too low can degrade performance. See also [http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582(v=vs.85).aspx)

video_pin_name

Select video capture pin to use by name or alternative name.

audio_pin_name

Select audio capture pin to use by name or alternative name.

crossbar_video_input_pin_number

Select video input pin number for crossbar device. This will be routed to the crossbar device's Video Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

crossbar_audio_input_pin_number

Select audio input pin number for crossbar device. This will be routed to the crossbar device's Audio Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

show_video_device_dialog

If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to change video filter properties and configurations manually. Note that for crossbar devices, adjusting values in this dialog may be needed at times to toggle between PAL (25 fps) and NTSC (29.97) input frame rates, sizes, interlacing, etc. Changing these values can enable different scan rates/frame rates and avoiding green bars at the bottom, flickering scan lines, etc. Note that with some devices, changing these properties can also affect future invocations (sets new defaults) until system reboot occurs.

show_audio_device_dialog

If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to change audio filter properties and configurations manually.

show_video_crossbar_connection_dialog

If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens a video device.

show_audio_crossbar_connection_dialog

If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens an audio device.

show_analog_tv_tuner_dialog

If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV channels and frequencies.

show_analog_tv_tuner_audio_dialog

If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV audio (like mono vs. stereo, Language A,B or C).

audio_device_load

Load an audio capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this an audio capture source has to be specified, but it can be anything even fake one.

audio_device_save

Save the currently used audio capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

video_device_load

Load a video capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this a video capture source has to be specified, but it can be anything even fake one.

video_device_save

Save the currently used video capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

use_video_device_timestamps

If set to **false**, the timestamp for video frames will be derived from the wallclock instead of the timestamp provided by the capture device. This allows working around devices that provide unreliable timestamps.

Examples

- ⊕ Print the list of DirectShow supported devices and exit:

```
$ ffmpeg -list_devices true -f dshow -i dummy
```


- ⊕ Open video device *Camera*:

```
$ ffmpeg -f dshow -i video="Camera"
```

- ⊕ Open second video device with name *Camera*:

```
$ ffmpeg -f dshow -video_device_number 1 -i video="Camera"
```

- ⊕ Open video device *Camera* and audio device *Microphone*:

```
$ ffmpeg -f dshow -i video="Camera":audio="Microphone"
```

- ⊕ Print the list of supported options in selected device and exit:

```
$ ffmpeg -list_options true -f dshow -i video="Camera"
```

- ⊕ Specify pin names to capture by name or alternative name, specify alternative device name:

```
$ ffmpeg -f dshow -audio_pin_name "Audio Out" -video_pin_name 2 -i video=video="@device_pnp_\\?\pci\
```

- ⊕ Configure a crossbar device, specifying crossbar pins, allow user to adjust video capture properties at startup:

```
$ ffmpeg -f dshow -show_video_device_dialog true -crossbar_video_input_pin_number 0  
-crossbar_audio_input_pin_number 3 -i video="AVerMedia BDA Analog Capture":audio="AVerMedia B
```

fbdev

Linux framebuffer input device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually */dev/fb0*.

For more detailed information read the file *Documentation/fb/framebuffer.txt* included in the Linux source tree.

See also <<http://linux-fbdev.sourceforge.net/>>, and **fbset**(1).

To record from the framebuffer device */dev/fb0* with **ffmpeg**:

```
ffmpeg -f fbdev -framerate 10 -i /dev/fb0 out.avi
```

You can take a single screenshot image with the command:

```
ffmpeg -f fbdev -framerate 1 -i /dev/fb0 -frames:v 1 screenshot.jpeg
```

Options

framerate

Set the frame rate. Default is 25.

gdigrab

Win32 GDI-based screen capture device.

This device allows you to capture a region of the display on Windows.

There are two options for the input filename:

desktop

or

title=<window_title>

The first option will capture the entire desktop, or a fixed region of the desktop. The second option will instead capture the contents of a single window, regardless of its position on the screen.

For example, to grab the entire desktop using **ffmpeg**:

```
ffmpeg -f gdigrab -framerate 6 -i desktop out.mpg
```

Grab a 640x480 region at position "10,20":

```
ffmpeg -f gdigrab -framerate 6 -offset_x 10 -offset_y 20 -video_size vga -i desktop out.mpg
```

Grab the contents of the window named "Calculator"

```
ffmpeg -f gdigrab -framerate 6 -i title=Calculator out.mpg
```

Options

draw_mouse

Specify whether to draw the mouse pointer. Use the value 0 to not draw the pointer. Default value is 1.

framerate

Set the grabbing frame rate. Default value is "ntsc", corresponding to a frame rate of "30000/1001".

show_region

Show grabbed region on screen.

If *show_region* is specified with 1, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

Note that *show_region* is incompatible with grabbing the contents of a single window.

For example:

```
ffmpeg -f gdigrab -show_region 1 -framerate 6 -video_size cif -offset_x 10 -offset_y 20 -i desktop out.mpg
```

video_size

Set the video frame size. The default is to capture the full screen if *desktop* is selected, or the full window size if *title=window_title* is selected.

offset_x

When capturing a region with *video_size*, set the distance from the left edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned to the left of your primary monitor, you will need to use a negative *offset_x* value to move the region to that monitor.

offset_y

When capturing a region with *video_size*, set the distance from the top edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned above your primary monitor, you will need to use a negative *offset_y* value to move the region to that monitor.

iec61883

FireWire DV/HDV input device using libiec61883.

To enable this input device, you need libiec61883, libraw1394 and libavc1394 installed on your system. Use the configure option "--enable-libiec61883" to compile with the device enabled.

The iec61883 capture device supports capturing from a video device connected via IEEE1394 (FireWire), using libiec61883 and the new Linux FireWire stack (juju). This is the default DV/HDV input method in Linux Kernel 2.6.37 and later, since the old FireWire stack was removed.

Specify the FireWire port to be used as input file, or "auto" to choose the first port connected.

Options

dvtype

Override autodetection of DV/HDV. This should only be used if auto detection does not work, or if usage of a different device type should be prohibited. Treating a DV device as HDV (or vice versa) will not work and result in undefined behavior. The values **auto**, **dv** and **hdv** are supported.

dvbuffer

Set maximum size of buffer for incoming data, in frames. For DV, this is an exact value. For HDV, it is not frame exact, since HDV does not have a fixed frame size.

dvguid

Select the capture device by specifying its GUID. Capturing will only be performed from the specified device and fails if no device with the given GUID is found. This is useful to select the input if multiple devices are connected at the same time. Look at `/sys/bus/firewire/devices` to find out the GUIDs.

Examples

- ⊕ Grab and show the input of a FireWire DV/HDV device.

```
ffplay -f iec61883 -i auto
```

- ⊕ Grab and record the input of a FireWire DV/HDV device, using a packet buffer of 100000 packets if the source is HDV.

```
ffmpeg -f iec61883 -i auto -dvbuffer 100000 out.mpg
```

jack

JACK input device.

To enable this input device during configuration you need libjack installed on your system.

A JACK input device creates one or more JACK writable clients, one for each audio channel, with name *client_name*:input_*N*, where *client_name* is the name provided by the application, and *N* is a number which identifies the channel. Each writable client will send the acquired data to the FFMpeg input device.

Once you have created one or more JACK readable clients, you need to connect them to one or more JACK writable clients.

To connect or disconnect JACK clients you can use the **jack_connect** and **jack_disconnect** programs, or do it through a graphical interface, for example with **qjackctl**.

To list the JACK clients and their properties you can invoke the command **jack_lsp**.

Follows an example which shows how to capture a JACK readable client with **ffmpeg**.

```
# Create a JACK writable client with name "ffmpeg".
```

```
$ ffmpeg -f jack -i ffmpeg -y out.wav
```

```
# Start the sample jack_metro readable client.
```

```
$ jack_metro -b 120 -d 0.2 -f 4000
```

```
# List the current JACK clients.
```

```
$ jack_lsp -c
```

```
system:capture_1
```

```
system:capture_2
```

```
system:playback_1
```

```
system:playback_2
```

```
ffmpeg:input_1
```

```
metro:120_bpm
```

```
# Connect metro to the ffmpeg writable client.
```

```
$ jack_connect metro:120_bpm ffmpeg:input_1
```

For more information read: <<http://jackaudio.org/>>

Options

channels

Set the number of channels. Default is 2.

kmsgrab

KMS video input device.

Captures the KMS scanout framebuffer associated with a specified CRTC or plane as a DRM object that can be passed to other hardware functions.

Requires either DRM master or CAP_SYS_ADMIN to run.

If you don't understand what all of that means, you probably don't want this. Look at **x11grab** instead.

*Options***device**

DRM device to capture on. Defaults to **/dev/dri/card0**.

format

Pixel format of the framebuffer. This can be autodetected if you are running Linux 5.7 or later, but needs to be provided for earlier versions. Defaults to **bgr0**, which is the most common format used by the Linux console and Xorg X server.

format_modifier

Format modifier to signal on output frames. This is necessary to import correctly into some APIs. It can be autodetected if you are running Linux 5.7 or later, but will need to be provided explicitly when needed in earlier versions. See the libdrm documentation for possible values.

crtc_id

KMS CRTC ID to define the capture source. The first active plane on the given CRTC will be used.

plane_id

KMS plane ID to define the capture source. Defaults to the first active plane found if neither **crtc_id** nor **plane_id** are specified.

framerate

Framerate to capture at. This is not synchronised to any page flipping or framebuffer changes - it just defines the interval at which the framebuffer is sampled. Sampling faster than the framebuffer update rate will generate independent frames with the same content. Defaults to 30.

Examples

- ⊕ Capture from the first active plane, download the result to normal frames and encode. This will only work if the framebuffer is both linear and mappable - if not, the result may be scrambled or fail to download.

```
ffmpeg -f kmsgrab -i - -vf 'hwdownload,format=bgr0' output.mp4
```

- ⊕ Capture from CRTC ID 42 at 60fps, map the result to VAAPI, convert to NV12 and encode as H.264.

```
ffmpeg -crtc_id 42 -framerate 60 -f kmsgrab -i - -vf 'hwmap=derive_device=vaapi,scale_vaapi=w=1920:h=1080' output.mp4
```

- ⊕ To capture only part of a plane the output can be cropped - this can be used to capture a single window, as long as it has a known absolute position and size. For example, to capture and encode the middle quarter of a 1920x1080 plane:

```
ffmpeg -f kmsgrab -i - -vf 'hwmap=derive_device=vaapi,crop=960:540:480:270,scale_vaapi=960:540:nv12' output.mp4
```

lavfi

Libavfilter input virtual device.

This input device reads data from the open output pads of a libavfilter filtergraph.

For each filtergraph open output, the input device will create a corresponding stream which is mapped to the generated output. Currently only video data is supported. The filtergraph is specified through the option **graph**.

*Options***graph**

Specify the filtergraph to use as input. Each video open output must be labelled by a unique string of the form "outN", where *N* is a number starting from 0 corresponding to the mapped input stream generated by the device. The first unlabelled output is automatically assigned to the "out0" label, but all the others need to be specified explicitly.

The suffix "+subcc" can be appended to the output label to create an extra stream with the closed captions packets attached to that output (experimental; only for EIA-608 / CEA-708 for now). The subcc streams are created after all the normal streams, in the order of the corresponding stream. For example, if there is "out19+subcc", "out7+subcc" and up to "out42", the stream #43 is

subcc for stream #7 and stream #44 is subcc for stream #19.

If not specified defaults to the filename specified for the input device.

graph_file

Set the filename of the filtergraph to be read and sent to the other filters. Syntax of the filtergraph is the same as the one specified by the option *graph*.

dumpgraph

Dump graph to stderr.

Examples

- ⊕ Create a color video stream and play it back with **ffplay**:

```
ffplay -f lavfi -graph "color=c=pink [out0]" dummy
```

- ⊕ As the previous example, but use filename for specifying the graph description, and omit the "out0" label:

```
ffplay -f lavfi color=c=pink
```

- ⊕ Create three different video test filtered sources and play them:

```
ffplay -f lavfi -graph "testsrc [out0]; testsrc,hflip [out1]; testsrc,negate [out2]" test3
```

- ⊕ Read an audio stream from a file using the amovie source and play it back with **ffplay**:

```
ffplay -f lavfi "amovie=test.wav"
```

- ⊕ Read an audio stream and a video stream and play it back with **ffplay**:

```
ffplay -f lavfi "movie=test.avi[out0];amovie=test.wav[out1]"
```

- ⊕ Dump decoded frames to images and closed captions to a file (experimental):

```
ffmpeg -f lavfi -i "movie=test.ts[out0+subcc]" -map v frame%08d.png -map s -c copy -f rawvideo subcc.bin
```

libcdio

Audio-CD input device based on libcdio.

To enable this input device during configuration you need libcdio installed on your system. It requires the configure option "--enable-libcdio".

This device allows playing and grabbing from an Audio-CD.

For example to copy with **ffmpeg** the entire Audio-CD in `/dev/sr0`, you may run the command:

```
ffmpeg -f libcdio -i /dev/sr0 cd.wav
```

Options

speed

Set drive reading speed. Default value is 0.

The speed is specified CD-ROM speed units. The speed is set through the libcdio "cdio_cddap_speed_set" function. On many CD-ROM drives, specifying a value too large will result in using the fastest speed.

paranoia_mode

Set paranoia recovery mode flags. It accepts one of the following values:

disable
verify
overlap
neverskip
full

Default value is **disable**.

For more information about the available recovery modes, consult the paranoia project documentation.

libdc1394

IIDC1394 input device, based on libdc1394 and libraw1394.

Requires the configure option "--enable-libdc1394".

Options

framerate

Set the frame rate. Default is "ntsc", corresponding to a frame rate of "30000/1001".

pixel_format

Select the pixel format. Default is "uyvy422".

video_size

Set the video size given as a string such as "640x480" or "hd720". Default is "qvga".

openal

The OpenAL input device provides audio capture on all systems with a working OpenAL 1.1 implementation.

To enable this input device during configuration, you need OpenAL headers and libraries installed on your system, and need to configure FFmpeg with "--enable-openal".

OpenAL headers and libraries should be provided as part of your OpenAL implementation, or as an additional download (an SDK). Depending on your installation you may need to specify additional flags via the "--extra-cflags" and "--extra-ldflags" for allowing the build system to locate the OpenAL headers and libraries.

An incomplete list of OpenAL implementations follows:

Creative

The official Windows implementation, providing hardware acceleration with supported devices and software fallback. See <<http://openal.org/>>.

OpenAL Soft

Portable, open source (LGPL) software implementation. Includes backends for the most common sound APIs on the Windows, Linux, Solaris, and BSD operating systems. See <<http://kcat.strangesoft.net/openal.html>>.

Apple

OpenAL is part of Core Audio, the official Mac OS X Audio interface. See <<http://developer.apple.com/technologies/mac/audio-and-video.html>>

This device allows one to capture from an audio input device handled through OpenAL.

You need to specify the name of the device to capture in the provided filename. If the empty string is provided, the device will automatically select the default device. You can get the list of the supported devices by using the option *list_devices*.

*Options***channels**

Set the number of channels in the captured audio. Only the values **1** (monaural) and **2** (stereo) are currently supported. Defaults to **2**.

sample_size

Set the sample size (in bits) of the captured audio. Only the values **8** and **16** are currently supported. Defaults to **16**.

sample_rate

Set the sample rate (in Hz) of the captured audio. Defaults to **44.1k**.

list_devices

If set to **true**, print a list of devices and exit. Defaults to **false**.

Examples

Print the list of OpenAL supported devices and exit:

```
$ ffmpeg -list_devices true -f openal -i dummy out.ogg
```

Capture from the OpenAL device *DR-BT101 via PulseAudio*:

```
$ ffmpeg -f openal -i 'DR-BT101 via PulseAudio' out.ogg
```

Capture from the default device (note the empty string '' as filename):

```
$ ffmpeg -f openal -i '' out.ogg
```

Capture from two devices simultaneously, writing to two different files, within the same **ffmpeg** command:

```
$ ffmpeg -f openal -i 'DR-BT101 via PulseAudio' out1.ogg -f openal -i 'ALSA Default' out2.ogg
```

Note: not all OpenAL implementations support multiple simultaneous capture - try the latest OpenAL Soft if the above does not work.

oss

Open Sound System input device.

The filename to provide to the input device is the device node representing the OSS input device, and is usually set to `/dev/dsp`.

For example to grab from `/dev/dsp` using **ffmpeg** use the command:

```
ffmpeg -f oss -i /dev/dsp /tmp/oss.wav
```

For more information about OSS see: <<http://manuals.opensound.com/usersguide/dsp.html>>

Options

sample_rate

Set the sample rate in Hz. Default is 48000.

channels

Set the number of channels. Default is 2.

pulse

PulseAudio input device.

To enable this output device you need to configure FFmpeg with "`--enable-libpulse`".

The filename to provide to the input device is a source device or the string "default"

To list the PulseAudio source devices and their properties you can invoke the command **pactl list sources**.

More information about PulseAudio can be found on <<http://www.pulseaudio.org>>.

Options

server

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

name

Specify the application name PulseAudio will use when showing active clients, by default it is the "LIBAVFORMAT_IDENT" string.

stream_name

Specify the stream name PulseAudio will use when showing active streams, by default it is "record".

sample_rate

Specify the samplerate in Hz, by default 48kHz is used.

channels

Specify the channels in use, by default 2 (stereo) is set.

frame_size

This option does nothing and is deprecated.

fragment_size

Specify the size in bytes of the minimal buffering fragment in PulseAudio, it will affect the audio latency. By default it is set to 50 ms amount of data.

wallclock

Set the initial PTS using the current time. Default is 1.

Examples

Record a stream from default device:

```
ffmpeg -f pulse -i default /tmp/pulse.wav
```

sndio

sndio input device.

To enable this input device during configuration you need libsndio installed on your system.

The filename to provide to the input device is the device node representing the sndio input device, and is usually set to */dev/audio0*.

For example to grab from */dev/audio0* using **ffmpeg** use the command:

```
ffmpeg -f sndio -i /dev/audio0 /tmp/oss.wav
```

*Options***sample_rate**

Set the sample rate in Hz. Default is 48000.

channels

Set the number of channels. Default is 2.

video4linux2, v4l2

Video4Linux2 input video device.

"v4l2" can be used as alias for "video4linux2".

If FFmpeg is built with v4l-utils support (by using the "--enable-libv4l2" configure option), it is possible to use it with the "-use_libv4l2" input device option.

The name of the device to grab is a file device node, usually Linux systems tend to automatically create such nodes when the device (e.g. an USB webcam) is plugged into the system, and has a name of the kind */dev/videoN*, where *N* is a number associated to the device.

Video4Linux2 devices usually support a limited set of *widthxheight* sizes and frame rates. You can check which are supported using **-list_formats all** for Video4Linux2 devices. Some devices, like TV cards, support one or more standards. It is possible to list all the supported standards using **-list_standards all**.

The time base for the timestamps is 1 microsecond. Depending on the kernel version and configuration, the timestamps may be derived from the real time clock (origin at the Unix Epoch) or the monotonic clock (origin usually at boot time, unaffected by NTP or manual changes to the clock). The **-timestamps abs** or **-ts abs** option can be used to force conversion into the real time clock.

Some usage examples of the video4linux2 device with **ffmpeg** and **ffplay**:

- ⊕ List supported formats for a video4linux2 device:

```
ffplay -f video4linux2 -list_formats all /dev/video0
```

- ⊕ Grab and show the input of a video4linux2 device:

```
ffplay -f video4linux2 -framerate 30 -video_size hd720 /dev/video0
```

- ⊕ Grab and record the input of a video4linux2 device, leave the frame rate and size as previously set:

```
ffmpeg -f video4linux2 -input_format mjpeg -i /dev/video0 out.mpeg
```

For more information about Video4Linux, check <<http://linuxtv.org/>>.

Options

standard

Set the standard. Must be the name of a supported standard. To get a list of the supported standards, use the **list_standards** option.

channel

Set the input channel number. Default to -1, which means using the previously selected channel.

video_size

Set the video frame size. The argument must be a string in the form *WIDTHxHEIGHT* or a valid size abbreviation.

pixel_format

Select the pixel format (only valid for raw video input).

input_format

Set the preferred pixel format (for raw video) or a codec name. This option allows one to select the input format, when several are available.

framerate

Set the preferred video frame rate.

list_formats

List available formats (supported pixel formats, codecs, and frame sizes) and exit.

Available values are:

all Show all available (compressed and non-compressed) formats.

raw Show only raw video (non-compressed) formats.

compressed

Show only compressed formats.

list_standards

List supported standards and exit.

Available values are:

all Show all supported standards.

timestamps, ts

Set type of timestamps for grabbed frames.

Available values are:

default

Use timestamps from the kernel.

abs Use absolute timestamps (wall clock).

mono2abs

Force conversion from monotonic to absolute timestamps.

Default value is "default".

use_libv4l2

Use libv4l2 (v4l-utils) conversion functions. Default is 0.

vfwcaps

VfW (Video for Windows) capture input device.

The filename passed as input is the capture driver number, ranging from 0 to 9. You may use "list" as filename to print a list of drivers. Any other filename will be interpreted as device number 0.

Options

video_size

Set the video frame size.

framerate

Set the grabbing frame rate. Default value is "ntsc", corresponding to a frame rate of "30000/1001".

x11grab

X11 video input device.

To enable this input device during configuration you need libxcb installed on your system. It will be automatically detected during configuration.

This device allows one to capture a region of an X11 display.

The filename passed as input has the syntax:

```
[<hostname>]:<display_number>.<screen_number>[+<x_offset>,<y_offset>]
```

hostname:display_number.screen_number specifies the X11 display name of the screen to grab from. *hostname* can be omitted, and defaults to "localhost". The environment variable **DISPLAY** contains the default display name.

x_offset and *y_offset* specify the offsets of the grabbed area with respect to the top-left border of the X11 screen. They default to 0.

Check the X11 documentation (e.g. **man X**) for more detailed information.

Use the **xdpinfo** program for getting basic information about the properties of your X11 display (e.g. `grep` for "name" or "dimensions").

For example to grab from `:0.0` using **ffmpeg**:

```
ffmpeg -f x11grab -framerate 25 -video_size cif -i :0.0 out.mpg
```

Grab at position "10,20":

```
ffmpeg -f x11grab -framerate 25 -video_size cif -i :0.0+10,20 out.mpg
```

Options

select_region

Specify whether to select the grabbing area graphically using the pointer. A value of 1 prompts the user to select the grabbing area graphically by clicking and dragging. A single click with no dragging will select the whole screen. A region with zero width or height will also select the whole screen. This option overwrites the *video_size*, *grab_x*, and *grab_y* options. Default value is 0.

draw_mouse

Specify whether to draw the mouse pointer. A value of 0 specifies not to draw the pointer. Default value is 1.

follow_mouse

Make the grabbed area follow the mouse. The argument can be "centered" or a number of pixels *PIXELS*.

When it is specified with "centered", the grabbing region follows the mouse pointer and keeps the pointer at the center of region; otherwise, the region follows only when the mouse pointer reaches within *PIXELS* (greater than zero) to the edge of region.

For example:

```
ffmpeg -f x11grab -follow_mouse centered -framerate 25 -video_size cif -i :0.0 out.mpg
```

To follow only when the mouse pointer reaches within 100 pixels to edge:

```
ffmpeg -f x11grab -follow_mouse 100 -framerate 25 -video_size cif -i :0.0 out.mpg
```

framerate

Set the grabbing frame rate. Default value is "ntsc", corresponding to a frame rate of "30000/1001".

show_region

Show grabbed region on screen.

If *show_region* is specified with 1, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

region_border

Set the region border thickness if **-show_region 1** is used. Range is 1 to 128 and default is 3 (XCB-based x11grab only).

For example:

```
ffmpeg -f x11grab -show_region 1 -framerate 25 -video_size cif -i :0.0+10,20 out.mpg
```

With *follow_mouse*:

```
ffmpeg -f x11grab -follow_mouse centered -show_region 1 -framerate 25 -video_size cif -i :0.0 out.mpg
```

window_id

Grab this window, instead of the whole screen. Default value is 0, which maps to the whole screen (root window).

The id of a window can be found using the **xwininfo** program, possibly with options **-tree** and **-root**.

If the window is later enlarged, the new area is not recorded. Video ends when the window is closed, unmapped (i.e., iconified) or shrunk beyond the video size (which defaults to the initial window size).

This option disables options **follow_mouse** and **select_region**.

video_size

Set the video frame size. Default is the full desktop or window.

grab_x**grab_y**

Set the grabbing region coordinates. They are expressed as offset from the top left corner of the X11 window and correspond to the *x_offset* and *y_offset* parameters in the device name. The default value for both options is 0.

OUTPUT DEVICES

Output devices are configured elements in FFmpeg that can write multimedia data to an output device attached to your system.

When you configure your FFmpeg build, all the supported output devices are enabled by default. You can list all available ones using the configure option "**--list-outdevs**".

You can disable all the output devices using the configure option "**--disable-outdevs**", and selectively enable an output device using the option "**--enable-outdev=OUTDEV**", or you can disable a particular input device using the option "**--disable-outdev=OUTDEV**".

The option "**-devices**" of the ff* tools will display the list of enabled output devices.

A description of the currently available output devices follows.

alsa

ALSA (Advanced Linux Sound Architecture) output device.

Examples

- ⊕ Play a file on default ALSA device:

```
ffmpeg -i INPUT -f alsa default
```

- ⊕ Play a file on soundcard 1, audio device 7:

```
ffmpeg -i INPUT -f alsa hw:1,7
```

AudioToolbox

AudioToolbox output device.

Allows native output to CoreAudio devices on OSX.

The output filename can be empty (or "-") to refer to the default system output device or a number that refers to the device index as shown using: "-list_devices true".

Alternatively, the audio input device can be chosen by index using the

```
B<-audio_device_index E<lt>INDEXE<gt>>
```

, overriding any device name or index given in the input filename.

All available devices can be enumerated by using **-list_devices true**, listing all device names, UIDs and corresponding indices.

Options

AudioToolbox supports the following options:

-audio_device_index <INDEX>

Specify the audio device by its index. Overrides anything given in the output filename.

Examples

- ⊕ Print the list of supported devices and output a sine wave to the default device:

```
$ ffmpeg -f lavfi -i sine=r=44100 -f audiotoolbox -list_devices true -
```

- ⊕ Output a sine wave to the device with the index 2, overriding any output filename:

```
$ ffmpeg -f lavfi -i sine=r=44100 -f audiotoolbox -audio_device_index 2 -
```

caca

CACA output device.

This output device allows one to show a video stream in CACA window. Only one CACA window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need to configure FFmpeg with "--enable-libcaca". libcaca is a graphics library that outputs text instead of pixels.

For more information about libcaca, check: <<http://caca.zoy.org/wiki/libcaca>>

Options

window_title

Set the CACA window title, if not specified default to the filename specified for the output device.

window_size

Set the CACA window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video.

driver

Set display driver.

algorithm

Set dithering algorithm. Dithering is necessary because the picture being rendered has usually far more colours than the available palette. The accepted values are listed with "-list_dither algorithms".

antialias

Set antialias method. Antialiasing smoothens the rendered image and avoids the commonly seen staircase effect. The accepted values are listed with "-list_dither antialiases".

charset

Set which characters are going to be used when rendering text. The accepted values are listed

with "-list_dither charsets".

color

Set color to be used when rendering text. The accepted values are listed with "-list_dither colors".

list_drivers

If set to **true**, print a list of available drivers and exit.

list_dither

List available dither options related to the argument. The argument must be one of "algorithms", "antialiases", "charsets", "colors".

Examples

- ⊕ The following command shows the **ffmpeg** output is an CACA window, forcing its size to 80x25:

```
ffmpeg -i INPUT -c:v rawvideo -pix_fmt rgb24 -window_size 80x25 -f caca -
```

- ⊕ Show the list of available drivers and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_drivers true -
```

- ⊕ Show the list of available dither colors and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_dither colors -
```

decklink

The decklink output device provides playback capabilities for Blackmagic DeckLink devices.

To enable this output device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate "--extra-cflags" and "--extra-ldflags". On Windows, you need to run the IDL files through **widl**.

DeckLink is very picky about the formats it supports. Pixel format is always uyvy422, framerate, field order and video size must be determined for your device with **-list_formats 1**. Audio sample rate is always 48 kHz.

*Options***list_devices**

If set to **true**, print a list of devices and exit. Defaults to **false**. This option is deprecated, please use the "-sinks" option of ffmpeg to list the available output devices.

list_formats

If set to **true**, print a list of supported formats and exit. Defaults to **false**.

preroll

Amount of time to preroll video in seconds. Defaults to **0.5**.

duplex_mode

Sets the decklink device duplex/profile mode. Must be **unset**, **half**, **full**, **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half** Defaults to **unset**.

Note: DeckLink SDK 11.0 have replaced the duplex property by a profile property. For the DeckLink Duo 2 and DeckLink Quad 2, a profile is shared between any 2 sub-devices that utilize the same connectors. For the DeckLink 8K Pro, a profile is shared between all 4 sub-devices. So DeckLink 8K Pro support four profiles.

Valid profile modes for DeckLink 8K Pro(with DeckLink SDK >= 11.0): **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half**

Valid profile modes for DeckLink Quad 2 and DeckLink Duo 2: **half**, **full**

timing_offset

Sets the genlock timing pixel offset on the used output. Defaults to **unset**.

link Sets the SDI video link configuration on the used output. Must be **unset**, **single** link SDI, **dual** link SDI or **quad** link SDI. Defaults to **unset**.

sqd Enable Square Division Quad Split mode for Quad-link SDI output. Must be **unset**, **true** or **false**. Defaults to **unset**.

level_a

Enable SMPTE Level A mode on the used output. Must be **unset**, **true** or **false**. Defaults to **unset**.

Examples

⊕ List output devices:

```
ffmpeg -sinks decklink
```

- ⊕ List supported formats:

```
ffmpeg -i test.avi -f decklink -list_formats 1 'DeckLink Mini Monitor'
```

- ⊕ Play video clip:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 'DeckLink Mini Monitor'
```

- ⊕ Play video clip with non-standard framerate or video size:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 -s 720x486 -r 24000/1001 'DeckLink Mini Monitor'
```

fbdev

Linux framebuffer output device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually */dev/fb0*.

For more detailed information read the file *Documentation/fb/framebuffer.txt* included in the Linux source tree.

Options

xoffset

yoffset

Set x/y coordinate of top left corner. Default is 0.

Examples

Play a file on framebuffer device */dev/fb0*. Required pixel format depends on current framebuffer settings.

```
ffmpeg -re -i INPUT -c:v rawvideo -pix_fmt bgra -f fbdev /dev/fb0
```

See also <<http://linux-fbdev.sourceforge.net/>>, and **fbset(1)**.

opengl

OpenGL output device.

To enable this output device you need to configure FFmpeg with "--enable-opengl".

This output device allows one to render to OpenGL context. Context may be provided by application or default SDL window is created.

When device renders to external context, application must implement handlers for following messages:

"AV_DEV_TO_APP_CREATE_WINDOW_BUFFER" - create OpenGL context on current thread.

"AV_DEV_TO_APP_PREPARE_WINDOW_BUFFER" - make OpenGL context current.

"AV_DEV_TO_APP_DISPLAY_WINDOW_BUFFER" - swap buffers.

"AV_DEV_TO_APP_DESTROY_WINDOW_BUFFER" - destroy OpenGL context. Application is also required to inform a device about current resolution by sending

"AV_APP_TO_DEV_WINDOW_SIZE" message.

Options

background

Set background color. Black is a default.

no_window

Disables default SDL window when set to non-zero value. Application must provide OpenGL context and both "window_size_cb" and "window_swap_buffers_cb" callbacks when set.

window_title

Set the SDL window title, if not specified default to the filename specified for the output device. Ignored when **no_window** is set.

window_size

Set preferred window size, can be a string of the form widthxheight or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio. Mostly usable when **no_window** is not set.

Examples

Play a file on SDL window using OpenGL rendering:

```
ffmpeg -i INPUT -f opengl "window title"
```

oss

OSS (Open Sound System) output device.

pulse

PulseAudio output device.

To enable this output device you need to configure FFmpeg with "--enable-libpulse".

More information about PulseAudio can be found on <<http://www.pulseaudio.org>>

Options

server

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

name

Specify the application name PulseAudio will use when showing active clients, by default it is the "LIBAVFORMAT_IDENT" string.

stream_name

Specify the stream name PulseAudio will use when showing active streams, by default it is set to the specified output name.

device

Specify the device to use. Default device is used when not provided. List of output devices can be obtained with command **pactl list sinks**.

buffer_size

buffer_duration

Control the size and duration of the PulseAudio buffer. A small buffer gives more control, but requires more frequent updates.

buffer_size specifies size in bytes while **buffer_duration** specifies duration in milliseconds.

When both options are provided then the highest value is used (duration is recalculated to bytes using stream parameters). If they are set to 0 (which is default), the device will use the default PulseAudio duration value. By default PulseAudio set buffer duration to around 2 seconds.

prebuf

Specify pre-buffering size in bytes. The server does not start with playback before at least **prebuf** bytes are available in the buffer. By default this option is initialized to the same value as **buffer_size** or **buffer_duration** (whichever is bigger).

minreq

Specify minimum request size in bytes. The server does not request less than **minreq** bytes from

the client, instead waits until the buffer is free enough to request more bytes at once. It is recommended to not set this option, which will initialize this to a value that is deemed sensible by the server.

Examples

Play a file on default device on default server:

```
ffmpeg -i INPUT -f pulse "stream name"
```

sdl

SDL (Simple DirectMedia Layer) output device.

"sdl2" can be used as alias for "sdl".

This output device allows one to show a video stream in an SDL window. Only one SDL window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need libSDL installed on your system when configuring your build.

For more information about SDL, check: [<http://www.libsdl.org/>](http://www.libsdl.org/)

Options

window_title

Set the SDL window title, if not specified default to the filename specified for the output device.

icon_title

Set the name of the iconified SDL window, if not specified it is set to the same value of *window_title*.

window_size

Set the SDL window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio.

window_x

window_y

Set the position of the window on the screen.

window_fullscreen

Set fullscreen mode when non-zero value is provided. Default value is zero.

window_enable_quit

Enable quit action (using window button or keyboard key) when non-zero value is provided.
Default value is 1 (enable quit action)

Interactive commands

The window created by the device can be controlled through the following interactive commands.

q, ESC

Quit the device immediately.

Examples

The following command shows the **ffmpeg** output is an SDL window, forcing its size to the qcif format:

```
ffmpeg -i INPUT -c:v rawvideo -pix_fmt yuv420p -window_size qcif -f sdl "SDL output"
```

sndio

sndio audio output device.

v4l2

Video4Linux2 output device.

xv

XV (XVideo) output device.

This output device allows one to show a video stream in a X Window System window.

*Options***display_name**

Specify the hardware display name, which determines the display and communications domain to be used.

The display name or DISPLAY environment variable can be a string in the format *hostname[:number[.screen_number]]*.

hostname specifies the name of the host machine on which the display is physically attached. *number* specifies the number of the display server on that host machine. *screen_number* specifies the screen to be used on that server.

If unspecified, it defaults to the value of the DISPLAY environment variable.

For example, "dual-headed:0.1" would specify screen 1 of display 0 on the machine named "dual-headed".

Check the X11 specification for more detailed information about the display name format.

window_id

When set to non-zero value then device doesn't create new window, but uses existing one with provided *window_id*. By default this options is set to zero and device creates its own window.

window_size

Set the created window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video. Ignored when *window_id* is set.

window_x

window_y

Set the X and Y window offsets for the created window. They are both set to 0 by default. The values may be ignored by the window manager. Ignored when *window_id* is set.

window_title

Set the window title, if not specified default to the filename specified for the output device. Ignored when *window_id* is set.

For more information about XVideo see <<http://www.x.org/>>.

Examples

- ⊕ Decode, display and encode video input with **ffmpeg** at the same time:

```
ffmpeg -i INPUT OUTPUT -f xv display
```

- ⊕ Decode and display the input video to multiple X11 windows:

```
ffmpeg -i INPUT -f xv normal -vf negate -f xv negated
```

RESAMPLER OPTIONS

The audio resampler supports the following named options.

Options may be set by specifying *-option value* in the FFmpeg tools, *option=value* for the aresample filter, by setting the value explicitly in the "SwrContext" options or using the *libavutil/opt.h* API for programmatic use.

uchl, used_chlayout

Set used input channel layout. Default is unset. This option is only used for special remapping.

isr, in_sample_rate

Set the input sample rate. Default value is 0.

osr, out_sample_rate

Set the output sample rate. Default value is 0.

isf, in_sample_fmt

Specify the input sample format. It is set by default to "none".

osf, out_sample_fmt

Specify the output sample format. It is set by default to "none".

tsf, internal_sample_fmt

Set the internal sample format. Default value is "none". This will automatically be chosen when it is not explicitly set.

ichl, in_chlayout

ochl, out_chlayout

Set the input/output channel layout.

See **the Channel Layout section in the ffmpeg-utils(1) manual** for the required syntax.

clev, center_mix_level

Set the center mix level. It is a value expressed in deciBel, and must be in the interval [-32,32].

slev, surround_mix_level

Set the surround mix level. It is a value expressed in deciBel, and must be in the interval [-32,32].

lfe_mix_level

Set LFE mix into non LFE level. It is used when there is a LFE input but no LFE output. It is a

value expressed in deciBel, and must be in the interval [-32,32].

rmvol, rematrix_volume

Set rematrix volume. Default value is 1.0.

rematrix_maxval

Set maximum output value for rematrixing. This can be used to prevent clipping vs. preventing volume reduction. A value of 1.0 prevents clipping.

flags, swr_flags

Set flags used by the converter. Default value is 0.

It supports the following individual flags:

res force resampling, this flag forces resampling to be used even when the input and output sample rates match.

dither_scale

Set the dither scale. Default value is 1.

dither_method

Set dither method. Default value is 0.

Supported values:

rectangular

select rectangular dither

triangular

select triangular dither

triangular_hp

select triangular dither with high pass

lipshitz

select Lipshitz noise shaping dither.

shibata

select Shibata noise shaping dither.

low_shibata

select low Shibata noise shaping dither.

high_shibata

select high Shibata noise shaping dither.

f_weighted

select f-weighted noise shaping dither

modified_e_weighted

select modified-e-weighted noise shaping dither

improved_e_weighted

select improved-e-weighted noise shaping dither

resampler

Set resampling engine. Default value is swr.

Supported values:

swr select the native SW Resampler; filter options precision and cheby are not applicable in this case.

soxr select the SoX Resampler (where available); compensation, and filter options filter_size, phase_shift, exact_rational, filter_type & kaiser_beta, are not applicable in this case.

filter_size

For swr only, set resampling filter size, default value is 32.

phase_shift

For swr only, set resampling phase shift, default value is 10, and must be in the interval [0,30].

linear_interp

Use linear interpolation when enabled (the default). Disable it if you want to preserve speed instead of quality when exact_rational fails.

exact_rational

For swr only, when enabled, try to use exact phase_count based on input and output sample rate. However, if it is larger than "1 << phase_shift", the phase_count will be "1 << phase_shift" as fallback. Default is enabled.

cutoff

Set cutoff frequency (swr: 6dB point; soxr: 0dB point) ratio; must be a float value between 0 and 1. Default value is 0.97 with swr, and 0.91 with soxr (which, with a sample-rate of 44100, preserves the entire audio band to 20kHz).

precision

For soxr only, the precision in bits to which the resampled signal will be calculated. The default value of 20 (which, with suitable dithering, is appropriate for a destination bit-depth of 16) gives SoX's 'High Quality'; a value of 28 gives SoX's 'Very High Quality'.

cheby

For soxr only, selects passband rolloff none (Chebyshev) & higher-precision approximation for 'irrational' ratios. Default value is 0.

async

For swr only, simple 1 parameter audio sync to timestamps using stretching, squeezing, filling and trimming. Setting this to 1 will enable filling and trimming, larger values represent the maximum amount in samples that the data may be stretched or squeezed for each second. Default value is 0, thus no compensation is applied to make the samples match the audio timestamps.

first_pts

For swr only, assume the first pts should be this value. The time unit is 1 / sample rate. This allows for padding/trimming at the start of stream. By default, no assumption is made about the first frame's expected pts, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with silence if an audio stream starts after the video stream or to trim any samples with a negative pts due to encoder delay.

min_comp

For swr only, set the minimum difference between timestamps and audio data (in seconds) to trigger stretching/squeezing/filling or trimming of the data to make it match the timestamps. The default is that stretching/squeezing/filling and trimming is disabled (**min_comp** = "FLT_MAX").

min_hard_comp

For swr only, set the minimum difference between timestamps and audio data (in seconds) to trigger adding/dropping samples to make it match the timestamps. This option effectively is a threshold to select between hard (trim/fill) and soft (squeeze/stretch) compensation. Note that all compensation is by default disabled through **min_comp**. The default is 0.1.

comp_duration

For swr only, set duration (in seconds) over which data is stretched/squeezed to make it match the

timestamps. Must be a non-negative double float value, default value is 1.0.

max_soft_comp

For swr only, set maximum factor by which data is stretched/squeezed to make it match the timestamps. Must be a non-negative double float value, default value is 0.

matrix_encoding

Select matrixed stereo encoding.

It accepts the following values:

none

select none

dolby

select Dolby

dplii

select Dolby Pro Logic II

Default value is "none".

filter_type

For swr only, select resampling filter type. This only affects resampling operations.

It accepts the following values:

cubic

select cubic

blackman_nuttall

select Blackman Nuttall windowed sinc

kaiser

select Kaiser windowed sinc

kaiser_beta

For swr only, set Kaiser window beta value. Must be a double float value in the interval [2,16], default value is 9.

output_sample_bits

For swr only, set number of used output sample bits for dithering. Must be an integer in the interval [0,64], default value is 0, which means it's not used.

SCALER OPTIONS

The video scaler supports the following named options.

Options may be set by specifying *-option value* in the FFmpeg tools, with a few API-only exceptions noted below. For programmatic use, they can be set explicitly in the "SwsContext" options or through the *libavutil/opt.h* API.

sws_flags

Set the scaler flags. This is also used to set the scaling algorithm. Only a single algorithm should be selected. Default value is **bicubic**.

It accepts the following values:

fast_bilinear

Select fast bilinear scaling algorithm.

bilinear

Select bilinear scaling algorithm.

bicubic

Select bicubic scaling algorithm.

experimental

Select experimental scaling algorithm.

neighbor

Select nearest neighbor rescaling algorithm.

area Select averaging area rescaling algorithm.

bicublin

Select bicubic scaling algorithm for the luma component, bilinear for chroma components.

gauss

Select Gaussian rescaling algorithm.

sinc Select sinc rescaling algorithm.

lanczos

Select Lanczos rescaling algorithm. The default width (alpha) is 3 and can be changed by setting "param0".

spline

Select natural bicubic spline rescaling algorithm.

print_info

Enable printing/debug logging.

accurate_rnd

Enable accurate rounding.

full_chroma_int

Enable full chroma interpolation.

full_chroma_inp

Select full chroma input.

bitexact

Enable bitexact output.

srcw (*API only*)

Set source width.

srch (*API only*)

Set source height.

dstw (*API only*)

Set destination width.

dsth (*API only*)

Set destination height.

src_format (*API only*)

Set source pixel format (must be expressed as an integer).

dst_format (*API only*)

Set destination pixel format (must be expressed as an integer).

src_range (*boolean*)

If value is set to 1, indicates source is full range. Default value is 0, which indicates source is limited range.

dst_range (*boolean*)

If value is set to 1, enable full range for destination. Default value is 0, which enables limited range.

param0, param1

Set scaling algorithm parameters. The specified values are specific of some scaling algorithms and ignored by others. The specified values are floating point number values.

sws_dither

Set the dithering algorithm. Accepts one of the following values. Default value is **auto**.

auto automatic choice

none

no dithering

bayer

bayer dither

ed error diffusion dither

a_dither

arithmetic dither, based using addition

x_dither

arithmetic dither, based using xor (more random/less apparent patterning than a_dither).

alphablend

Set the alpha blending to use when the input has alpha but the output does not. Default value is **none**.

uniform_color

Blend onto a uniform background color

checkerboard

Blend onto a checkerboard

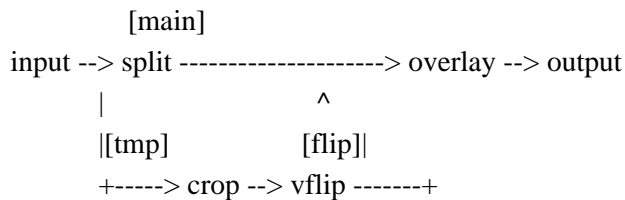
none

No blending

FILTERING INTRODUCTION

Filtering in FFmpeg is enabled through the libavfilter library.

In libavfilter, a filter can have multiple inputs and multiple outputs. To illustrate the sorts of things that are possible, we consider the following filtergraph.



This filtergraph splits the input stream in two streams, then sends one stream through the *crop* filter and the *vflip* filter, before merging it back with the other stream by overlaying it on top. You can use the following command to achieve this:

```
ffmpeg -i INPUT -vf "split [main][tmp]; [tmp] crop=iw:ih/2:0:0, vflip [flip]; [main][flip] overlay=0:H/2" OUTPUT
```

The result will be that the top half of the video is mirrored onto the bottom half of the output video.

Filters in the same linear chain are separated by commas, and distinct linear chains of filters are separated by semicolons. In our example, *crop,vflip* are in one linear chain, *split* and *overlay* are separately in another. The points where the linear chains join are labelled by names enclosed in square brackets. In the example, the *split* filter generates two outputs that are associated to the labels *[main]* and *[tmp]*.

The stream sent to the second output of *split*, labelled as *[tmp]*, is processed through the *crop* filter, which crops away the lower half part of the video, and then vertically flipped. The *overlay* filter takes in input the first unchanged output of the *split* filter (which was labelled as *[main]*), and overlay on its lower half the output generated by the *crop,vflip* filterchain.

Some filters take in input a list of parameters: they are specified after the filter name and an equal sign, and are separated from each other by a colon.

There exist so-called *source filters* that do not have an audio/video input, and *sink filters* that will not have audio/video output.

GRAPH

The *graph2dot* program included in the FFmpeg *tools* directory can be used to parse a filtergraph description and issue a corresponding textual representation in the dot language.

Invoke the command:

```
graph2dot -h
```

to see how to use *graph2dot*.

You can then pass the dot description to the *dot* program (from the graphviz suite of programs) and obtain a graphical representation of the filtergraph.

For example the sequence of commands:

```
echo <GRAPH_DESCRIPTION> | \
tools/graph2dot -o graph.tmp && \
dot -Tpng graph.tmp -o graph.png && \
display graph.png
```

can be used to create and display an image representing the graph described by the *GRAPH_DESCRIPTION* string. Note that this string must be a complete self-contained graph, with its inputs and outputs explicitly defined. For example if your command line is of the form:

```
ffmpeg -i infile -vf scale=640:360 outfile
```

your *GRAPH_DESCRIPTION* string will need to be of the form:

```
nullsrc,scale=640:360,nullsink
```

you may also need to set the *nullsrc* parameters and add a *format* filter in order to simulate a specific input file.

FILTERGRAPH DESCRIPTION

A filtergraph is a directed graph of connected filters. It can contain cycles, and there can be multiple links between a pair of filters. Each link has one input pad on one side connecting it to one filter from which it takes its input, and one output pad on the other side connecting it to one filter accepting its

output.

Each filter in a filtergraph is an instance of a filter class registered in the application, which defines the features and the number of input and output pads of the filter.

A filter with no input pads is called a "source", and a filter with no output pads is called a "sink".

Filtergraph syntax

A filtergraph has a textual representation, which is recognized by the **-filter/-vf/-af** and **-filter_complex** options in **ffmpeg** and **-vf/-af** in **ffplay**, and by the "avfilter_graph_parse_ptr()" function defined in *libavfilter/avfilter.h*.

A filterchain consists of a sequence of connected filters, each one connected to the previous one in the sequence. A filterchain is represented by a list of ","-separated filter descriptions.

A filtergraph consists of a sequence of filterchains. A sequence of filterchains is represented by a list of ";"-separated filterchain descriptions.

A filter is represented by a string of the form:

```
[in_link_1]...[in_link_N]filter_name@id=arguments[out_link_1]...[out_link_M]
```

filter_name is the name of the filter class of which the described filter is an instance of, and has to be the name of one of the filter classes registered in the program optionally followed by "@id". The name of the filter class is optionally followed by a string "=arguments".

arguments is a string which contains the parameters used to initialize the filter instance. It may have one of two forms:

- ⊕ A ':'-separated list of *key=value* pairs.
- ⊕ A ':'-separated list of *value*. In this case, the keys are assumed to be the option names in the order they are declared. E.g. the "fade" filter declares three options in this order -- **type**, **start_frame** and **nb_frames**. Then the parameter list *in:0:30* means that the value *in* is assigned to the option **type**, *0* to **start_frame** and *30* to **nb_frames**.
- ⊕ A ':'-separated list of mixed direct *value* and long *key=value* pairs. The direct *value* must precede the *key=value* pairs, and follow the same constraints order of the previous point. The following *key=value* pairs can be set in any preferred order.

If the option value itself is a list of items (e.g. the "format" filter takes a list of pixel formats), the items

in the list are usually separated by |.

The list of arguments can be quoted using the character ' as initial and ending mark, and the character \ for escaping the characters within the quoted text; otherwise the argument string is considered terminated when the next special character (belonging to the set [=;,)) is encountered.

A special syntax implemented in the **ffmpeg** CLI tool allows loading option values from files. This is done by prepending a slash '/' to the option name, then the supplied value is interpreted as a path from which the actual value is loaded. E.g.

```
ffmpeg -i <INPUT> -vf drawtext=/text=/tmp/some_text <OUTPUT>
```

will load the text to be drawn from */tmp/some_text*. API users wishing to implement a similar feature should use the "avfilter_graph_segment_*(*)" functions together with custom IO code.

The name and arguments of the filter are optionally preceded and followed by a list of link labels. A link label allows one to name a link and associate it to a filter output or input pad. The preceding labels *in_link_1* ... *in_link_N*, are associated to the filter input pads, the following labels *out_link_1* ... *out_link_M*, are associated to the output pads.

When two link labels with the same name are found in the filtergraph, a link between the corresponding input and output pad is created.

If an output pad is not labelled, it is linked by default to the first unlabelled input pad of the next filter in the filterchain. For example in the filterchain

```
nullsrc, split[L1], [L2]overlay, nullsink
```

the split filter instance has two output pads, and the overlay filter instance two input pads. The first output pad of split is labelled "L1", the first input pad of overlay is labelled "L2", and the second output pad of split is linked to the second input pad of overlay, which are both unlabelled.

In a filter description, if the input label of the first filter is not specified, "in" is assumed; if the output label of the last filter is not specified, "out" is assumed.

In a complete filterchain all the unlabelled filter input and output pads must be connected. A filtergraph is considered valid if all the filter input and output pads of all the filterchains are connected.

Libavfilter will automatically insert **scale** filters where format conversion is required. It is possible to specify swscale flags for those automatically inserted scalers by prepending "sws_flags=*flags*;" to the

filtergraph description.

Here is a BNF description of the filtergraph syntax:

```

<NAME>          ::= sequence of alphanumeric characters and ' _ '
<FILTER_NAME>   ::= <NAME>["@"<NAME>]
<LINKLABEL>    ::= "[" <NAME> "]"
<LINKLABELS>   ::= <LINKLABEL> [<LINKLABELS>]
<FILTER_ARGUMENTS> ::= sequence of chars (possibly quoted)
<FILTER>        ::= [<LINKLABELS>] <FILTER_NAME> ["=" <FILTER_ARGUMENTS>] [<LINKLABELS>]
<FILTERCHAIN>  ::= <FILTER> [,<FILTERCHAIN>]
<FILTERGRAPH>  ::= [sws_flags=<flags>;] <FILTERCHAIN> [;<FILTERGRAPH>]

```

Notes on filtergraph escaping

Filtergraph description composition entails several levels of escaping. See **the "Quoting and escaping" section in the ffmpeg-utils(1) manual** for more information about the employed escaping procedure.

A first level escaping affects the content of each filter option value, which may contain the special character ":" used to separate values, or one of the escaping characters "\".

A second level escaping affects the whole filter description, which may contain the escaping characters "\" or the special characters "[,;" used by the filtergraph description.

Finally, when you specify a filtergraph on a shell commandline, you need to perform a third level escaping for the shell special characters contained within it.

For example, consider the following string to be embedded in the **drawtext** filter description **text** value:

```
text=this is a 'string': may contain one, or more, special characters
```

This string contains the "'" special escaping character, and the ":" special character, so it needs to be escaped in this way:

```
text=this is a \'string\': may contain one, or more, special characters
```

A second level of escaping is required when embedding the filter description in a filtergraph description, in order to escape all the filtergraph special characters. Thus the example above becomes:

```
drawtext=text=this is a \\\'string\\\'\\: may contain one\, or more\, special characters
```

(note that in addition to the "\" escaping special characters, also "," needs to be escaped).

Finally an additional level of escaping is needed when writing the filtergraph description in a shell command, which depends on the escaping rules of the adopted shell. For example, assuming that "\" is special and needs to be escaped with another "\", the previous string will finally result in:

```
-vf "drawtext=text=this is a \\\\'string\\\\\\': may contain one\\, or more\\, special characters"
```

TIMELINE EDITING

Some filters support a generic **enable** option. For the filters supporting timeline editing, this option can be set to an expression which is evaluated before sending a frame to the filter. If the evaluation is non-zero, the filter will be enabled, otherwise the frame will be sent unchanged to the next filter in the filtergraph.

The expression accepts the following values:

t timestamp expressed in seconds, NAN if the input timestamp is unknown

n sequential number of the input frame, starting from 0

pos the position in the file of the input frame, NAN if unknown

w

h width and height of the input frame if video

Additionally, these filters support an **enable** command that can be used to re-define the expression.

Like any other filtering option, the **enable** option follows the same rules.

For example, to enable a blur filter (**smartblur**) from 10 seconds to 3 minutes, and a **curves** filter starting at 3 seconds:

```
smartblur = enable='between(t,10,3*60)',
curves = enable='gte(t,3)' : preset=cross_process
```

See "ffmpeg -filters" to view which filters have timeline support.

CHANGING OPTIONS AT RUNTIME WITH A COMMAND

Some options can be changed during the operation of the filter using a command. These options are marked 'T' on the output of **ffmpeg -h filter=<name of filter>**. The name of the command is the name

of the option and the argument is the new value.

OPTIONS FOR FILTERS WITH SEVERAL INPUTS

Some filters with several inputs support a common set of options. These options can only be set by name, not with the short notation.

eof_action

The action to take when EOF is encountered on the secondary input; it accepts one of the following values:

repeat

Repeat the last frame (the default).

endall

End both streams.

pass Pass the main input through.

shortest

If set to 1, force the output to terminate when the shortest input terminates. Default value is 0.

repeatlast

If set to 1, force the filter to extend the last frame of secondary streams until the end of the primary stream. A value of 0 disables this behavior. Default value is 1.

ts_sync_mode

How strictly to sync streams based on secondary input timestamps; it accepts one of the following values:

default

Frame from secondary input with the nearest lower or equal timestamp to the primary input frame.

nearest

Frame from secondary input with the absolute nearest timestamp to the primary input frame.

AUDIO FILTERS

When you configure your FFmpeg build, you can disable any of the existing filters using "--disable-filters". The configure output will show the audio filters included in your build.

Below is a description of the currently available audio filters.

acompressor

A compressor is mainly used to reduce the dynamic range of a signal. Especially modern music is mostly compressed at a high ratio to improve the overall loudness. It's done to get the highest attention of a listener, "fatten" the sound and bring more "power" to the track. If a signal is compressed too much it may sound dull or "dead" afterwards or it may start to "pump" (which could be a powerful effect but can also destroy a track completely). The right compression is the key to reach a professional sound and is the high art of mixing and mastering. Because of its complex settings it may take a long time to get the right feeling for this kind of effect.

Compression is done by detecting the volume above a chosen level "threshold" and dividing it by the factor set with "ratio". So if you set the threshold to -12dB and your signal reaches -6dB a ratio of 2:1 will result in a signal at -9dB. Because an exact manipulation of the signal would cause distortion of the waveform the reduction can be levelled over the time. This is done by setting "Attack" and "Release". "attack" determines how long the signal has to rise above the threshold before any reduction will occur and "release" sets the time the signal has to fall below the threshold to reduce the reduction again. Shorter signals than the chosen attack time will be left untouched. The overall reduction of the signal can be made up afterwards with the "makeup" setting. So compressing the peaks of a signal about 6dB and raising the makeup to this level results in a signal twice as loud than the source. To gain a softer entry in the compression the "knee" flattens the hard edge at the threshold in the range of the chosen decibels.

The filter accepts the following options:

level_in

Set input gain. Default is 1. Range is between 0.015625 and 64.

mode

Set mode of compressor operation. Can be "upward" or "downward". Default is "downward".

threshold

If a signal of stream rises above this level it will affect the gain reduction. By default it is 0.125. Range is between 0.00097563 and 1.

ratio

Set a ratio by which the signal is reduced. 1:2 means that if the level rose 4dB above the threshold, it will be only 2dB above after the reduction. Default is 2. Range is between 1 and 20.

attack

Amount of milliseconds the signal has to rise above the threshold before gain reduction starts. Default is 20. Range is between 0.01 and 2000.

release

Amount of milliseconds the signal has to fall below the threshold before reduction is decreased again. Default is 250. Range is between 0.01 and 9000.

makeup

Set the amount by how much signal will be amplified after processing. Default is 1. Range is from 1 to 64.

knee

Curve the sharp knee around the threshold to enter gain reduction more softly. Default is 2.82843. Range is between 1 and 8.

link Choose if the "average" level between all channels of input stream or the louder("maximum") channel of input stream affects the reduction. Default is "average".

detection

Should the exact signal be taken in case of "peak" or an RMS one in case of "rms". Default is "rms" which is mostly smoother.

mix How much to use compressed signal in output. Default is 1. Range is between 0 and 1.

Commands

This filter supports the all above options as **commands**.

acontrast

Simple audio dynamic range compression/expansion filter.

The filter accepts the following options:

contrast

Set contrast. Default is 33. Allowed range is between 0 and 100.

acopy

Copy the input audio source unchanged to the output. This is mainly useful for testing purposes.

acrossfade

Apply cross fade from one input audio stream to another input audio stream. The cross fade is applied for specified duration near the end of first stream.

The filter accepts the following options:

nb_samples, ns

Specify the number of samples for which the cross fade effect has to last. At the end of the cross fade effect the first input audio will be completely silent. Default is 44100.

duration, d

Specify the duration of the cross fade effect. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. By default the duration is determined by *nb_samples*. If set this option is used instead of *nb_samples*.

overlap, o

Should first stream end overlap with second stream start. Default is enabled.

curve1

Set curve for cross fade transition for first stream.

curve2

Set curve for cross fade transition for second stream.

For description of available curve types see **afade** filter description.

Examples

- ⊕ Cross fade from one input to another:

```
ffmpeg -i first.flac -i second.flac -filter_complex acrossofade=d=10:c1=exp:c2=exp output.flac
```

- ⊕ Cross fade from one input to another but without overlapping:

```
ffmpeg -i first.flac -i second.flac -filter_complex acrossofade=d=10:o=0:c1=exp:c2=exp output.flac
```

acrossover

Split audio stream into several bands.

This filter splits audio stream into two or more frequency ranges. Summing all streams back will give flat output.

The filter accepts the following options:

split Set split frequencies. Those must be positive and increasing.

order

Set filter order for each band split. This controls filter roll-off or steepness of filter transfer function. Available values are:

2nd 12 dB per octave.

4th 24 dB per octave.

6th 36 dB per octave.

8th 48 dB per octave.

10th

60 dB per octave.

12th

72 dB per octave.

14th

84 dB per octave.

16th

96 dB per octave.

18th

108 dB per octave.

20th

120 dB per octave.

Default is *4th*.

level

Set input gain level. Allowed range is from 0 to 1. Default value is 1.

gains

Set output gain for each band. Default value is 1 for all bands.

precision

Set which precision to use when processing samples.

auto Auto pick internal sample format depending on other filters.

float

Always use single-floating point precision sample format.

double

Always use double-floating point precision sample format.

Default value is "auto".

Examples

- ⊕ Split input audio stream into two bands (low and high) with split frequency of 1500 Hz, each band will be in separate stream:

```
ffmpeg -i in.flac -filter_complex 'acrossover=split=1500[LOW][HIGH]' -map '[LOW]' low.wav -map '[HIGH]' high.wav
```

- ⊕ Same as above, but with higher filter order:

```
ffmpeg -i in.flac -filter_complex 'acrossover=split=1500:order=8th[LOW][HIGH]' -map '[LOW]' low.wav -map '[HIGH]' high.wav
```

- ⊕ Same as above, but also with additional middle band (frequencies between 1500 and 8000):

```
ffmpeg -i in.flac -filter_complex 'acrossover=split=1500 8000:order=8th[LOW][MID][HIGH]' -map '[LOW]' low.wav -map '[MID]' mid.wav -map '[HIGH]' high.wav
```

acrusher

Reduce audio bit resolution.

This filter is bit crusher with enhanced functionality. A bit crusher is used to audibly reduce number of bits an audio signal is sampled with. This doesn't change the bit depth at all, it just produces the effect. Material reduced in bit depth sounds more harsh and "digital". This filter is able to even round to continuous values instead of discrete bit depths. Additionally it has a D/C offset which results in different crushing of the lower and the upper half of the signal. An Anti-Aliasing setting is able to produce "softer" crushing sounds.

Another feature of this filter is the logarithmic mode. This setting switches from linear distances between bits to logarithmic ones. The result is a much more "natural" sounding crusher which doesn't gate low signals for example. The human ear has a logarithmic perception, so this kind of crushing is much more pleasant. Logarithmic crushing is also able to get anti-aliased.

The filter accepts the following options:

level_in

Set level in.

level_out

Set level out.

bits Set bit reduction.

mix Set mixing amount.

mode

Can be linear: "lin" or logarithmic: "log".

dc Set DC.

aa Set anti-aliasing.

samples

Set sample reduction.

lfo Enable LFO. By default disabled.

lforange

Set LFO range.

lforate

Set LFO rate.

Commands

This filter supports the all above options as **commands**.

acue

Delay audio filtering until a given wallclock timestamp. See the **cue** filter.

adeclick

Remove impulsive noise from input audio.

Samples detected as impulsive noise are replaced by interpolated samples using autoregressive modelling.

window, w

Set window size, in milliseconds. Allowed range is from 10 to 100. Default value is 55 milliseconds. This sets size of window which will be processed at once.

overlap, o

Set window overlap, in percentage of window size. Allowed range is from 50 to 95. Default value is 75 percent. Setting this to a very high value increases impulsive noise removal but makes whole process much slower.

arorder, a

Set autoregression order, in percentage of window size. Allowed range is from 0 to 25. Default value is 2 percent. This option also controls quality of interpolated samples using neighbour good samples.

threshold, t

Set threshold value. Allowed range is from 1 to 100. Default value is 2. This controls the strength of impulsive noise which is going to be removed. The lower value, the more samples will be detected as impulsive noise.

burst, b

Set burst fusion, in percentage of window size. Allowed range is 0 to 10. Default value is 2. If any two samples detected as noise are spaced less than this value then any sample between those two samples will be also detected as noise.

method, m

Set overlap method.

It accepts the following values:

add, a

Select overlap-add method. Even not interpolated samples are slightly changed with this method.

save, s

Select overlap-save method. Not interpolated samples remain unchanged.

Default value is "a".

adeclip

Remove clipped samples from input audio.

Samples detected as clipped are replaced by interpolated samples using autoregressive modelling.

window, w

Set window size, in milliseconds. Allowed range is from 10 to 100. Default value is 55 milliseconds. This sets size of window which will be processed at once.

overlap, o

Set window overlap, in percentage of window size. Allowed range is from 50 to 95. Default value is 75 percent.

arorder, a

Set autoregression order, in percentage of window size. Allowed range is from 0 to 25. Default value is 8 percent. This option also controls quality of interpolated samples using neighbour good samples.

threshold, t

Set threshold value. Allowed range is from 1 to 100. Default value is 10. Higher values make clip detection less aggressive.

hsize, n

Set size of histogram used to detect clips. Allowed range is from 100 to 9999. Default value is 1000. Higher values make clip detection less aggressive.

method, m

Set overlap method.

It accepts the following values:

add, a

Select overlap-add method. Even not interpolated samples are slightly changed with this method.

save, s

Select overlap-save method. Not interpolated samples remain unchanged.

Default value is "a".

adecorrelate

Apply decorrelation to input audio stream.

The filter accepts the following options:

stages

Set decorrelation stages of filtering. Allowed range is from 1 to 16. Default value is 6.

seed

Set random seed used for setting delay in samples across channels.

adelay

Delay one or more audio channels.

Samples in delayed channel are filled with silence.

The filter accepts the following option:

delays

Set list of delays in milliseconds for each channel separated by '|'. Unused delays will be silently ignored. If number of given delays is smaller than number of channels all remaining channels will not be delayed. If you want to delay exact number of samples, append 'S' to number. If you want instead to delay in seconds, append 's' to number.

all Use last set delay for all remaining channels. By default is disabled. This option if enabled changes how option "delays" is interpreted.

Examples

- ⊕ Delay first channel by 1.5 seconds, the third channel by 0.5 seconds and leave the second channel (and any other channels that may be present) unchanged.

```
adelay=1500|0|500
```

- ⊕ Delay second channel by 500 samples, the third channel by 700 samples and leave the first

channel (and any other channels that may be present) unchanged.

```
adelay=0|500S|700S
```

⊕ Delay all channels by same number of samples:

```
adelay=delays=64S:all=1
```

adenorm

Remedy denormals in audio by adding extremely low-level noise.

This filter shall be placed before any filter that can produce denormals.

A description of the accepted parameters follows.

level

Set level of added noise in dB. Default is "-351". Allowed range is from -451 to -90.

type Set type of added noise.

dc Add DC signal.

ac Add AC signal.

square

Add square signal.

pulse

Add pulse signal.

Default is "dc".

Commands

This filter supports the all above options as **commands**.

aderivative, aintegral

Compute derivative/integral of audio stream.

Applying both filters one after another produces original audio.

adrc

Apply spectral dynamic range controller filter to input audio stream.

A description of the accepted options follows.

transfer

Set the transfer expression.

The expression can contain the following constants:

ch current channel number

sn current sample number

nb_channels

number of channels

t timestamp expressed in seconds

sr sample rate

p current frequency power value, in dB

f current frequency in Hz

Default value is "p".

attack

Set the attack in milliseconds. Default is 50 milliseconds. Allowed range is from 1 to 1000 milliseconds.

release

Set the release in milliseconds. Default is 100 milliseconds. Allowed range is from 5 to 2000 milliseconds.

channels

Set which channels to filter, by default "all" channels in audio stream are filtered.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Apply spectral compression to all frequencies with threshold of -50 dB and 1:6 ratio:

```
adrc=transfer='if(gt(p,-50),-50+(p-(-50))/6,p)':attack=50:release=100
```

- ⊕ Similar to above but with 1:2 ratio and filtering only front center channel:

```
adrc=transfer='if(gt(p,-50),-50+(p-(-50))/2,p)':attack=50:release=100:channels=FC
```

- ⊕ Apply spectral noise gate to all frequencies with threshold of -85 dB and with short attack time and short release time:

```
adrc=transfer='if(lte(p,-85),p-800,p)':attack=1:release=5
```

- ⊕ Apply spectral expansion to all frequencies with threshold of -10 dB and 1:2 ratio:

```
adrc=transfer='if(lt(p,-10),-10+(p-(-10))*2,p)':attack=50:release=100
```

- ⊕ Apply limiter to max -60 dB to all frequencies, with attack of 2 ms and release of 10 ms:

```
adrc=transfer='min(p,-60)':attack=2:release=10
```

adynamicequalizer

Apply dynamic equalization to input audio stream.

A description of the accepted options follows.

threshold

Set the detection threshold used to trigger equalization. Threshold detection is using bandpass filter. Default value is 0. Allowed range is from 0 to 100.

dfrequency

Set the detection frequency in Hz used for bandpass filter used to trigger equalization. Default value is 1000 Hz. Allowed range is between 2 and 1000000 Hz.

dqfactor

Set the detection resonance factor for bandpass filter used to trigger equalization. Default value is

1. Allowed range is from 0.001 to 1000.

tfrequency

Set the target frequency of equalization filter. Default value is 1000 Hz. Allowed range is between 2 and 1000000 Hz.

tqfactor

Set the target resonance factor for target equalization filter. Default value is 1. Allowed range is from 0.001 to 1000.

attack

Set the amount of milliseconds the signal from detection has to rise above the detection threshold before equalization starts. Default is 20. Allowed range is between 1 and 2000.

release

Set the amount of milliseconds the signal from detection has to fall below the detection threshold before equalization ends. Default is 200. Allowed range is between 1 and 2000.

ratio

Set the ratio by which the equalization gain is raised. Default is 1. Allowed range is between 0 and 30.

makeup

Set the makeup offset by which the equalization gain is raised. Default is 0. Allowed range is between 0 and 100.

range

Set the max allowed cut/boost amount. Default is 50. Allowed range is from 1 to 200.

mode

Set the mode of filter operation, can be one of the following:

listen

Output only isolated bandpass signal.

cut Cut frequencies above detection threshold.

boost

Boost frequencies bellow detection threshold.

Default mode is **cut**.

tftype

Set the type of target filter, can be one of the following:

bell

lowshelf

highshelf

Default type is **bell**.

direction

Set processing direction relative to threshold.

downward

Boost/Cut if threshold is higher/lower than detected volume.

upward

Boost/Cut if threshold is lower/higher than detected volume.

Default direction is **downward**.

auto Automatically gather threshold from detection filter. By default is **disabled**. This option is useful to detect threshold in certain time frame of input audio stream, in such case option value is changed at runtime.

Available values are:

disabled

Disable using automatically gathered threshold value.

off Stop picking threshold value.

on Start picking threshold value.

Commands

This filter supports the all above options as **commands**.

adynamicsmooth

Apply dynamic smoothing to input audio stream.

A description of the accepted options follows.

sensitivity

Set an amount of sensitivity to frequency fluctuations. Default is 2. Allowed range is from 0 to 1e+06.

basefreq

Set a base frequency for smoothing. Default value is 22050. Allowed range is from 2 to 1e+06.

Commands

This filter supports the all above options as **commands**.

aecho

Apply echoing to the input audio.

Echoes are reflected sound and can occur naturally amongst mountains (and sometimes large buildings) when talking or shouting; digital echo effects emulate this behaviour and are often used to help fill out the sound of a single instrument or vocal. The time difference between the original signal and the reflection is the "delay", and the loudness of the reflected signal is the "decay". Multiple echoes can have different delays and decays.

A description of the accepted parameters follows.

in_gain

Set input gain of reflected signal. Default is 0.6.

out_gain

Set output gain of reflected signal. Default is 0.3.

delays

Set list of time intervals in milliseconds between original signal and reflections separated by '|'. Allowed range for each "delay" is "(0 - 90000.0]". Default is 1000.

decays

Set list of loudness of reflected signals separated by '|'. Allowed range for each "decay" is "(0 - 1.0]". Default is 0.5.

Examples

- ⊕ Make it sound as if there are twice as many instruments as are actually playing:

```
aecho=0.8:0.88:60:0.4
```

- ⊕ If delay is very short, then it sounds like a (metallic) robot playing music:

```
aecho=0.8:0.88:6:0.4
```

- ⊕ A longer delay will sound like an open air concert in the mountains:

```
aecho=0.8:0.9:1000:0.3
```

- ⊕ Same as above but with one more mountain:

```
aecho=0.8:0.9:1000|1800:0.3|0.25
```

aemphasis

Audio emphasis filter creates or restores material directly taken from LPs or emphasized CDs with different filter curves. E.g. to store music on vinyl the signal has to be altered by a filter first to even out the disadvantages of this recording medium. Once the material is played back the inverse filter has to be applied to restore the distortion of the frequency response.

The filter accepts the following options:

level_in

Set input gain.

level_out

Set output gain.

mode

Set filter mode. For restoring material use "reproduction" mode, otherwise use "production" mode. Default is "reproduction" mode.

type Set filter type. Selects medium. Can be one of the following:

col select Columbia.

emi select EMI.

bsi select BSI (78RPM).

riaa select RIAA.

cd select Compact Disc (CD).

50fm

select 50Xs (FM).

75fm

select 75Xs (FM).

50kf

select 50Xs (FM-KF).

75kf

select 75Xs (FM-KF).

Commands

This filter supports the all above options as **commands**.

aeval

Modify an audio signal according to the specified expressions.

This filter accepts one or more expressions (one for each channel), which are evaluated and used to modify a corresponding audio signal.

It accepts the following parameters:

exprs

Set the '|'-separated expressions list for each separate channel. If the number of input channels is greater than the number of expressions, the last specified expression is used for the remaining output channels.

channel_layout, c

Set output channel layout. If not specified, the channel layout is specified by the number of expressions. If set to **same**, it will use by default the same input channel layout.

Each expression in *exprs* can contain the following constants and functions:

- ch** channel number of the current expression
- n** number of the evaluated sample, starting from 0
- s** sample rate
- t** time of the evaluated sample expressed in seconds

nb_in_channels

nb_out_channels

input and output number of channels

val(CH)

the value of input channel with number *CH*

Note: this filter is slow. For faster processing you should use a dedicated filter.

Examples

- ⊕ Half volume:

```
aeval=val(ch)/2:c=same
```

- ⊕ Invert phase of the second channel:

```
aeval=val(0)|-val(1)
```

aexciter

An exciter is used to produce high sound that is not present in the original signal. This is done by creating harmonic distortions of the signal which are restricted in range and added to the original signal. An Exciter raises the upper end of an audio signal without simply raising the higher frequencies like an equalizer would do to create a more "crisp" or "brilliant" sound.

The filter accepts the following options:

level_in

Set input level prior processing of signal. Allowed range is from 0 to 64. Default value is 1.

level_out

Set output level after processing of signal. Allowed range is from 0 to 64. Default value is 1.

amount

Set the amount of harmonics added to original signal. Allowed range is from 0 to 64. Default value is 1.

drive

Set the amount of newly created harmonics. Allowed range is from 0.1 to 10. Default value is 8.5.

blend

Set the octave of newly created harmonics. Allowed range is from -10 to 10. Default value is 0.

freq Set the lower frequency limit of producing harmonics in Hz. Allowed range is from 2000 to 12000 Hz. Default is 7500 Hz.

ceil Set the upper frequency limit of producing harmonics. Allowed range is from 9999 to 20000 Hz. If value is lower than 10000 Hz no limit is applied.

listen

Mute the original signal and output only added harmonics. By default is disabled.

Commands

This filter supports the all above options as **commands**.

afade

Apply fade-in/out effect to input audio.

A description of the accepted parameters follows.

type, t

Specify the effect type, can be either "in" for fade-in, or "out" for a fade-out effect. Default is "in".

start_sample, ss

Specify the number of the start sample for starting to apply the fade effect. Default is 0.

nb_samples, ns

Specify the number of samples for which the fade effect has to last. At the end of the fade-in

effect the output audio will have the same volume as the input audio, at the end of the fade-out transition the output audio will be silence. Default is 44100.

start_time, st

Specify the start time of the fade effect. Default is 0. The value must be specified as a time duration; see **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. If set this option is used instead of *start_sample*.

duration, d

Specify the duration of the fade effect. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. At the end of the fade-in effect the output audio will have the same volume as the input audio, at the end of the fade-out transition the output audio will be silence. By default the duration is determined by *nb_samples*. If set this option is used instead of *nb_samples*.

curve

Set curve for fade transition.

It accepts the following values:

tri select triangular, linear slope (default)

qsin select quarter of sine wave

hsin select half of sine wave

esin select exponential sine wave

log select logarithmic

ipar select inverted parabola

qua select quadratic

cub select cubic

squ select square root

cbr select cubic root

par select parabola

exp select exponential

iqsin
select inverted quarter of sine wave

ihsin
select inverted half of sine wave

dese
select double-exponential seat

desi select double-exponential sigmoid

losi select logistic sigmoid

sinc select sine cardinal function

isinc
select inverted sine cardinal function

nofade
no fade applied

silence
Set the initial gain for fade-in or final gain for fade-out. Default value is 0.0.

unity
Set the initial gain for fade-out or final gain for fade-in. Default value is 1.0.

Commands

This filter supports the all above options as **commands**.

Examples

⊕ Fade in first 15 seconds of audio:

```
afade=t=in:ss=0:d=15
```

- ⊕ Fade out last 25 seconds of a 900 seconds audio:

```
afade=t=out:st=875:d=25
```

afftdn

Denoise audio samples with FFT.

A description of the accepted parameters follows.

noise_reduction, nr

Set the noise reduction in dB, allowed range is 0.01 to 97. Default value is 12 dB.

noise_floor, nf

Set the noise floor in dB, allowed range is -80 to -20. Default value is -50 dB.

noise_type, nt

Set the noise type.

It accepts the following values:

white, w

Select white noise.

vinyl, v

Select vinyl noise.

shellac, s

Select shellac noise.

custom, c

Select custom noise, defined in "bn" option.

Default value is white noise.

band_noise, bn

Set custom band noise profile for every one of 15 bands. Bands are separated by ' ' or '|'.

residual_floor, rf

Set the residual floor in dB, allowed range is -80 to -20. Default value is -38 dB.

track_noise, tn

Enable noise floor tracking. By default is disabled. With this enabled, noise floor is automatically adjusted.

track_residual, tr

Enable residual tracking. By default is disabled.

output_mode, om

Set the output mode.

It accepts the following values:

input, i

Pass input unchanged.

output, o

Pass noise filtered out.

noise, n

Pass only noise.

Default value is *output*.

adaptivity, ad

Set the adaptivity factor, used how fast to adapt gains adjustments per each frequency bin. Value *0* enables instant adaptation, while higher values react much slower. Allowed range is from *0* to *1*. Default value is *0.5*.

floor_offset, fo

Set the noise floor offset factor. This option is used to adjust offset applied to measured noise floor. It is only effective when noise floor tracking is enabled. Allowed range is from *-2.0* to *2.0*. Default value is *1.0*.

noise_link, nl

Set the noise link used for multichannel audio.

It accepts the following values:

none

Use unchanged channel's noise floor.

min Use measured min noise floor of all channels.

max Use measured max noise floor of all channels.

average

Use measured average noise floor of all channels.

Default value is *min*.

band_multiplier, bm

Set the band multiplier factor, used how much to spread bands across frequency bins. Allowed range is from 0.2 to 5. Default value is 1.25.

sample_noise, sn

Toggle capturing and measurement of noise profile from input audio.

It accepts the following values:

start, begin

Start sample noise capture.

stop, end

Stop sample noise capture and measure new noise band profile.

Default value is "none".

gain_smooth, gs

Set gain smooth spatial radius, used to smooth gains applied to each frequency bin. Useful to reduce random music noise artefacts. Higher values increases smoothing of gains. Allowed range is from 0 to 50. Default value is 0.

Commands

This filter supports the some above mentioned options as **commands**.

Examples

- ⊕ Reduce white noise by 10dB, and use previously measured noise floor of -40dB:

```
afftdn=nr=10:nf=-40
```

- ⊕ Reduce white noise by 10dB, also set initial noise floor to -80dB and enable automatic tracking of noise floor so noise floor will gradually change during processing:

```
afftdn=nr=10:nf=-80:tn=1
```

- ⊕ Reduce noise by 20dB, using noise floor of -40dB and using commands to take noise profile of first 0.4 seconds of input audio:

```
asendcmd=0.0 afftdn sn start,asendcmd=0.4 afftdn sn stop,afftdn=nr=20:nf=-40
```

afftfilt

Apply arbitrary expressions to samples in frequency domain.

real Set frequency domain real expression for each separate channel separated by '|'. Default is "re". If the number of input channels is greater than the number of expressions, the last specified expression is used for the remaining output channels.

imag

Set frequency domain imaginary expression for each separate channel separated by '|'. Default is "im".

Each expression in *real* and *imag* can contain the following constants and functions:

sr sample rate

b current frequency bin number

nb number of available bins

ch channel number of the current expression

chs number of channels

pts current frame pts

re current real part of frequency bin of current channel

im current imaginary part of frequency bin of current channel

real(b, ch)

Return the value of real part of frequency bin at location (*bin,channel*)

imag(b, ch)

Return the value of imaginary part of frequency bin at location (*bin,channel*)

win_size

Set window size. Allowed range is from 16 to 131072. Default is 4096

win_func

Set window function.

It accepts the following values:

rect
bartlett
hann, hanning
hamming
blackman
welch
flattop
bharris
bnutall
bhann
sine
nutall
lanczos
gauss
tukey
dolph
cauchy
parzen
poisson
bohman
kaiser

Default is "hann".

overlap

Set window overlap. If set to 1, the recommended overlap for selected window function will be picked. Default is 0.75.

Examples

- ⊕ Leave almost only low frequencies in audio:

```
afftfilt="'real=re * (1-clip((b/nb)*b,0,1))':imag='im * (1-clip((b/nb)*b,0,1))'"
```

- ⊕ Apply robotize effect:

```
afftfilt="real='hypot(re,im)*sin(0)':imag='hypot(re,im)*cos(0)':win_size=512:overlap=0.75"
```

- ⊕ Apply whisper effect:

```
afftfilt="real='hypot(re,im)*cos((random(0)*2-1)*2*3.14)':imag='hypot(re,im)*sin((random(1)*2-1)*2*3.14)"
```

- ⊕ Apply phase shift:

```
afftfilt="real=re*cos(1)-im*sin(1):imag=re*sin(1)+im*cos(1)"
```

afir

Apply an arbitrary Finite Impulse Response filter.

This filter is designed for applying long FIR filters, up to 60 seconds long.

It can be used as component for digital crossover filters, room equalization, cross talk cancellation, wavefield synthesis, auralization, ambiophonics, ambisonics and spatialization.

This filter uses the streams higher than first one as FIR coefficients. If the non-first stream holds a single channel, it will be used for all input channels in the first stream, otherwise the number of channels in the non-first stream must be same as the number of channels in the first stream.

It accepts the following parameters:

dry Set dry gain. This sets input gain.

wet Set wet gain. This sets final output gain.

length

Set Impulse Response filter length. Default is 1, which means whole IR is processed.

gtype

Enable applying gain measured from power of IR.

Set which approach to use for auto gain measurement.

none

Do not apply any gain.

peak

select peak gain, very conservative approach. This is default value.

dc select DC gain, limited application.

gn select gain to noise approach, this is most popular one.

ac select AC gain.

rms select RMS gain.

irgain

Set gain to be applied to IR coefficients before filtering. Allowed range is 0 to 1. This gain is applied after any gain applied with *gtype* option.

irfmt

Set format of IR stream. Can be "mono" or "input". Default is "input".

maxir

Set max allowed Impulse Response filter duration in seconds. Default is 30 seconds. Allowed range is 0.1 to 60 seconds.

response

Show IR frequency response, magnitude(magenta), phase(green) and group delay(yellow) in additional video stream. By default it is disabled.

channel

Set for which IR channel to display frequency response. By default is first channel displayed. This option is used only when *response* is enabled.

size Set video stream size. This option is used only when *response* is enabled.

rate Set video stream frame rate. This option is used only when *response* is enabled.

minp

Set minimal partition size used for convolution. Default is *8192*. Allowed range is from *1* to *65536*. Lower values decreases latency at cost of higher CPU usage.

maxp

Set maximal partition size used for convolution. Default is *8192*. Allowed range is from *8* to *65536*. Lower values may increase CPU usage.

nbirs

Set number of input impulse responses streams which will be switchable at runtime. Allowed range is from *1* to *32*. Default is *1*.

ir Set IR stream which will be used for convolution, starting from *0*, should always be lower than supplied value by "nbirs" option. Default is *0*. This option can be changed at runtime via **commands**.

precision

Set which precision to use when processing samples.

auto Auto pick internal sample format depending on other filters.

float

Always use single-floating point precision sample format.

double

Always use double-floating point precision sample format.

Default value is auto.

Examples

- ⊕ Apply reverb to stream using mono IR file as second input, complete command using ffmpeg:

```
ffmpeg -i input.wav -i middle_tunnel_1way_mono.wav -lavfi afir output.wav
```

- ⊕ Apply true stereo processing given input stereo stream, and two stereo impulse responses for left and right channel, the impulse response files are files with names *l_ir.wav* and *r_ir.wav*:

```
"pan=4C|c0=FL|c1=FL|c2=FR|c3=FR[a];amovie=l_ir.wav[LIR];amovie=r_ir.wav[RIR];[LIR][RIR]amerge[i
```

aformat

Set output format constraints for the input audio. The framework will negotiate the most appropriate format to minimize conversions.

It accepts the following parameters:

sample_fmts, f

A '|' -separated list of requested sample formats.

sample_rates, r

A '|' -separated list of requested sample rates.

channel_layouts, cl

A '|' -separated list of requested channel layouts.

See **the Channel Layout section in the ffmpeg-utils(1) manual** for the required syntax.

If a parameter is omitted, all values are allowed.

Force the output to either unsigned 8-bit or signed 16-bit stereo

```
aformat=sample_fmts=u8|s16:channel_layouts=stereo
```

afreqshift

Apply frequency shift to input audio samples.

The filter accepts the following options:

shift

Specify frequency shift. Allowed range is -INT_MAX to INT_MAX. Default value is 0.0.

level

Set output gain applied to final output. Allowed range is from 0.0 to 1.0. Default value is 1.0.

order

Set filter order used for filtering. Allowed range is from 1 to 16. Default value is 8.

Commands

This filter supports the all above options as **commands**.

afwtdn

Reduce broadband noise from input samples using Wavelets.

A description of the accepted options follows.

sigma

Set the noise sigma, allowed range is from 0 to 1. Default value is 0. This option controls strength of denoising applied to input samples. Most useful way to set this option is via decibels, eg. -45dB.

levels

Set the number of wavelet levels of decomposition. Allowed range is from 1 to 12. Default value is 10. Setting this too low make denoising performance very poor.

wavet

Set wavelet type for decomposition of input frame. They are sorted by number of coefficients, from lowest to highest. More coefficients means worse filtering speed, but overall better quality. Available wavelets are:

sym2

sym4

rbior68

deb10

sym10

coif5

bl3

percent

Set percent of full denoising. Allowed range is from 0 to 100 percent. Default value is 85 percent or partial denoising.

profile

If enabled, first input frame will be used as noise profile. If first frame samples contain non-noise performance will be very poor.

adaptive

If enabled, input frames are analyzed for presence of noise. If noise is detected with high possibility then input frame profile will be used for processing following frames, until new noise frame is detected.

samples

Set size of single frame in number of samples. Allowed range is from 512 to 65536. Default frame size is 8192 samples.

softness

Set softness applied inside thresholding function. Allowed range is from 0 to 10. Default softness is 1.

Commands

This filter supports the all above options as **commands**.

agate

A gate is mainly used to reduce lower parts of a signal. This kind of signal processing reduces disturbing noise between useful signals.

Gating is done by detecting the volume below a chosen level *threshold* and dividing it by the factor set with *ratio*. The bottom of the noise floor is set via *range*. Because an exact manipulation of the signal would cause distortion of the waveform the reduction can be levelled over time. This is done by setting *attack* and *release*.

attack determines how long the signal has to fall below the threshold before any reduction will occur and *release* sets the time the signal has to rise above the threshold to reduce the reduction again. Shorter signals than the chosen attack time will be left untouched.

level_in

Set input level before filtering. Default is 1. Allowed range is from 0.015625 to 64.

mode

Set the mode of operation. Can be "upward" or "downward". Default is "downward". If set to "upward" mode, higher parts of signal will be amplified, expanding dynamic range in upward direction. Otherwise, in case of "downward" lower parts of signal will be reduced.

range

Set the level of gain reduction when the signal is below the threshold. Default is 0.06125. Allowed range is from 0 to 1. Setting this to 0 disables reduction and then filter behaves like expander.

threshold

If a signal rises above this level the gain reduction is released. Default is 0.125. Allowed range is from 0 to 1.

ratio

Set a ratio by which the signal is reduced. Default is 2. Allowed range is from 1 to 9000.

attack

Amount of milliseconds the signal has to rise above the threshold before gain reduction stops. Default is 20 milliseconds. Allowed range is from 0.01 to 9000.

release

Amount of milliseconds the signal has to fall below the threshold before the reduction is increased again. Default is 250 milliseconds. Allowed range is from 0.01 to 9000.

makeup

Set amount of amplification of signal after processing. Default is 1. Allowed range is from 1 to 64.

knee

Curve the sharp knee around the threshold to enter gain reduction more softly. Default is 2.828427125. Allowed range is from 1 to 8.

detection

Choose if exact signal should be taken for detection or an RMS like one. Default is "rms". Can be "peak" or "rms".

link Choose if the average level between all channels or the louder channel affects the reduction. Default is "average". Can be "average" or "maximum".

Commands

This filter supports the all above options as **commands**.

aiir

Apply an arbitrary Infinite Impulse Response filter.

It accepts the following parameters:

zeros, z

Set B/numerator/zeros/reflection coefficients.

poles, p

Set A/denominator/poles/ladder coefficients.

gains, k

Set channels gains.

dry_gain

Set input gain.

wet_gain

Set output gain.

format, f

Set coefficients format.

ll lattice-ladder function

sf analog transfer function

tf digital transfer function

zp Z-plane zeros/poles, cartesian (default)

pr Z-plane zeros/poles, polar radians

pd Z-plane zeros/poles, polar degrees

sp S-plane zeros/poles

process, r

Set type of processing.

d direct processing

s serial processing

p parallel processing

precision, e

Set filtering precision.

dbl double-precision floating-point (default)

flt single-precision floating-point

i32 32-bit integers

i16 16-bit integers

normalize, n

Normalize filter coefficients, by default is enabled. Enabling it will normalize magnitude response at DC to 0dB.

mix How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

response

Show IR frequency response, magnitude(magenta), phase(green) and group delay(yellow) in additional video stream. By default it is disabled.

channel

Set for which IR channel to display frequency response. By default is first channel displayed. This option is used only when *response* is enabled.

size Set video stream size. This option is used only when *response* is enabled.

Coefficients in "tf" and "sf" format are separated by spaces and are in ascending order.

Coefficients in "zp" format are separated by spaces and order of coefficients doesn't matter.

Coefficients in "zp" format are complex numbers with *i* imaginary unit.

Different coefficients and gains can be provided for every channel, in such case use '|' to separate coefficients or gains. Last provided coefficients will be used for all remaining channels.

Examples

- ⊕ Apply 2 pole elliptic notch at around 5000Hz for 48000 Hz sample rate:

```
aiir=k=1:z=7.957584807809675810E-1 -2.575128568908332300 3.674839853930788710 -2.575128752897
```

- ⊕ Same as above but in "zp" format:

```
aiir=k=0.79575848078096756:z=0.80918701+0.58773007i 0.80918701-0.58773007i 0.80884700+0.587840
```

- ⊕ Apply 3-rd order analog normalized Butterworth low-pass filter, using analog transfer function format:

```
aiir=z=1.3057 0 0 0:p=1.3057 2.3892 2.1860 1:f=sf:r=d
```

alimiter

The limiter prevents an input signal from rising over a desired threshold. This limiter uses lookahead technology to prevent your signal from distorting. It means that there is a small delay after the signal is processed. Keep in mind that the delay it produces is the attack time you set.

The filter accepts the following options:

level_in

Set input gain. Default is 1.

level_out

Set output gain. Default is 1.

limit

Don't let signals above this level pass the limiter. Default is 1.

attack

The limiter will reach its attenuation level in this amount of time in milliseconds. Default is 5 milliseconds.

release

Come back from limiting to attenuation 1.0 in this amount of milliseconds. Default is 50 milliseconds.

asc When gain reduction is always needed ASC takes care of releasing to an average reduction level rather than reaching a reduction of 0 in the release time.

asc_level

Select how much the release time is affected by ASC, 0 means nearly no changes in release time while 1 produces higher release times.

level

Auto level output signal. Default is enabled. This normalizes audio back to 0dB if enabled.

latency

Compensate the delay introduced by using the lookahead buffer set with attack parameter. Also flush the valid audio data in the lookahead buffer when the stream hits EOF.

Depending on picked setting it is recommended to upsample input 2x or 4x times with **aresample** before applying this filter.

allpass

Apply a two-pole all-pass filter with central frequency (in Hz) *frequency*, and filter-width *width*. An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship.

The filter accepts the following options:

frequency, f

Set frequency in Hz.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Specify the band-width of a filter in width_type units.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude

response at DC to 0dB.

order, o

Set the filter order, can be 1 or 2. Default is 2.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

Commands

This filter supports the following commands:

frequency, f

Change allpass frequency. Syntax for the command is : "*frequency*"

width_type, t

Change allpass width_type. Syntax for the command is : "*width_type*"

width, w

Change allpass width. Syntax for the command is : "*width*"

mix, m

Change allpass mix. Syntax for the command is : "*mix*"

aloop

Loop audio samples.

The filter accepts the following options:

loop

Set the number of loops. Setting this value to -1 will result in infinite loops. Default is 0.

size Set maximal number of samples. Default is 0.

start Set first sample of loop. Default is 0.

amerge

Merge two or more audio streams into a single multi-channel stream.

The filter accepts the following options:

inputs

Set the number of inputs. Default is 2.

If the channel layouts of the inputs are disjoint, and therefore compatible, the channel layout of the output will be set accordingly and the channels will be reordered as necessary. If the channel layouts of the inputs are not disjoint, the output will have all the channels of the first input then all the channels of the second input, in that order, and the channel layout of the output will be the default value corresponding to the total number of channels.

For example, if the first input is in 2.1 (FL+FR+LF) and the second input is FC+BL+BR, then the output will be in 5.1, with the channels in the following order: a1, a2, b1, a3, b2, b3 (a1 is the first channel of the first input, b1 is the first channel of the second input).

On the other hand, if both input are in stereo, the output channels will be in the default order: a1, a2, b1, b2, and the channel layout will be arbitrarily set to 4.0, which may or may not be the expected value.

All inputs must have the same sample rate, and format.

If inputs do not have the same duration, the output will stop with the shortest.

Examples

- ⊕ Merge two mono files into a stereo stream:

```
amovie=left.wav [l] ; amovie=right.mp3 [r] ; [l] [r] amerge
```

- ⊕ Multiple merges assuming 1 video stream and 6 audio streams in *input.mkv*:

```
ffmpeg -i input.mkv -filter_complex "[0:1][0:2][0:3][0:4][0:5][0:6] amerge=inputs=6" -c:a pcm_s16le output
```

amix

Mixes multiple audio inputs into a single output.

Note that this filter only supports float samples (the *amerge* and *pan* audio filters support many formats). If the *amix* input has integer samples then **aresample** will be automatically inserted to perform the conversion to float samples.

It accepts the following parameters:

inputs

The number of inputs. If unspecified, it defaults to 2.

duration

How to determine the end-of-stream.

longest

The duration of the longest input. (default)

shortest

The duration of the shortest input.

first The duration of the first input.

dropout_transition

The transition time, in seconds, for volume renormalization when an input stream ends. The default value is 2 seconds.

weights

Specify weight of each input audio stream as a sequence of numbers separated by a space. If fewer weights are specified compared to number of inputs, the last weight is assigned to the remaining

inputs. Default weight for each input is 1.

normalize

Always scale inputs instead of only doing summation of samples. Beware of heavy clipping if inputs are not normalized prior or after filtering by this filter if this option is disabled. By default is enabled.

Examples

- ⊕ This will mix 3 input audio streams to a single output with the same duration as the first input and a dropout transition time of 3 seconds:

```
ffmpeg -i INPUT1 -i INPUT2 -i INPUT3 -filter_complex amix=inputs=3:duration=first:dropout_transition=3
```

- ⊕ This will mix one vocal and one music input audio stream to a single output with the same duration as the longest input. The music will have quarter the weight as the vocals, and the inputs are not normalized:

```
ffmpeg -i VOCALS -i MUSIC -filter_complex amix=inputs=2:duration=longest:dropout_transition=0:weights=1:0.25
```

Commands

This filter supports the following commands:

weights

normalize

Syntax is same as option with same name.

amultiply

Multiply first audio stream with second audio stream and store result in output audio stream.

Multiplication is done by multiplying each sample from first stream with sample at same position from second stream.

With this element-wise multiplication one can create amplitude fades and amplitude modulations.

anequalizer

High-order parametric multiband equalizer for each channel.

It accepts the following parameters:

params

This option string is in format: "*chn f=cf w=w g=g t=f| ...*" Each equalizer band is separated by `'|'`.

chn Set channel number to which equalization will be applied. If input doesn't have that channel the entry is ignored.

f Set central frequency for band. If input doesn't have that frequency the entry is ignored.

w Set band width in Hertz.

g Set band gain in dB.

t Set filter type for band, optional, can be:

0 Butterworth, this is default.

1 Chebyshev type 1.

2 Chebyshev type 2.

curves

With this option activated frequency response of anequalizer is displayed in video stream.

size Set video stream size. Only useful if curves option is activated.

mgain

Set max gain that will be displayed. Only useful if curves option is activated. Setting this to a reasonable value makes it possible to display gain which is derived from neighbour bands which are too close to each other and thus produce higher gain when both are activated.

fscale

Set frequency scale used to draw frequency response in video output. Can be linear or logarithmic. Default is logarithmic.

colors

Set color for each channel curve which is going to be displayed in video stream. This is list of color names separated by space or by `'|'`. Unrecognised or missing colors will be replaced by white color.

Examples

- ⊕ Lower gain by 10 of central frequency 200Hz and width 100 Hz for first 2 channels using Chebyshev type 1 filter:

```
anequalizer=c0 f=200 w=100 g=-10 t=1|c1 f=200 w=100 g=-10 t=1
```

Commands

This filter supports the following commands:

change

Alter existing filter parameters. Syntax for the commands is : "*fN*|*f=freq*|*w=width*|*g=gain*"

fN is existing filter number, starting from 0, if no such filter is available error is returned. *freq* set new frequency parameter. *width* set new width parameter in Hertz. *gain* set new gain parameter in dB.

Full filter invocation with `asendcmd` may look like this: `asendcmd=c='4.0 anequalizer change 0|f=200|w=50|g=1',anequalizer=...`

anlmdn

Reduce broadband noise in audio samples using Non-Local Means algorithm.

Each sample is adjusted by looking for other samples with similar contexts. This context similarity is defined by comparing their surrounding patches of size **p**. Patches are searched in an area of **r** around the sample.

The filter accepts the following options:

strength, s

Set denoising strength. Allowed range is from 0.00001 to 10000. Default value is 0.00001.

patch, p

Set patch radius duration. Allowed range is from 1 to 100 milliseconds. Default value is 2 milliseconds.

research, r

Set research radius duration. Allowed range is from 2 to 300 milliseconds. Default value is 6 milliseconds.

output, o

Set the output mode.

It accepts the following values:

- i** Pass input unchanged.
- o** Pass noise filtered out.
- n** Pass only noise.

Default value is *o*.

smooth, m

Set smooth factor. Default value is *11*. Allowed range is from *1* to *1000*.

Commands

This filter supports the all above options as **commands**.

anlmf, anlms

Apply Normalized Least-Mean-(Squares|Fourth) algorithm to the first audio stream using the second audio stream.

This adaptive filter is used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal (difference between the desired, 2nd input audio stream and the actual signal, the 1st input audio stream).

A description of the accepted options follows.

order

Set filter order.

mu Set filter mu.

eps Set the filter eps.

leakage

Set the filter leakage.

out_mode

It accepts the following values:

- i** Pass the 1st input.
- d** Pass the 2nd input.
- o** Pass filtered samples.
- n** Pass difference between desired and filtered samples.

Default value is *o*.

Examples

- ⊕ One of many usages of this filter is noise reduction, input audio is filtered with same samples that are delayed by fixed amount, one such example for stereo audio is:

```
asplit[a][b],[a]adelay=32S|32S[a],[b][a]nlms=order=128:leakage=0.0005:mu=.5:out_mode=o
```

Commands

This filter supports the same commands as options, excluding option "order".

anull

Pass the audio source unchanged to the output.

apad

Pad the end of an audio stream with silence.

This can be used together with **ffmpeg -shortest** to extend audio streams to the same length as the video stream.

A description of the accepted options follows.

packet_size

Set silence packet size. Default value is 4096.

pad_len

Set the number of samples of silence to add to the end. After the value is reached, the stream is

terminated. This option is mutually exclusive with **whole_len**.

whole_len

Set the minimum total number of samples in the output audio stream. If the value is longer than the input audio length, silence is added to the end, until the value is reached. This option is mutually exclusive with **pad_len**.

pad_dur

Specify the duration of samples of silence to add. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. Used only if set to non-negative value.

whole_dur

Specify the minimum total duration in the output audio stream. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. Used only if set to non-negative value. If the value is longer than the input audio length, silence is added to the end, until the value is reached. This option is mutually exclusive with **pad_dur**

If neither the **pad_len** nor the **whole_len** nor **pad_dur** nor **whole_dur** option is set, the filter will add silence to the end of the input stream indefinitely.

Note that for ffmpeg 4.4 and earlier a zero **pad_dur** or **whole_dur** also caused the filter to add silence indefinitely.

Examples

- ⊕ Add 1024 samples of silence to the end of the input:

```
apad=pad_len=1024
```

- ⊕ Make sure the audio output will contain at least 10000 samples, pad the input with silence if required:

```
apad=whole_len=10000
```

- ⊕ Use **ffmpeg** to pad the audio input with silence, so that the video stream will always result the shortest and will be converted until the end in the output file when using the **shortest** option:

```
ffmpeg -i VIDEO -i AUDIO -filter_complex "[1:0]apad" -shortest OUTPUT
```

aphaser

Add a phasing effect to the input audio.

A phaser filter creates series of peaks and troughs in the frequency spectrum. The position of the peaks and troughs are modulated so that they vary over time, creating a sweeping effect.

A description of the accepted parameters follows.

in_gain

Set input gain. Default is 0.4.

out_gain

Set output gain. Default is 0.74

delay

Set delay in milliseconds. Default is 3.0.

decay

Set decay. Default is 0.4.

speed

Set modulation speed in Hz. Default is 0.5.

type Set modulation type. Default is triangular.

It accepts the following values:

triangular, t
sinusoidal, s

aphaseshift

Apply phase shift to input audio samples.

The filter accepts the following options:

shift

Specify phase shift. Allowed range is from -1.0 to 1.0. Default value is 0.0.

level

Set output gain applied to final output. Allowed range is from 0.0 to 1.0. Default value is 1.0.

order

Set filter order used for filtering. Allowed range is from 1 to 16. Default value is 8.

Commands

This filter supports the all above options as **commands**.

apsyclip

Apply Psychoacoustic clipper to input audio stream.

The filter accepts the following options:

level_in

Set input gain. By default it is 1. Range is [0.015625 - 64].

level_out

Set output gain. By default it is 1. Range is [0.015625 - 64].

clip Set the clipping start value. Default value is 0dBFS or 1.

diff Output only difference samples, useful to hear introduced distortions. By default is disabled.

adaptive

Set strength of adaptive distortion applied. Default value is 0.5. Allowed range is from 0 to 1.

iterations

Set number of iterations of psychoacoustic clipper. Allowed range is from 1 to 20. Default value is 10.

level

Auto level output signal. Default is disabled. This normalizes audio back to 0dBFS if enabled.

Commands

This filter supports the all above options as **commands**.

apulsator

Audio pulsator is something between an autopanner and a tremolo. But it can produce funny stereo effects as well. Pulsator changes the volume of the left and right channel based on a LFO (low frequency oscillator) with different waveforms and shifted phases. This filter have the ability to define

an offset between left and right channel. An offset of 0 means that both LFO shapes match each other. The left and right channel are altered equally - a conventional tremolo. An offset of 50% means that the shape of the right channel is exactly shifted in phase (or moved backwards about half of the frequency) - pulsator acts as an autopanner. At 1 both curves match again. Every setting in between moves the phase shift gapless between all stages and produces some "bypassing" sounds with sine and triangle waveforms. The more you set the offset near 1 (starting from the 0.5) the faster the signal passes from the left to the right speaker.

The filter accepts the following options:

level_in

Set input gain. By default it is 1. Range is [0.015625 - 64].

level_out

Set output gain. By default it is 1. Range is [0.015625 - 64].

mode

Set waveform shape the LFO will use. Can be one of: sine, triangle, square, sawup or sawdown. Default is sine.

amount

Set modulation. Define how much of original signal is affected by the LFO.

offset_l

Set left channel offset. Default is 0. Allowed range is [0 - 1].

offset_r

Set right channel offset. Default is 0.5. Allowed range is [0 - 1].

width

Set pulse width. Default is 1. Allowed range is [0 - 2].

timing

Set possible timing mode. Can be one of: bpm, ms or hz. Default is hz.

bpm

Set bpm. Default is 120. Allowed range is [30 - 300]. Only used if timing is set to bpm.

ms Set ms. Default is 500. Allowed range is [10 - 2000]. Only used if timing is set to ms.

hz Set frequency in Hz. Default is 2. Allowed range is [0.01 - 100]. Only used if timing is set to hz.

aresample

Resample the input audio to the specified parameters, using the libswresample library. If none are specified then the filter will automatically convert between its input and output.

This filter is also able to stretch/squeeze the audio data to make it match the timestamps or to inject silence / cut out audio to make it match the timestamps, do a combination of both or do neither.

The filter accepts the syntax [*sample_rate*:]*resampler_options*, where *sample_rate* expresses a sample rate and *resampler_options* is a list of *key=value* pairs, separated by ":". See the "**Resampler Options**" section in the **ffmpeg-resampler(1) manual** for the complete list of supported options.

Examples

- ⊕ Resample the input audio to 44100Hz:

```
aresample=44100
```

- ⊕ Stretch/squeeze samples to the given timestamps, with a maximum of 1000 samples per second compensation:

```
aresample=async=1000
```

areverse

Reverse an audio clip.

Warning: This filter requires memory to buffer the entire clip, so trimming is suggested.

Examples

- ⊕ Take the first 5 seconds of a clip, and reverse it.

```
atrim=end=5,areverse
```

arnndn

Reduce noise from speech using Recurrent Neural Networks.

This filter accepts the following options:

model, m

Set train model file to load. This option is always required.

mix Set how much to mix filtered samples into final output. Allowed range is from -1 to 1. Default value is 1. Negative values are special, they set how much to keep filtered noise in the final filter output. Set this option to -1 to hear actual noise removed from input signal.

Commands

This filter supports the all above options as **commands**.

asdr

Measure Audio Signal-to-Distortion Ratio.

This filter takes two audio streams for input, and outputs first audio stream. Results are in dB per channel at end of either input.

asetnsamples

Set the number of samples per each output audio frame.

The last output packet may contain a different number of samples, as the filter will flush all the remaining samples when the input audio signals its end.

The filter accepts the following options:

nb_out_samples, n

Set the number of frames per each output audio frame. The number is intended as the number of samples *per each channel*. Default value is 1024.

pad, p

If set to 1, the filter will pad the last audio frame with zeroes, so that the last frame will contain the same number of samples as the previous ones. Default value is 1.

For example, to set the number of per-frame samples to 1234 and disable padding for the last frame, use:

```
asetnsamples=n=1234:p=0
```

asetrate

Set the sample rate without altering the PCM data. This will result in a change of speed and pitch.

The filter accepts the following options:

sample_rate, r

Set the output sample rate. Default is 44100 Hz.

ashowinfo

Show a line containing various information for each input audio frame. The input audio is not modified.

The shown line contains a sequence of key/value pairs of the form *key:value*.

The following values are shown in the output:

n The (sequential) number of the input frame, starting from 0.

pts The presentation timestamp of the input frame, in time base units; the time base depends on the filter input pad, and is usually $1/sample_rate$.

pts_time

The presentation timestamp of the input frame in seconds.

pos position of the frame in the input stream, -1 if this information is unavailable and/or meaningless (for example in case of synthetic audio)

fmt The sample format.

chlayout

The channel layout.

rate The sample rate for the audio frame.

nb_samples

The number of samples (per channel) in the frame.

checksum

The Adler-32 checksum (printed in hexadecimal) of the audio data. For planar audio, the data is treated as if all the planes were concatenated.

plane_checksums

A list of Adler-32 checksums for each data plane.

asoftclip

Apply audio soft clipping.

Soft clipping is a type of distortion effect where the amplitude of a signal is saturated along a smooth curve, rather than the abrupt shape of hard-clipping.

This filter accepts the following options:

type Set type of soft-clipping.

It accepts the following values:

hard

tanh

atan

cubic

exp

alg

quintic

sin

erf

threshold

Set threshold from where to start clipping. Default value is 0dB or 1.

output

Set gain applied to output. Default value is 0dB or 1.

param

Set additional parameter which controls sigmoid function.

oversample

Set oversampling factor.

Commands

This filter supports the all above options as **commands**.

aspectralstats

Display frequency domain statistical information about the audio channels. Statistics are calculated and stored as metadata for each audio channel and for each audio frame.

It accepts the following option:

win_size

Set the window length in samples. Default value is 2048. Allowed range is from 32 to 65536.

win_func

Set window function.

It accepts the following values:

rect

bartlett

hann, hanning

hamming

blackman

welch

flattop

bharris

bnutall

bhann

sine

nutall

lanczos

gauss

tukey

dolph

cauchy

parzen

poisson

bohman

kaiser

Default is "hann".

overlap

Set window overlap. Allowed range is from 0 to 1. Default value is 0.5.

measure

Select the parameters which are measured. The metadata keys can be used as flags, default is **all** which measures everything. **none** disables all measurement.

A list of each metadata key follows:

mean
variance
centroid
spread
skewness
kurtosis
entropy
flatness
crest
flux
slope
decrease
rolloff

asr

Automatic Speech Recognition

This filter uses PocketSphinx for speech recognition. To enable compilation of this filter, you need to configure FFmpeg with "--enable-pocketsphinx".

It accepts the following options:

rate Set sampling rate of input audio. Defaults is 16000. This need to match speech models, otherwise one will get poor results.

hmm

Set dictionary containing acoustic model files.

dict Set pronunciation dictionary.

lm Set language model file.

lmctl

Set language model set.

lmname

Set which language model to use.

logfn

Set output for log messages.

The filter exports recognized speech as the frame metadata "lavfi.asr.text".

astats

Display time domain statistical information about the audio channels. Statistics are calculated and displayed for each audio channel and, where applicable, an overall figure is also given.

It accepts the following option:

length

Short window length in seconds, used for peak and trough RMS measurement. Default is 0.05 (50 milliseconds). Allowed range is "[0 - 10]".

metadata

Set metadata injection. All the metadata keys are prefixed with "lavfi.astats.X", where "X" is channel number starting from 1 or string "Overall". Default is disabled.

Available keys for each channel are: *Bit_depth Crest_factor DC_offset Dynamic_range Entropy Flat_factor Max_difference Max_level Mean_difference Min_difference Min_level Noise_floor Noise_floor_count Number_of_Infs Number_of_NaNs Number_of_denormals Peak_count Peak_level RMS_difference RMS_peak RMS_trough Zero_crossings Zero_crossings_rate*

and for "Overall": *Bit_depth DC_offset Entropy Flat_factor Max_difference Max_level Mean_difference Min_difference Min_level Noise_floor Noise_floor_count Number_of_Infs Number_of_NaNs Number_of_denormals Number_of_samples Peak_count Peak_level RMS_difference RMS_level RMS_peak RMS_trough*

For example, a full key looks like "lavfi.astats.1.DC_offset" or "lavfi.astats.Overall.Peak_count".

Read below for the description of the keys.

reset

Set the number of frames over which cumulative stats are calculated before being reset. Default is disabled.

measure_perchannel

Select the parameters which are measured per channel. The metadata keys can be used as flags, default is **all** which measures everything. **none** disables all per channel measurement.

measure_overall

Select the parameters which are measured overall. The metadata keys can be used as flags, default is **all** which measures everything. **none** disables all overall measurement.

A description of the measure keys follow:

none

no measures

all all measures

Bit_depth

overall bit depth of audio, i.e. number of bits used for each sample

Crest_factor

standard ratio of peak to RMS level (note: not in dB)

DC_offset

mean amplitude displacement from zero

Dynamic_range

measured dynamic range of audio in dB

Entropy

entropy measured across whole audio, entropy of value near 1.0 is typically measured for white noise

Flat_factor

flatness (i.e. consecutive samples with the same value) of the signal at its peak levels (i.e. either *Min_level* or *Max_level*)

Max_difference

maximal difference between two consecutive samples

Max_level

maximal sample level

Mean_difference

mean difference between two consecutive samples, i.e. the average of each difference between two consecutive samples

Min_difference

minimal difference between two consecutive samples

Min_level

minimal sample level

Noise_floor

minimum local peak measured in dBFS over a short window

Noise_floor_count

number of occasions (not the number of samples) that the signal attained *Noise floor*

Number_of_Infs

number of samples with an infinite value

Number_of_NaNs

number of samples with a NaN (not a number) value

Number_of_denormals

number of samples with a subnormal value

Number_of_samples

number of samples

Peak_count

number of occasions (not the number of samples) that the signal attained either *Min_level* or *Max_level*

Peak_level

standard peak level measured in dBFS

RMS_difference

Root Mean Square difference between two consecutive samples

RMS_level

standard RMS level measured in dBFS

RMS_peak**RMS_trough**

peak and trough values for RMS level measured over a short window, measured in dBFS.

Zero crossings

number of points where the waveform crosses the zero level axis

Zero crossings rate

rate of Zero crossings and number of audio samples

asubboost

Boost subwoofer frequencies.

The filter accepts the following options:

dry Set dry gain, how much of original signal is kept. Allowed range is from 0 to 1. Default value is 1.0.

wet Set wet gain, how much of filtered signal is kept. Allowed range is from 0 to 1. Default value is 1.0.

boost

Set max boost factor. Allowed range is from 1 to 12. Default value is 2.

decay

Set delay line decay gain value. Allowed range is from 0 to 1. Default value is 0.0.

feedback

Set delay line feedback gain value. Allowed range is from 0 to 1. Default value is 0.9.

cutoff

Set cutoff frequency in Hertz. Allowed range is 50 to 900. Default value is 100.

slope

Set slope amount for cutoff frequency. Allowed range is 0.0001 to 1. Default value is 0.5.

delay

Set delay. Allowed range is from 1 to 100. Default value is 20.

channels

Set the channels to process. Default value is all available.

Commands

This filter supports the all above options as **commands**.

asubcut

Cut subwoofer frequencies.

This filter allows to set custom, steeper roll off than highpass filter, and thus is able to more attenuate frequency content in stop-band.

The filter accepts the following options:

cutoff

Set cutoff frequency in Hertz. Allowed range is 2 to 200. Default value is 20.

order

Set filter order. Available values are from 3 to 20. Default value is 10.

level

Set input gain level. Allowed range is from 0 to 1. Default value is 1.

Commands

This filter supports the all above options as **commands**.

asupercut

Cut super frequencies.

The filter accepts the following options:

cutoff

Set cutoff frequency in Hertz. Allowed range is 20000 to 192000. Default value is 20000.

order

Set filter order. Available values are from 3 to 20. Default value is 10.

level

Set input gain level. Allowed range is from 0 to 1. Default value is 1.

Commands

This filter supports the all above options as **commands**.

asuperpass

Apply high order Butterworth band-pass filter.

The filter accepts the following options:

centerf

Set center frequency in Hertz. Allowed range is 2 to 999999. Default value is 1000.

order

Set filter order. Available values are from 4 to 20. Default value is 4.

qfactor

Set Q-factor. Allowed range is from 0.01 to 100. Default value is 1.

level

Set input gain level. Allowed range is from 0 to 2. Default value is 1.

Commands

This filter supports the all above options as **commands**.

asuperstop

Apply high order Butterworth band-stop filter.

The filter accepts the following options:

centerf

Set center frequency in Hertz. Allowed range is 2 to 999999. Default value is 1000.

order

Set filter order. Available values are from 4 to 20. Default value is 4.

qfactor

Set Q-factor. Allowed range is from 0.01 to 100. Default value is 1.

level

Set input gain level. Allowed range is from 0 to 2. Default value is 1.

Commands

This filter supports the all above options as **commands**.

atempo

Adjust audio tempo.

The filter accepts exactly one parameter, the audio tempo. If not specified then the filter will assume nominal 1.0 tempo. Tempo must be in the [0.5, 100.0] range.

Note that tempo greater than 2 will skip some samples rather than blend them in. If for any reason this is a concern it is always possible to daisy-chain several instances of atempo to achieve the desired product tempo.

Examples

- ⊕ Slow down audio to 80% tempo:

```
atempo=0.8
```

- ⊕ To speed up audio to 300% tempo:

```
atempo=3
```

- ⊕ To speed up audio to 300% tempo by daisy-chaining two atempo instances:

```
atempo=sqrt(3),atempo=sqrt(3)
```

Commands

This filter supports the following commands:

tempo

Change filter tempo scale factor. Syntax for the command is : "*tempo*"

atilt

Apply spectral tilt filter to audio stream.

This filter apply any spectral roll-off slope over any specified frequency band.

The filter accepts the following options:

freq Set central frequency of tilt in Hz. Default is 10000 Hz.

slope

Set slope direction of tilt. Default is 0. Allowed range is from -1 to 1.

width

Set width of tilt. Default is 1000. Allowed range is from 100 to 10000.

order

Set order of tilt filter.

level

Set input volume level. Allowed range is from 0 to 4. Default is 1.

Commands

This filter supports the all above options as **commands**.

atrim

Trim the input so that the output contains one continuous subpart of the input.

It accepts the following parameters:

start Timestamp (in seconds) of the start of the section to keep. I.e. the audio sample with the timestamp *start* will be the first sample in the output.

end Specify time of the first audio sample that will be dropped, i.e. the audio sample immediately preceding the one with the timestamp *end* will be the last sample in the output.

start_pts

Same as *start*, except this option sets the start timestamp in samples instead of seconds.

end_pts

Same as *end*, except this option sets the end timestamp in samples instead of seconds.

duration

The maximum duration of the output in seconds.

start_sample

The number of the first sample that should be output.

end_sample

The number of the first sample that should be dropped.

start, **end**, and **duration** are expressed as time duration specifications; see **the Time duration section in the ffmpeg-utils(1) manual**.

Note that the first two sets of the start/end options and the **duration** option look at the frame timestamp, while the **_sample** options simply count the samples that pass through the filter. So **start/end_pts** and **start/end_sample** will give different results when the timestamps are wrong, inexact or do not start at zero. Also note that this filter does not modify the timestamps. If you wish to have the output timestamps start at zero, insert the **asetpts** filter after the **atrim** filter.

If multiple start or end options are set, this filter tries to be greedy and keep all samples that match at least one of the specified constraints. To keep only the part that matches all the constraints at once, chain multiple **atrim** filters.

The defaults are such that all the input is kept. So it is possible to set e.g. just the end values to keep everything before the specified time.

Examples:

- ⊕ Drop everything except the second minute of input:

```
ffmpeg -i INPUT -af atrim=60:120
```

- ⊕ Keep only the first 1000 samples:

```
ffmpeg -i INPUT -af atrim=end_sample=1000
```

axcorrelate

Calculate normalized windowed cross-correlation between two input audio streams.

Resulted samples are always between -1 and 1 inclusive. If result is 1 it means two input samples are highly correlated in that selected segment. Result 0 means they are not correlated at all. If result is -1 it means two input samples are out of phase, which means they cancel each other.

The filter accepts the following options:

size Set size of segment over which cross-correlation is calculated. Default is 256. Allowed range is from 2 to 131072.

algo Set algorithm for cross-correlation. Can be "slow" or "fast". Default is "slow". Fast algorithm assumes mean values over any given segment are always zero and thus need much less calculations to make. This is generally not true, but is valid for typical audio streams.

Examples

- ⊕ Calculate correlation between channels in stereo audio stream:

```
ffmpeg -i stereo.wav -af channelsplit,axcorrelate=size=1024:algo=fast correlation.wav
```

bandpass

Apply a two-pole Butterworth band-pass filter with central frequency *frequency*, and (3dB-point) band-width *width*. The *csg* option selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain. The filter roll off at 6dB per octave (20dB per decade).

The filter accepts the following options:

frequency, f

Set the filter's central frequency. Default is 3000.

csg Constant skirt gain if set to 1. Defaults to 0.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Specify the band-width of a filter in *width_type* units.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports the following commands:

frequency, f

Change bandpass frequency. Syntax for the command is : "*frequency*"

width_type, t

Change bandpass width_type. Syntax for the command is : "*width_type*"

width, w

Change bandpass width. Syntax for the command is : "*width*"

mix, m

Change bandpass mix. Syntax for the command is : "*mix*"

bandreject

Apply a two-pole Butterworth band-reject filter with central frequency *frequency*, and (3dB-point) band-width *width*. The filter roll off at 6dB per octave (20dB per decade).

The filter accepts the following options:

frequency, f

Set the filter's central frequency. Default is 3000.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Specify the band-width of a filter in width_type units.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports the following commands:

frequency, f

Change bandreject frequency. Syntax for the command is : "*frequency*"

width_type, t

Change bandreject width_type. Syntax for the command is : "*width_type*"

width, w

Change bandreject width. Syntax for the command is : "*width*"

mix, m

Change bandreject mix. Syntax for the command is : "*mix*"

bass, lowshelf

Boost or cut the bass (lower) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

gain, g

Give the gain at 0 Hz. Its useful range is about -20 (for a large cut) to +20 (for a large boost).

Beware of clipping when using a positive gain.

frequency, f

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 100 Hz.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Determine how steep is the filter's shelf transition.

poles, p

Set number of poles. Default is 2.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher

than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports the following commands:

frequency, f

Change bass frequency. Syntax for the command is : "*frequency*"

width_type, t

Change bass width_type. Syntax for the command is : "*width_type*"

width, w

Change bass width. Syntax for the command is : "*width*"

gain, g

Change bass gain. Syntax for the command is : "*gain*"

mix, m

Change bass mix. Syntax for the command is : "*mix*"

biquad

Apply a biquad IIR filter with the given coefficients. Where b_0, b_1, b_2 and a_0, a_1, a_2 are the numerator and denominator coefficients respectively. and *channels, c* specify which channels to filter, by default all available are filtered.

Commands

This filter supports the following commands:

a0

a1

a2

b0

b1

b2 Change biquad parameter. Syntax for the command is : "*value*"

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

bs2b

Bauer stereo to binaural transformation, which improves headphone listening of stereo audio records.

To enable compilation of this filter you need to configure FFmpeg with "--enable-libbs2b".

It accepts the following parameters:

profile

Pre-defined crossfeed level.

default

Default level (fcut=700, feed=50).

cmoy

Chu Moy circuit (fcut=700, feed=60).

jmeier

Jan Meier circuit (fcut=650, feed=95).

fcut Cut frequency (in Hz).

feed Feed level (in Hz).

channelmap

Remap input channels to new locations.

It accepts the following parameters:

map Map channels from input to output. The argument is a '|'-separated list of mappings, each in the "*in_channel-out_channel*" or *in_channel* form. *in_channel* can be either the name of the input channel (e.g. FL for front left) or its index in the input channel layout. *out_channel* is the name of the output channel or its index in the output channel layout. If *out_channel* is not given then it is implicitly an index, starting with zero and increasing by one for each mapping.

channel_layout

The channel layout of the output stream.

If no mapping is present, the filter will implicitly map input channels to output channels, preserving indices.

Examples

- ⊕ For example, assuming a 5.1+downmix input MOV file,

```
ffmpeg -i in.mov -filter 'channelmap=map=DL-FL|DR-FR' out.wav
```

will create an output WAV file tagged as stereo from the downmix channels of the input.

- ⊕ To fix a 5.1 WAV improperly encoded in AAC's native channel order

```
ffmpeg -i in.wav -filter 'channelmap=1|2|0|5|3|4:5.1' out.wav
```

channelsplit

Split each channel from an input audio stream into a separate output stream.

It accepts the following parameters:

channel_layout

The channel layout of the input stream. The default is "stereo".

channels

A channel layout describing the channels to be extracted as separate output streams or "all" to extract each input channel as a separate stream. The default is "all".

Choosing channels not present in channel layout in the input will result in an error.

Examples

- ⊕ For example, assuming a stereo input MP3 file,

```
ffmpeg -i in.mp3 -filter_complex channelsplit out.mkv
```

will create an output Matroska file with two audio streams, one containing only the left channel and the other the right channel.

- ⊕ Split a 5.1 WAV file into per-channel files:

```
ffmpeg -i in.wav -filter_complex
'channelsplit=channel_layout=5.1[FL][FR][FC][LFE][SL][SR]'
-map '[FL]' front_left.wav -map '[FR]' front_right.wav -map '[FC]'
```

```
front_center.wav -map '[LFE]' lfe.wav -map '[SL]' side_left.wav -map '[SR]'
side_right.wav
```

- ⊕ Extract only LFE from a 5.1 WAV file:

```
ffmpeg -i in.wav -filter_complex 'channelsplit=channel_layout=5.1:channels=LFE[LFE]'
-map '[LFE]' lfe.wav
```

chorus

Add a chorus effect to the audio.

Can make a single vocal sound like a chorus, but can also be applied to instrumentation.

Chorus resembles an echo effect with a short delay, but whereas with echo the delay is constant, with chorus, it is varied using sinusoidal or triangular modulation. The modulation depth defines the range the modulated delay is played before or after the delay. Hence the delayed sound will sound slower or faster, that is the delayed sound tuned around the original one, like in a chorus where some vocals are slightly off key.

It accepts the following parameters:

in_gain

Set input gain. Default is 0.4.

out_gain

Set output gain. Default is 0.4.

delays

Set delays. A typical delay is around 40ms to 60ms.

decays

Set decays.

speeds

Set speeds.

depths

Set depths.

Examples

- ⊕ A single delay:

chorus=0.7:0.9:55:0.4:0.25:2

- ⊕ Two delays:

chorus=0.6:0.9:50|60:0.4|0.32:0.25|0.4:2|1.3

- ⊕ Fuller sounding chorus with three delays:

chorus=0.5:0.9:50|60|40:0.4|0.32|0.3:0.25|0.4|0.3:2|2.3|1.3

compand

Compress or expand the audio's dynamic range.

It accepts the following parameters:

attacks

decays

A list of times in seconds for each channel over which the instantaneous level of the input signal is averaged to determine its volume. *attacks* refers to increase of volume and *decays* refers to decrease of volume. For most situations, the attack time (response to the audio getting louder) should be shorter than the decay time, because the human ear is more sensitive to sudden loud audio than sudden soft audio. A typical value for attack is 0.3 seconds and a typical value for decay is 0.8 seconds. If specified number of attacks & decays is lower than number of channels, the last set attack/decay will be used for all remaining channels.

points

A list of points for the transfer function, specified in dB relative to the maximum possible signal amplitude. Each key points list must be defined using the following syntax:

"x0/y0|x1/y1|x2/y2|...." or "x0/y0 x1/y1 x2/y2"

The input values must be in strictly increasing order but the transfer function does not have to be monotonically rising. The point "0/0" is assumed but may be overridden (by "0/out-dBn"). Typical values for the transfer function are "-70/-70|-60/-20|1/0".

soft-knee

Set the curve radius in dB for all joints. It defaults to 0.01.

gain Set the additional gain in dB to be applied at all points on the transfer function. This allows for

easy adjustment of the overall gain. It defaults to 0.

volume

Set an initial volume, in dB, to be assumed for each channel when filtering starts. This permits the user to supply a nominal level initially, so that, for example, a very large gain is not applied to initial signal levels before the companding has begun to operate. A typical value for audio which is initially quiet is -90 dB. It defaults to 0.

delay

Set a delay, in seconds. The input audio is analyzed immediately, but audio is delayed before being fed to the volume adjuster. Specifying a delay approximately equal to the attack/decay times allows the filter to effectively operate in predictive rather than reactive mode. It defaults to 0.

Examples

- ⊕ Make music with both quiet and loud passages suitable for listening to in a noisy environment:

```
compand=.3|.3:1|1:-90/-60|-60/-40|-40/-30|-20/-20:6:0:-90:0.2
```

Another example for audio with whisper and explosion parts:

```
compand=0|0:1|1:-90/-900|-70/-70|-30/-9|0/-3:6:0:0:0
```

- ⊕ A noise gate for when the noise is at a lower level than the signal:

```
compand=.1|.1:.2|.2:-900/-900|-50.1/-900|-50/-50:.01:0:-90:.1
```

- ⊕ Here is another noise gate, this time for when the noise is at a higher level than the signal (making it, in some ways, similar to squelch):

```
compand=.1|.1:.1|.1:-45.1/-45.1|-45/-900|0/-900:.01:45:-90:.1
```

- ⊕ 2:1 compression starting at -6dB:

```
compand=points=-80/-80|-6/-6|0/-3.8|20/3.5
```

- ⊕ 2:1 compression starting at -9dB:

```
compand=points=-80/-80|-9/-9|0/-5.3|20/2.9
```

- ⊕ 2:1 compression starting at -12dB:

compand=points=-80/-80|-12/-12|0/-6.8|20/1.9

- ⊕ 2:1 compression starting at -18dB:

compand=points=-80/-80|-18/-18|0/-9.8|20/0.7

- ⊕ 3:1 compression starting at -15dB:

compand=points=-80/-80|-15/-15|0/-10.8|20/-5.2

- ⊕ Compressor/Gate:

compand=points=-80/-105|-62/-80|-15.4/-15.4|0/-12|20/-7.6

- ⊕ Expander:

compand=attacks=0:points=-80/-169|-54/-80|-49.5/-64.6|-41.1/-41.1|-25.8/-15|-10.8/-4.5|0/0|20/8.3

- ⊕ Hard limiter at -6dB:

compand=attacks=0:points=-80/-80|-6/-6|20/-6

- ⊕ Hard limiter at -12dB:

compand=attacks=0:points=-80/-80|-12/-12|20/-12

- ⊕ Hard noise gate at -35 dB:

compand=attacks=0:points=-80/-115|-35.1/-80|-35/-35|20/20

- ⊕ Soft limiter:

compand=attacks=0:points=-80/-80|-12.4/-12.4|-6/-8|0/-6.8|20/-2.8

compensationdelay

Compensation Delay Line is a metric based delay to compensate differing positions of microphones or speakers.

For example, you have recorded guitar with two microphones placed in different locations. Because the front of sound wave has fixed speed in normal conditions, the phasing of microphones can vary and depends on their location and interposition. The best sound mix can be achieved when these microphones are in phase (synchronized). Note that a distance of ~30 cm between microphones makes one microphone capture the signal in antiphase to the other microphone. That makes the final mix sound moody. This filter helps to solve phasing problems by adding different delays to each microphone track and make them synchronized.

The best result can be reached when you take one track as base and synchronize other tracks one by one with it. Remember that synchronization/delay tolerance depends on sample rate, too. Higher sample rates will give more tolerance.

The filter accepts the following parameters:

mm Set millimeters distance. This is compensation distance for fine tuning. Default is 0.

cm Set cm distance. This is compensation distance for tightening distance setup. Default is 0.

m Set meters distance. This is compensation distance for hard distance setup. Default is 0.

dry Set dry amount. Amount of unprocessed (dry) signal. Default is 0.

wet Set wet amount. Amount of processed (wet) signal. Default is 1.

temp

Set temperature in degrees Celsius. This is the temperature of the environment. Default is 20.

Commands

This filter supports the all above options as **commands**.

crossfeed

Apply headphone crossfeed filter.

Crossfeed is the process of blending the left and right channels of stereo audio recording. It is mainly used to reduce extreme stereo separation of low frequencies.

The intent is to produce more speaker like sound to the listener.

The filter accepts the following options:

strength

Set strength of crossfeed. Default is 0.2. Allowed range is from 0 to 1. This sets gain of low shelf filter for side part of stereo image. Default is -6dB. Max allowed is -30db when strength is set to 1.

range

Set soundstage wideness. Default is 0.5. Allowed range is from 0 to 1. This sets cut off frequency of low shelf filter. Default is cut off near 1550 Hz. With range set to 1 cut off frequency is set to 2100 Hz.

slope

Set curve slope of low shelf filter. Default is 0.5. Allowed range is from 0.01 to 1.

level_in

Set input gain. Default is 0.9.

level_out

Set output gain. Default is 1.

block_size

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports the all above options as **commands**.

crystalizer

Simple algorithm for audio noise sharpening.

This filter linearly increases differences between each audio sample.

The filter accepts the following options:

- i** Sets the intensity of effect (default: 2.0). Must be in range between -10.0 to 0 (unchanged sound) to 10.0 (maximum effect). To inverse filtering use negative value.

- c** Enable clipping. By default is enabled.

Commands

This filter supports the all above options as **commands**.

dcshift

Apply a DC shift to the audio.

This can be useful to remove a DC offset (caused perhaps by a hardware problem in the recording chain) from the audio. The effect of a DC offset is reduced headroom and hence volume. The **astats** filter can be used to determine if a signal has a DC offset.

shift

Set the DC shift, allowed range is [-1, 1]. It indicates the amount to shift the audio.

limitergain

Optional. It should have a value much less than 1 (e.g. 0.05 or 0.02) and is used to prevent clipping.

deesser

Apply de-essing to the audio samples.

- i** Set intensity for triggering de-essing. Allowed range is from 0 to 1. Default is 0.
- m** Set amount of ducking on treble part of sound. Allowed range is from 0 to 1. Default is 0.5.
- f** How much of original frequency content to keep when de-essing. Allowed range is from 0 to 1. Default is 0.5.
- s** Set the output mode.

It accepts the following values:

- i** Pass input unchanged.
- o** Pass ess filtered out.
- e** Pass only ess.

Default value is *o*.

dialogueenhance

Enhance dialogue in stereo audio.

This filter accepts stereo input and produce surround (3.0) channels output. The newly produced front center channel have enhanced speech dialogue originally available in both stereo channels. This filter outputs front left and front right channels same as available in stereo input.

The filter accepts the following options:

original

Set the original center factor to keep in front center channel output. Allowed range is from 0 to 1. Default value is 1.

enhance

Set the dialogue enhance factor to put in front center channel output. Allowed range is from 0 to 3. Default value is 1.

voice

Set the voice detection factor. Allowed range is from 2 to 32. Default value is 2.

Commands

This filter supports the all above options as **commands**.

drmeter

Measure audio dynamic range.

DR values of 14 and higher is found in very dynamic material. DR of 8 to 13 is found in transition material. And anything less than 8 have very poor dynamics and is very compressed.

The filter accepts the following options:

length

Set window length in seconds used to split audio into segments of equal length. Default is 3 seconds.

dynaunorm

Dynamic Audio Normalizer.

This filter applies a certain amount of gain to the input audio in order to bring its peak magnitude to a target level (e.g. 0 dBFS). However, in contrast to more "simple" normalization algorithms, the Dynamic Audio Normalizer *dynamically* re-adjusts the gain factor to the input audio. This allows for applying extra gain to the "quiet" sections of the audio while avoiding distortions or clipping the "loud" sections. In other words: The Dynamic Audio Normalizer will "even out" the volume of quiet and loud sections, in the sense that the volume of each section is brought to the same target level. Note, however, that the Dynamic Audio Normalizer achieves this goal *without* applying "dynamic range compressing". It will retain 100% of the dynamic range *within* each section of the audio file.

framelen, f

Set the frame length in milliseconds. In range from 10 to 8000 milliseconds. Default is 500 milliseconds. The Dynamic Audio Normalizer processes the input audio in small chunks, referred to as frames. This is required, because a peak magnitude has no meaning for just a single sample value. Instead, we need to determine the peak magnitude for a contiguous sequence of sample values. While a "standard" normalizer would simply use the peak magnitude of the complete file, the Dynamic Audio Normalizer determines the peak magnitude individually for each frame. The length of a frame is specified in milliseconds. By default, the Dynamic Audio Normalizer uses a frame length of 500 milliseconds, which has been found to give good results with most files. Note that the exact frame length, in number of samples, will be determined automatically, based on the sampling rate of the individual input audio file.

gaussize, g

Set the Gaussian filter window size. In range from 3 to 301, must be odd number. Default is 31. Probably the most important parameter of the Dynamic Audio Normalizer is the "window size" of the Gaussian smoothing filter. The filter's window size is specified in frames, centered around the current frame. For the sake of simplicity, this must be an odd number. Consequently, the default value of 31 takes into account the current frame, as well as the 15 preceding frames and the 15 subsequent frames. Using a larger window results in a stronger smoothing effect and thus in less gain variation, i.e. slower gain adaptation. Conversely, using a smaller window results in a weaker smoothing effect and thus in more gain variation, i.e. faster gain adaptation. In other words, the more you increase this value, the more the Dynamic Audio Normalizer will behave like a "traditional" normalization filter. On the contrary, the more you decrease this value, the more the Dynamic Audio Normalizer will behave like a dynamic range compressor.

peak, p

Set the target peak value. This specifies the highest permissible magnitude level for the normalized audio input. This filter will try to approach the target peak magnitude as closely as possible, but at the same time it also makes sure that the normalized signal will never exceed the peak magnitude. A frame's maximum local gain factor is imposed directly by the target peak magnitude. The default value is 0.95 and thus leaves a headroom of 5%. It is not recommended

to go above this value.

maxgain, m

Set the maximum gain factor. In range from 1.0 to 100.0. Default is 10.0. The Dynamic Audio Normalizer determines the maximum possible (local) gain factor for each input frame, i.e. the maximum gain factor that does not result in clipping or distortion. The maximum gain factor is determined by the frame's highest magnitude sample. However, the Dynamic Audio Normalizer additionally bounds the frame's maximum gain factor by a predetermined (global) maximum gain factor. This is done in order to avoid excessive gain factors in "silent" or almost silent frames. By default, the maximum gain factor is 10.0, For most inputs the default value should be sufficient and it usually is not recommended to increase this value. Though, for input with an extremely low overall volume level, it may be necessary to allow even higher gain factors. Note, however, that the Dynamic Audio Normalizer does not simply apply a "hard" threshold (i.e. cut off values above the threshold). Instead, a "sigmoid" threshold function will be applied. This way, the gain factors will smoothly approach the threshold value, but never exceed that value.

targetrms, r

Set the target RMS. In range from 0.0 to 1.0. Default is 0.0 - disabled. By default, the Dynamic Audio Normalizer performs "peak" normalization. This means that the maximum local gain factor for each frame is defined (only) by the frame's highest magnitude sample. This way, the samples can be amplified as much as possible without exceeding the maximum signal level, i.e. without clipping. Optionally, however, the Dynamic Audio Normalizer can also take into account the frame's root mean square, abbreviated RMS. In electrical engineering, the RMS is commonly used to determine the power of a time-varying signal. It is therefore considered that the RMS is a better approximation of the "perceived loudness" than just looking at the signal's peak magnitude. Consequently, by adjusting all frames to a constant RMS value, a uniform "perceived loudness" can be established. If a target RMS value has been specified, a frame's local gain factor is defined as the factor that would result in exactly that RMS value. Note, however, that the maximum local gain factor is still restricted by the frame's highest magnitude sample, in order to prevent clipping.

coupling, n

Enable channels coupling. By default is enabled. By default, the Dynamic Audio Normalizer will amplify all channels by the same amount. This means the same gain factor will be applied to all channels, i.e. the maximum possible gain factor is determined by the "loudest" channel. However, in some recordings, it may happen that the volume of the different channels is uneven, e.g. one channel may be "quieter" than the other one(s). In this case, this option can be used to disable the channel coupling. This way, the gain factor will be determined independently for each channel, depending only on the individual channel's highest magnitude sample. This allows for harmonizing the volume of the different channels.

correctdc, c

Enable DC bias correction. By default is disabled. An audio signal (in the time domain) is a sequence of sample values. In the Dynamic Audio Normalizer these sample values are represented in the -1.0 to 1.0 range, regardless of the original input format. Normally, the audio signal, or "waveform", should be centered around the zero point. That means if we calculate the mean value of all samples in a file, or in a single frame, then the result should be 0.0 or at least very close to that value. If, however, there is a significant deviation of the mean value from 0.0, in either positive or negative direction, this is referred to as a DC bias or DC offset. Since a DC bias is clearly undesirable, the Dynamic Audio Normalizer provides optional DC bias correction. With DC bias correction enabled, the Dynamic Audio Normalizer will determine the mean value, or "DC correction" offset, of each input frame and subtract that value from all of the frame's sample values which ensures those samples are centered around 0.0 again. Also, in order to avoid "gaps" at the frame boundaries, the DC correction offset values will be interpolated smoothly between neighbouring frames.

altboundary, b

Enable alternative boundary mode. By default is disabled. The Dynamic Audio Normalizer takes into account a certain neighbourhood around each frame. This includes the preceding frames as well as the subsequent frames. However, for the "boundary" frames, located at the very beginning and at the very end of the audio file, not all neighbouring frames are available. In particular, for the first few frames in the audio file, the preceding frames are not known. And, similarly, for the last few frames in the audio file, the subsequent frames are not known. Thus, the question arises which gain factors should be assumed for the missing frames in the "boundary" region. The Dynamic Audio Normalizer implements two modes to deal with this situation. The default boundary mode assumes a gain factor of exactly 1.0 for the missing frames, resulting in a smooth "fade in" and "fade out" at the beginning and at the end of the input, respectively.

compress, s

Set the compress factor. In range from 0.0 to 30.0. Default is 0.0. By default, the Dynamic Audio Normalizer does not apply "traditional" compression. This means that signal peaks will not be pruned and thus the full dynamic range will be retained within each local neighbourhood. However, in some cases it may be desirable to combine the Dynamic Audio Normalizer's normalization algorithm with a more "traditional" compression. For this purpose, the Dynamic Audio Normalizer provides an optional compression (thresholding) function. If (and only if) the compression feature is enabled, all input frames will be processed by a soft knee thresholding function prior to the actual normalization process. Put simply, the thresholding function is going to prune all samples whose magnitude exceeds a certain threshold value. However, the Dynamic Audio Normalizer does not simply apply a fixed threshold value. Instead, the threshold value will be adjusted for each individual frame. In general, smaller parameters result in stronger compression, and vice versa. Values below 3.0 are not recommended, because audible distortion

may appear.

threshold, t

Set the target threshold value. This specifies the lowest permissible magnitude level for the audio input which will be normalized. If input frame volume is above this value frame will be normalized. Otherwise frame may not be normalized at all. The default value is set to 0, which means all input frames will be normalized. This option is mostly useful if digital noise is not wanted to be amplified.

channels, h

Specify which channels to filter, by default all available channels are filtered.

overlap, o

Specify overlap for frames. If set to 0 (default) no frame overlapping is done. Using >0 and <1 values will make less conservative gain adjustments, like when framelen option is set to smaller value, if framelen option value is compensated for non-zero overlap then gain adjustments will be smoother across time compared to zero overlap case.

curve, v

Specify the peak mapping curve expression which is going to be used when calculating gain applied to frames. The max output frame gain will still be limited by other options mentioned previously for this filter.

The expression can contain the following constants:

ch current channel number

sn current sample number

nb_channels

number of channels

t timestamp expressed in seconds

sr sample rate

p current frame peak value

Commands

This filter supports the all above options as **commands**.

earwax

Make audio easier to listen to on headphones.

This filter adds 'cues' to 44.1kHz stereo (i.e. audio CD format) audio so that when listened to on headphones the stereo image is moved from inside your head (standard for headphones) to outside and in front of the listener (standard for speakers).

Ported from SoX.

equalizer

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike bandpass and bandreject filters) that at all other frequencies is unchanged.

In order to produce complex equalisation curves, this filter can be given several times, each with a different central frequency.

The filter accepts the following options:

frequency, f

Set the filter's central frequency in Hz.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Specify the band-width of a filter in width_type units.

gain, g

Set the required gain or attenuation in dB. Beware of clipping when using a positive gain.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Examples

- ⊕ Attenuate 10 dB at 1000 Hz, with a bandwidth of 200 Hz:

```
equalizer=f=1000:t=h:width=200:g=-10
```

- ⊕ Apply 2 dB gain at 1000 Hz with Q 1 and attenuate 5 dB at 100 Hz with Q 2:

```
equalizer=f=1000:t=q:w=1:g=2,equalizer=f=100:t=q:w=2:g=-5
```

Commands

This filter supports the following commands:

frequency, f

Change equalizer frequency. Syntax for the command is : "*frequency*"

width_type, t

Change equalizer width_type. Syntax for the command is : "*width_type*"

width, w

Change equalizer width. Syntax for the command is : "*width*"

gain, g

Change equalizer gain. Syntax for the command is : "*gain*"

mix, m

Change equalizer mix. Syntax for the command is : "*mix*"

extrastereo

Linearly increases the difference between left and right channels which adds some sort of "live" effect to playback.

The filter accepts the following options:

- m** Sets the difference coefficient (default: 2.5). 0.0 means mono sound (average of both channels), with 1.0 sound will be unchanged, with -1.0 left and right channels will be swapped.

c Enable clipping. By default is enabled.

Commands

This filter supports the all above options as **commands**.

firequalizer

Apply FIR Equalization using arbitrary frequency response.

The filter accepts the following option:

gain Set gain curve equation (in dB). The expression can contain variables:

f the evaluated frequency

sr sample rate

ch channel number, set to 0 when multichannels evaluation is disabled

chid channel id, see libavutil/channel_layout.h, set to the first channel id when multichannels evaluation is disabled

chs number of channels

chlayout

channel_layout, see libavutil/channel_layout.h

and functions:

gain_interpolate(f)

interpolate gain on frequency f based on gain_entry

cubic_interpolate(f)

same as gain_interpolate, but smoother

This option is also available as command. Default is gain_interpolate(f).

gain_entry

Set gain entry for gain_interpolate function. The expression can contain functions:

entry(f, g)

store gain entry at frequency f with value g

This option is also available as command.

delay

Set filter delay in seconds. Higher value means more accurate. Default is 0.01.

accuracy

Set filter accuracy in Hz. Lower value means more accurate. Default is 5.

wfunc

Set window function. Acceptable values are:

rectangular

rectangular window, useful when gain curve is already smooth

hann

hann window (default)

hamming

hamming window

blackman

blackman window

nuttall3

3-terms continuous 1st derivative nuttall window

mnuttall3

minimum 3-terms discontinuous nuttall window

nuttall

4-terms continuous 1st derivative nuttall window

bnuttall

minimum 4-terms discontinuous nuttall (blackman-nuttall) window

bharris

blackman-harris window

tukey

tukey window

fixed

If enabled, use fixed number of audio samples. This improves speed when filtering with large delay. Default is disabled.

multi

Enable multichannels evaluation on gain. Default is disabled.

zero_phase

Enable zero phase mode by subtracting timestamp to compensate delay. Default is disabled.

scale

Set scale used by gain. Acceptable values are:

linlin

linear frequency, linear gain

linlog

linear frequency, logarithmic (in dB) gain (default)

loglin

logarithmic (in octave scale where 20 Hz is 0) frequency, linear gain

loglog

logarithmic frequency, logarithmic gain

dumpfile

Set file for dumping, suitable for gnuplot.

dumpscale

Set scale for dumpfile. Acceptable values are same with scale option. Default is linlog.

fft2 Enable 2-channel convolution using complex FFT. This improves speed significantly. Default is disabled.

min_phase

Enable minimum phase impulse response. Default is disabled.

Examples

- ⊕ lowpass at 1000 Hz:

```
firequalizer=gain='if(lt(f,1000), 0, -INF)'
```

- ⊕ lowpass at 1000 Hz with gain_entry:

```
firequalizer=gain_entry='entry(1000,0); entry(1001, -INF)'
```

- ⊕ custom equalization:

```
firequalizer=gain_entry='entry(100,0); entry(400, -4); entry(1000, -6); entry(2000, 0)'
```

- ⊕ higher delay with zero phase to compensate delay:

```
firequalizer=delay=0.1:fixed=on:zero_phase=on
```

- ⊕ lowpass on left channel, highpass on right channel:

```
firequalizer=gain='if(eq(chid,1), gain_interpolate(f), if(eq(chid,2), gain_interpolate(1e6+f), 0))'  
:gain_entry='entry(1000, 0); entry(1001,-INF); entry(1e6+1000,0)':multi=on
```

flanger

Apply a flanging effect to the audio.

The filter accepts the following options:

delay

Set base delay in milliseconds. Range from 0 to 30. Default value is 0.

depth

Set added sweep delay in milliseconds. Range from 0 to 10. Default value is 2.

regen

Set percentage regeneration (delayed signal feedback). Range from -95 to 95. Default value is 0.

width

Set percentage of delayed signal mixed with original. Range from 0 to 100. Default value is 71.

speed

Set sweeps per second (Hz). Range from 0.1 to 10. Default value is 0.5.

shape

Set swept wave shape, can be *triangular* or *sinusoidal*. Default value is *sinusoidal*.

phase

Set swept wave percentage-shift for multi channel. Range from 0 to 100. Default value is 25.

interp

Set delay-line interpolation, *linear* or *quadratic*. Default is *linear*.

haas

Apply Haas effect to audio.

Note that this makes most sense to apply on mono signals. With this filter applied to mono signals it give some directionality and stretches its stereo image.

The filter accepts the following options:

level_in

Set input level. By default is *1*, or 0dB

level_out

Set output level. By default is *1*, or 0dB.

side_gain

Set gain applied to side part of signal. By default is *1*.

middle_source

Set kind of middle source. Can be one of the following:

left Pick left channel.

right

Pick right channel.

mid Pick middle part signal of stereo image.

side Pick side part signal of stereo image.

middle_phase

Change middle phase. By default is disabled.

left_delay

Set left channel delay. By default is 2.05 milliseconds.

left_balance

Set left channel balance. By default is *-1*.

left_gain

Set left channel gain. By default is *1*.

left_phase

Change left phase. By default is disabled.

right_delay

Set right channel delay. By defaults is 2.12 milliseconds.

right_balance

Set right channel balance. By default is *1*.

right_gain

Set right channel gain. By default is *1*.

right_phase

Change right phase. By default is enabled.

hdcd

Decodes High Definition Compatible Digital (HDCD) data. A 16-bit PCM stream with embedded HDCD codes is expanded into a 20-bit PCM stream.

The filter supports the Peak Extend and Low-level Gain Adjustment features of HDCD, and detects the Transient Filter flag.

```
ffmpeg -i HDCD16.flac -af hdcd OUT24.flac
```

When using the filter with wav, note the default encoding for wav is 16-bit, so the resulting 20-bit stream will be truncated back to 16-bit. Use something like **-acodec pcm_s24le** after the filter to get 24-bit PCM output.

```
ffmpeg -i HDCD16.wav -af hcdcd OUT16.wav  
ffmpeg -i HDCD16.wav -af hcdcd -c:a pcm_s24le OUT24.wav
```

The filter accepts the following options:

disable_autoconvert

Disable any automatic format conversion or resampling in the filter graph.

process_stereo

Process the stereo channels together. If `target_gain` does not match between channels, consider it invalid and use the last valid `target_gain`.

cdt_ms

Set the code detect timer period in ms.

force_pe

Always extend peaks above -3dBFS even if PE isn't signaled.

analyze_mode

Replace audio with a solid tone and adjust the amplitude to signal some specific aspect of the decoding process. The output file can be loaded in an audio editor alongside the original to aid analysis.

"analyze_mode=pe:force_pe=true" can be used to see all samples above the PE level.

Modes are:

0, off

Disabled

1, ll

Gain adjustment level at each sample

2, pe

Samples where peak extend occurs

3, cdt

Samples where the code detect timer is active

4, tgm

Samples where the target gain does not match between channels

headphone

Apply head-related transfer functions (HRTFs) to create virtual loudspeakers around the user for binaural listening via headphones. The HRIRs are provided via additional streams, for each channel one stereo input stream is needed.

The filter accepts the following options:

map Set mapping of input streams for convolution. The argument is a '|' -separated list of channel names in order as they are given as additional stream inputs for filter. This also specify number of input streams. Number of input streams must be not less than number of channels in first stream plus one.

gain Set gain applied to audio. Value is in dB. Default is 0.

type Set processing type. Can be *time* or *freq*. *time* is processing audio in time domain which is slow. *freq* is processing audio in frequency domain which is fast. Default is *freq*.

lfe Set custom gain for LFE channels. Value is in dB. Default is 0.

size Set size of frame in number of samples which will be processed at once. Default value is *1024*. Allowed range is from 1024 to 96000.

hrir Set format of hrir stream. Default value is *stereo*. Alternative value is *multich*. If value is set to *stereo*, number of additional streams should be greater or equal to number of input channels in first input stream. Also each additional stream should have stereo number of channels. If value is set to *multich*, number of additional streams should be exactly one. Also number of input channels of additional stream should be equal or greater than twice number of channels of first input stream.

Examples

- ⊕ Full example using wav files as coefficients with amovie filters for 7.1 downmix, each amovie filter use stereo file with IR coefficients as input. The files give coefficients for each position of virtual loudspeaker:

```
ffmpeg -i input.wav
-filter_complex "amovie=azi_270_ele_0_DFC.wav[sr];amovie=azi_90_ele_0_DFC.wav[sl];amovie=azi_225
output.wav
```

- ⊕ Full example using wav files as coefficients with amovie filters for 7.1 downmix, but now in *multich hrir* format.

```
ffmpeg -i input.wav -filter_complex "amovie=minp.wav[hrirs];[0:a][hrirs]headphone=map=FL|FR|FC|LFE|B
output.wav
```

highpass

Apply a high-pass filter with 3dB point frequency. The filter can be either single-pole, or double-pole (the default). The filter roll off at 6dB per pole per octave (20dB per pole per decade).

The filter accepts the following options:

frequency, **f**

Set frequency in Hz. Default is 3000.

poles, **p**

Set number of poles. Default is 2.

width_type, **t**

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, **w**

Specify the band-width of a filter in width_type units. Applies only to double-pole filter. The default is 0.707q and gives a Butterworth response.

mix, **m**

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, **c**

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports the following commands:

frequency, f

Change highpass frequency. Syntax for the command is : "*frequency*"

width_type, t

Change highpass width_type. Syntax for the command is : "*width_type*"

width, w

Change highpass width. Syntax for the command is : "*width*"

mix, m

Change highpass mix. Syntax for the command is : "*mix*"

join

Join multiple input streams into one multi-channel stream.

It accepts the following parameters:

inputs

The number of input streams. It defaults to 2.

channel_layout

The desired output channel layout. It defaults to stereo.

map Map channels from inputs to output. The argument is a '|' -separated list of mappings, each in the "*input_idx.in_channel-out_channel*" form. *input_idx* is the 0-based index of the input stream. *in_channel* can be either the name of the input channel (e.g. FL for front left) or its index in the specified input stream. *out_channel* is the name of the output channel.

The filter will attempt to guess the mappings when they are not specified explicitly. It does so by first trying to find an unused matching input channel and if that fails it picks the first unused input channel.

Join 3 inputs (with properly set channel layouts):

```
ffmpeg -i INPUT1 -i INPUT2 -i INPUT3 -filter_complex join=inputs=3 OUTPUT
```

Build a 5.1 output from 6 single-channel streams:

```
ffmpeg -i fl -i fr -i fc -i sl -i sr -i lfe -filter_complex
'join=inputs=6:channel_layout=5.1:map=0.0-FL|1.0-FR|2.0-FC|3.0-SL|4.0-SR|5.0-LFE'
out
```

ladspa

Load a LADSPA (Linux Audio Developer's Simple Plugin API) plugin.

To enable compilation of this filter you need to configure FFmpeg with "--enable-ladspa".

file, f

Specifies the name of LADSPA plugin library to load. If the environment variable **LADSPA_PATH** is defined, the LADSPA plugin is searched in each one of the directories specified by the colon separated list in **LADSPA_PATH**, otherwise in the standard LADSPA paths, which are in this order: *HOME/.ladspa/lib/*, */usr/local/lib/ladspa/*, */usr/lib/ladspa/*.

plugin, p

Specifies the plugin within the library. Some libraries contain only one plugin, but others contain many of them. If this is not set filter will list all available plugins within the specified library.

controls, c

Set the '|' separated list of controls which are zero or more floating point values that determine the behavior of the loaded plugin (for example delay, threshold or gain). Controls need to be defined using the following syntax: *c0=value0|c1=value1|c2=value2|...*, where *valuei* is the value set on the *i*-th control. Alternatively they can be also defined using the following syntax: *value0|value1|value2|...*, where *valuei* is the value set on the *i*-th control. If **controls** is set to "help", all available controls and their valid ranges are printed.

sample_rate, s

Specify the sample rate, default to 44100. Only used if plugin have zero inputs.

nb_samples, n

Set the number of samples per channel per each output frame, default is 1024. Only used if plugin have zero inputs.

duration, d

Set the minimum duration of the sourced audio. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame. If not specified, or the expressed duration is negative, the audio is supposed to be generated forever. Only used if plugin have zero inputs.

latency, l

Enable latency compensation, by default is disabled. Only used if plugin have inputs.

Examples

- ⊕ List all available plugins within amp (LADSPA example plugin) library:

```
ladspa=file=amp
```

- ⊕ List all available controls and their valid ranges for "vcf_notch" plugin from "VCF" library:

```
ladspa=f=vcf:p=vcf_notch:c=help
```

- ⊕ Simulate low quality audio equipment using "Computer Music Toolkit" (CMT) plugin library:

```
ladspa=file=cmt:plugin=lofi:controls=c0=22|c1=12|c2=12
```

- ⊕ Add reverberation to the audio using TAP-plugins (Tom's Audio Processing plugins):

```
ladspa=file=tap_reverb:tap_reverb
```

- ⊕ Generate white noise, with 0.2 amplitude:

```
ladspa=file=cmt:noise_source_white:c=c0=.2
```

- ⊕ Generate 20 bpm clicks using plugin "C* Click - Metronome" from the "C* Audio Plugin Suite" (CAPS) library:

```
ladspa=file=caps:Click:c=c1=20'
```

- ⊕ Apply "C* Eq10X2 - Stereo 10-band equaliser" effect:

```
ladspa=caps:Eq10X2:c=c0=-48|c9=-24|c3=12|c4=2
```

- ⊕ Increase volume by 20dB using fast lookahead limiter from Steve Harris "SWH Plugins" collection:

```
ladspa=fast_lookahead_limiter_1913:fastLookaheadLimiter:20|0|2
```

- ⊕ Attenuate low frequencies using Multiband EQ from Steve Harris "SWH Plugins" collection:

```
ladspa=mbeq_1197:mbeq:-24|-24|-24|0|0|0|0|0|0|0|0|0|0|0
```

- ⊕ Reduce stereo image using "Narrower" from the "C* Audio Plugin Suite" (CAPS) library:

```
ladspa=caps:Narrower
```

- ⊕ Another white noise, now using "C* Audio Plugin Suite" (CAPS) library:

```
ladspa=caps:White:.2
```

- ⊕ Some fractal noise, using "C* Audio Plugin Suite" (CAPS) library:

```
ladspa=caps:Fractal:c=c1=1
```

- ⊕ Dynamic volume normalization using "VLevel" plugin:

```
ladspa=vlevel-ladspa:vlevel_mono
```

Commands

This filter supports the following commands:

- cN** Modify the *N*-th control value.

If the specified value is not valid, it is ignored and prior one is kept.

loudnorm

EBU R128 loudness normalization. Includes both dynamic and linear normalization modes. Support for both single pass (livestreams, files) and double pass (files) modes. This algorithm can target IL, LRA, and maximum true peak. In dynamic mode, to accurately detect true peaks, the audio stream will be upsampled to 192 kHz. Use the "-ar" option or "aresample" filter to explicitly set an output sample rate.

The filter accepts the following options:

- I, i** Set integrated loudness target. Range is -70.0 - -5.0. Default value is -24.0.

LRA, lra

Set loudness range target. Range is 1.0 - 50.0. Default value is 7.0.

TP, tp

Set maximum true peak. Range is -9.0 - +0.0. Default value is -2.0.

measured_I, measured_i

Measured IL of input file. Range is -99.0 - +0.0.

measured_LRA, measured_lra

Measured LRA of input file. Range is 0.0 - 99.0.

measured_TP, measured_tp

Measured true peak of input file. Range is -99.0 - +99.0.

measured_thresh

Measured threshold of input file. Range is -99.0 - +0.0.

offset

Set offset gain. Gain is applied before the true-peak limiter. Range is -99.0 - +99.0. Default is +0.0.

linear

Normalize by linearly scaling the source audio. "measured_I", "measured_LRA", "measured_TP", and "measured_thresh" must all be specified. Target LRA shouldn't be lower than source LRA and the change in integrated loudness shouldn't result in a true peak which exceeds the target TP. If any of these conditions aren't met, normalization mode will revert to *dynamic*. Options are "true" or "false". Default is "true".

dual_mono

Treat mono input files as "dual-mono". If a mono file is intended for playback on a stereo system, its EBU R128 measurement will be perceptually incorrect. If set to "true", this option will compensate for this effect. Multi-channel input files are not affected by this option. Options are true or false. Default is false.

print_format

Set print format for stats. Options are summary, json, or none. Default value is none.

lowpass

Apply a low-pass filter with 3dB point frequency. The filter can be either single-pole or double-pole (the default). The filter roll off at 6dB per pole per octave (20dB per pole per decade).

The filter accepts the following options:

frequency, f

Set frequency in Hz. Default is 500.

poles, p

Set number of poles. Default is 2.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Specify the band-width of a filter in width_type units. Applies only to double-pole filter. The default is 0.707q and gives a Butterworth response.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Examples

- ⊕ Lowpass only LFE channel, if LFE is not present it does nothing:

```
lowpass=c=LFE
```

Commands

This filter supports the following commands:

frequency, f

Change lowpass frequency. Syntax for the command is : "*frequency*"

width_type, t

Change lowpass width_type. Syntax for the command is : "*width_type*"

width, w

Change lowpass width. Syntax for the command is : "*width*"

mix, m

Change lowpass mix. Syntax for the command is : "*mix*"

Load a LV2 (LADSPA Version 2) plugin.

To enable compilation of this filter you need to configure FFmpeg with "--enable-lv2".

plugin, p

Specifies the plugin URI. You may need to escape `:`.

controls, c

Set the `|` separated list of controls which are zero or more floating point values that determine the behavior of the loaded plugin (for example delay, threshold or gain). If **controls** is set to "help", all available controls and their valid ranges are printed.

sample_rate, s

Specify the sample rate, default to 44100. Only used if plugin have zero inputs.

nb_samples, n

Set the number of samples per channel per each output frame, default is 1024. Only used if plugin have zero inputs.

duration, d

Set the minimum duration of the sourced audio. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame. If not specified, or the expressed duration is negative, the audio is supposed to be generated forever. Only used if plugin have zero inputs.

Examples

- ⊕ Apply bass enhancer plugin from Calf:

```
lv2=p=http\\:\\:\\calf.sourceforge.net/plugins/BassEnhancer:c=amount=2
```

- ⊕ Apply vinyl plugin from Calf:

```
lv2=p=http\\:\\:\\calf.sourceforge.net/plugins/Vinyl:c=drone=0.2|aging=0.5
```

- ⊕ Apply bit crusher plugin from ArtyFX:

```
lv2=p=http\\:\\:\\www.openavproductions.com/artyfx#bitta:c=crush=0.3
```

Commands

This filter supports all options that are exported by plugin as commands.

mcompand

Multiband Compress or expand the audio's dynamic range.

The input audio is divided into bands using 4th order Linkwitz-Riley IIRs. This is akin to the crossover of a loudspeaker, and results in flat frequency response when absent compander action.

It accepts the following parameters:

args This option syntax is: attack,decay,[attack,decay..] soft-knee points crossover_frequency [delay [initial_volume [gain]]] | attack,decay ... For explanation of each item refer to compand filter documentation.

pan

Mix channels with specific gain levels. The filter accepts the output channel layout followed by a set of channels definitions.

This filter is also designed to efficiently remap the channels of an audio stream.

The filter accepts parameters of the form: "*l*|*outdef*{*outdef*}|..."

l output channel layout or number of channels

outdef

output channel specification, of the form: "*out_name*=[*gain**]*in_name*[(+)*gain**]*in_name*..."

out_name

output channel to define, either a channel name (FL, FR, etc.) or a channel number (c0, c1, etc.)

gain multiplicative coefficient for the channel, 1 leaving the volume unchanged

in_name

input channel to use, see *out_name* for details; it is not possible to mix named and numbered input channels

If the '=' in a channel specification is replaced by '<', then the gains for that specification will be renormalized so that the total is 1, thus avoiding clipping noise.

Mixing examples

For example, if you want to down-mix from stereo to mono, but with a bigger factor for the left channel:

```
pan=1c|c0=0.9*c0+0.1*c1
```

A customized down-mix to stereo that works automatically for 3-, 4-, 5- and 7-channels surround:

```
pan=stereo| FL < FL + 0.5*FC + 0.6*BL + 0.6*SL | FR < FR + 0.5*FC + 0.6*BR + 0.6*SR
```

Note that **ffmpeg** integrates a default down-mix (and up-mix) system that should be preferred (see "-ac" option) unless you have very specific needs.

Remapping examples

The channel remapping will be effective if, and only if:

*<gain coefficients are zeroes or ones,>

*<only one input per channel output,>

If all these conditions are satisfied, the filter will notify the user ("Pure channel mapping detected"), and use an optimized and lossless method to do the remapping.

For example, if you have a 5.1 source and want a stereo audio stream by dropping the extra channels:

```
pan="stereo| c0=FL | c1=FR"
```

Given the same source, you can also switch front left and front right channels and keep the input channel layout:

```
pan="5.1| c0=c1 | c1=c0 | c2=c2 | c3=c3 | c4=c4 | c5=c5"
```

If the input is a stereo audio stream, you can mute the front left channel (and still keep the stereo channel layout) with:

```
pan="stereo|c1=c1"
```

Still with a stereo audio stream input, you can copy the right channel in both front left and right:

```
pan="stereo| c0=FR | c1=FR"
```

replaygain

ReplayGain scanner filter. This filter takes an audio stream as an input and outputs it unchanged. At end of filtering it displays "track_gain" and "track_peak".

resample

Convert the audio sample format, sample rate and channel layout. It is not meant to be used directly.

rubberband

Apply time-stretching and pitch-shifting with librubberband.

To enable compilation of this filter, you need to configure FFmpeg with "--enable-librubberband".

The filter accepts the following options:

tempo

Set tempo scale factor.

pitch

Set pitch scale factor.

transients

Set transients detector. Possible values are:

crisp

mixed

smooth

detector

Set detector. Possible values are:

compound

percussive

soft

phase

Set phase. Possible values are:

laminar

independent

window

Set processing window size. Possible values are:

standard

short

long

smoothing

Set smoothing. Possible values are:

off

on

formant

Enable formant preservation when shift pitching. Possible values are:

shifted

preserved

pitchq

Set pitch quality. Possible values are:

quality

speed

consistency

channels

Set channels. Possible values are:

apart

together

Commands

This filter supports the following commands:

tempo

Change filter tempo scale factor. Syntax for the command is : "*tempo*"

pitch

Change filter pitch scale factor. Syntax for the command is : "*pitch*"

sidechaincompress

This filter acts like normal compressor but has the ability to compress detected signal using second input signal. It needs two input streams and returns one output stream. First input stream will be

processed depending on second stream signal. The filtered signal then can be filtered with other filters in later stages of processing. See **pan** and **amerge** filter.

The filter accepts the following options:

level_in

Set input gain. Default is 1. Range is between 0.015625 and 64.

mode

Set mode of compressor operation. Can be "upward" or "downward". Default is "downward".

threshold

If a signal of second stream raises above this level it will affect the gain reduction of first stream. By default is 0.125. Range is between 0.00097563 and 1.

ratio

Set a ratio about which the signal is reduced. 1:2 means that if the level raised 4dB above the threshold, it will be only 2dB above after the reduction. Default is 2. Range is between 1 and 20.

attack

Amount of milliseconds the signal has to rise above the threshold before gain reduction starts. Default is 20. Range is between 0.01 and 2000.

release

Amount of milliseconds the signal has to fall below the threshold before reduction is decreased again. Default is 250. Range is between 0.01 and 9000.

makeup

Set the amount by how much signal will be amplified after processing. Default is 1. Range is from 1 to 64.

knee

Curve the sharp knee around the threshold to enter gain reduction more softly. Default is 2.82843. Range is between 1 and 8.

link Choose if the "average" level between all channels of side-chain stream or the louder("maximum") channel of side-chain stream affects the reduction. Default is "average".

detection

Should the exact signal be taken in case of "peak" or an RMS one in case of "rms". Default is

"rms" which is mainly smoother.

level_sc

Set sidechain gain. Default is 1. Range is between 0.015625 and 64.

mix How much to use compressed signal in output. Default is 1. Range is between 0 and 1.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Full ffmpeg example taking 2 audio inputs, 1st input to be compressed depending on the signal of 2nd input and later compressed signal to be merged with 2nd input:

```
ffmpeg -i main.flac -i sidechain.flac -filter_complex "[1:a]asplit=2[sc][mix];[0:a][sc]sidechaincompress[com
```

sidechaingate

A sidechain gate acts like a normal (wideband) gate but has the ability to filter the detected signal before sending it to the gain reduction stage. Normally a gate uses the full range signal to detect a level above the threshold. For example: If you cut all lower frequencies from your sidechain signal the gate will decrease the volume of your track only if not enough highs appear. With this technique you are able to reduce the resonance of a natural drum or remove "rumbling" of muted strokes from a heavily distorted guitar. It needs two input streams and returns one output stream. First input stream will be processed depending on second stream signal.

The filter accepts the following options:

level_in

Set input level before filtering. Default is 1. Allowed range is from 0.015625 to 64.

mode

Set the mode of operation. Can be "upward" or "downward". Default is "downward". If set to "upward" mode, higher parts of signal will be amplified, expanding dynamic range in upward direction. Otherwise, in case of "downward" lower parts of signal will be reduced.

range

Set the level of gain reduction when the signal is below the threshold. Default is 0.06125. Allowed range is from 0 to 1. Setting this to 0 disables reduction and then filter behaves like

expander.

threshold

If a signal rises above this level the gain reduction is released. Default is 0.125. Allowed range is from 0 to 1.

ratio

Set a ratio about which the signal is reduced. Default is 2. Allowed range is from 1 to 9000.

attack

Amount of milliseconds the signal has to rise above the threshold before gain reduction stops. Default is 20 milliseconds. Allowed range is from 0.01 to 9000.

release

Amount of milliseconds the signal has to fall below the threshold before the reduction is increased again. Default is 250 milliseconds. Allowed range is from 0.01 to 9000.

makeup

Set amount of amplification of signal after processing. Default is 1. Allowed range is from 1 to 64.

knee

Curve the sharp knee around the threshold to enter gain reduction more softly. Default is 2.828427125. Allowed range is from 1 to 8.

detection

Choose if exact signal should be taken for detection or an RMS like one. Default is rms. Can be peak or rms.

link Choose if the average level between all channels or the louder channel affects the reduction.

Default is average. Can be average or maximum.

level_sc

Set sidechain gain. Default is 1. Range is from 0.015625 to 64.

Commands

This filter supports the all above options as **commands**.

silencedetect

Detect silence in an audio stream.

This filter logs a message when it detects that the input audio volume is less or equal to a noise tolerance value for a duration greater or equal to the minimum detected noise duration.

The printed times and duration are expressed in seconds. The "lavfi.silence_start" or "lavfi.silence_start.X" metadata key is set on the first frame whose timestamp equals or exceeds the detection duration and it contains the timestamp of the first frame of the silence.

The "lavfi.silence_duration" or "lavfi.silence_duration.X" and "lavfi.silence_end" or "lavfi.silence_end.X" metadata keys are set on the first frame after the silence. If **mono** is enabled, and each channel is evaluated separately, the ".X" suffixed keys are used, and "X" corresponds to the channel number.

The filter accepts the following options:

noise, n

Set noise tolerance. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default is -60dB, or 0.001.

duration, d

Set silence duration until notification (default is 2 seconds). See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

mono, m

Process each channel separately, instead of combined. By default is disabled.

Examples

- ⊕ Detect 5 seconds of silence with -50dB noise tolerance:

```
silencedetect=n=-50dB:d=5
```

- ⊕ Complete example with **ffmpeg** to detect silence with 0.0001 noise tolerance in *silence.mp3*:

```
ffmpeg -i silence.mp3 -af silencedetect=noise=0.0001 -f null -
```

silenceremove

Remove silence from the beginning, middle or end of the audio.

The filter accepts the following options:

start_periods

This value is used to indicate if audio should be trimmed at beginning of the audio. A value of zero indicates no silence should be trimmed from the beginning. When specifying a non-zero value, it trims audio up until it finds non-silence. Normally, when trimming silence from beginning of audio the *start_periods* will be 1 but it can be increased to higher values to trim all audio up to specific count of non-silence periods. Default value is 0.

start_duration

Specify the amount of time that non-silence must be detected before it stops trimming audio. By increasing the duration, bursts of noises can be treated as silence and trimmed off. Default value is 0.

start_threshold

This indicates what sample value should be treated as silence. For digital audio, a value of 0 may be fine but for audio recorded from analog, you may wish to increase the value to account for background noise. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default value is 0.

start_silence

Specify max duration of silence at beginning that will be kept after trimming. Default is 0, which is equal to trimming all samples detected as silence.

start_mode

Specify mode of detection of silence end in start of multi-channel audio. Can be *any* or *all*. Default is *any*. With *any*, any sample that is detected as non-silence will cause stopped trimming of silence. With *all*, only if all channels are detected as non-silence will cause stopped trimming of silence.

stop_periods

Set the count for trimming silence from the end of audio. To remove silence from the middle of a file, specify a *stop_periods* that is negative. This value is then treated as a positive value and is used to indicate the effect should restart processing as specified by *start_periods*, making it suitable for removing periods of silence in the middle of the audio. Default value is 0.

stop_duration

Specify a duration of silence that must exist before audio is not copied any more. By specifying a higher duration, silence that is wanted can be left in the audio. Default value is 0.

stop_threshold

This is the same as **start_threshold** but for trimming silence from the end of audio. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default value is 0.

stop_silence

Specify max duration of silence at end that will be kept after trimming. Default is 0, which is equal to trimming all samples detected as silence.

stop_mode

Specify mode of detection of silence start in end of multi-channel audio. Can be *any* or *all*. Default is *any*. With *any*, any sample that is detected as non-silence will cause stopped trimming of silence. With *all*, only if all channels are detected as non-silence will cause stopped trimming of silence.

detection

Set how is silence detected. Can be "rms" or "peak". Second is faster and works better with digital silence which is exactly 0. Default value is "rms".

window

Set duration in number of seconds used to calculate size of window in number of samples for detecting silence. Default value is 0.02. Allowed range is from 0 to 10.

Examples

- ⊕ The following example shows how this filter can be used to start a recording that does not contain the delay at the start which usually occurs between pressing the record button and the start of the performance:

```
silenceremove=start_periods=1:start_duration=5:start_threshold=0.02
```

- ⊕ Trim all silence encountered from beginning to end where there is more than 1 second of silence in audio:

```
silenceremove=stop_periods=-1:stop_duration=1:stop_threshold=-90dB
```

- ⊕ Trim all digital silence samples, using peak detection, from beginning to end where there is more than 0 samples of digital silence in audio and digital silence is detected in all channels at same positions in stream:

```
silenceremove=window=0:detection=peak:stop_mode=all:start_mode=all:stop_periods=-1:stop_threshold=0
```

sofalizer

SOFAizer uses head-related transfer functions (HRTFs) to create virtual loudspeakers around the user for binaural listening via headphones (audio formats up to 9 channels supported). The HRTFs are stored in SOFA files (see <<http://www.sofacoustics.org/>> for a database). SOFAizer is developed at the Acoustics Research Institute (ARI) of the Austrian Academy of Sciences.

To enable compilation of this filter you need to configure FFmpeg with "--enable-libmysofa".

The filter accepts the following options:

sofa Set the SOFA file used for rendering.

gain Set gain applied to audio. Value is in dB. Default is 0.

rotation

Set rotation of virtual loudspeakers in deg. Default is 0.

elevation

Set elevation of virtual speakers in deg. Default is 0.

radius

Set distance in meters between loudspeakers and the listener with near-field HRTFs. Default is 1.

type Set processing type. Can be *time* or *freq*. *time* is processing audio in time domain which is slow. *freq* is processing audio in frequency domain which is fast. Default is *freq*.

speakers

Set custom positions of virtual loudspeakers. Syntax for this option is: <CH> <AZIM> <ELEV>[|<CH> <AZIM> <ELEV>|...]. Each virtual loudspeaker is described with short channel name following with azimuth and elevation in degrees. Each virtual loudspeaker description is separated by '|'. For example to override front left and front right channel positions use: 'speakers=FL 45 15|FR 345 15'. Descriptions with unrecognised channel names are ignored.

lfegain

Set custom gain for LFE channels. Value is in dB. Default is 0.

framesize

Set custom frame size in number of samples. Default is 1024. Allowed range is from 1024 to

96000. Only used if option **type** is set to *freq*.

normalize

Should all IRs be normalized upon importing SOFA file. By default is enabled.

interpolate

Should nearest IRs be interpolated with neighbor IRs if exact position does not match. By default is disabled.

minphase

Minphase all IRs upon loading of SOFA file. By default is disabled.

anglestep

Set neighbor search angle step. Only used if option *interpolate* is enabled.

radstep

Set neighbor search radius step. Only used if option *interpolate* is enabled.

Examples

- ⊕ Using ClubFritz6 sofa file:

```
sofalizer=sofa=/path/to/ClubFritz6.sofa:type=freq:radius=1
```

- ⊕ Using ClubFritz12 sofa file and bigger radius with small rotation:

```
sofalizer=sofa=/path/to/ClubFritz12.sofa:type=freq:radius=2:rotation=5
```

- ⊕ Similar as above but with custom speaker positions for front left, front right, back left and back right and also with custom gain:

```
"sofalizer=sofa=/path/to/ClubFritz6.sofa:type=freq:radius=2:speakers=FL 45|FR 315|BL 135|BR 225:gain=2
```

spechnorm

Speech Normalizer.

This filter expands or compresses each half-cycle of audio samples (local set of samples all above or all below zero and between two nearest zero crossings) depending on threshold value, so audio reaches target peak value under conditions controlled by below options.

The filter accepts the following options:

peak, p

Set the expansion target peak value. This specifies the highest allowed absolute amplitude level for the normalized audio input. Default value is 0.95. Allowed range is from 0.0 to 1.0.

expansion, e

Set the maximum expansion factor. Allowed range is from 1.0 to 50.0. Default value is 2.0. This option controls maximum local half-cycle of samples expansion. The maximum expansion would be such that local peak value reaches target peak value but never to surpass it and that ratio between new and previous peak value does not surpass this option value.

compression, c

Set the maximum compression factor. Allowed range is from 1.0 to 50.0. Default value is 2.0. This option controls maximum local half-cycle of samples compression. This option is used only if **threshold** option is set to value greater than 0.0, then in such cases when local peak is lower or same as value set by **threshold** all samples belonging to that peak's half-cycle will be compressed by current compression factor.

threshold, t

Set the threshold value. Default value is 0.0. Allowed range is from 0.0 to 1.0. This option specifies which half-cycles of samples will be compressed and which will be expanded. Any half-cycle samples with their local peak value below or same as this option value will be compressed by current compression factor, otherwise, if greater than threshold value they will be expanded with expansion factor so that it could reach peak target value but never surpass it.

raise, r

Set the expansion raising amount per each half-cycle of samples. Default value is 0.001. Allowed range is from 0.0 to 1.0. This controls how fast expansion factor is raised per each new half-cycle until it reaches **expansion** value. Setting this options too high may lead to distortions.

fall, f

Set the compression raising amount per each half-cycle of samples. Default value is 0.001. Allowed range is from 0.0 to 1.0. This controls how fast compression factor is raised per each new half-cycle until it reaches **compression** value.

channels, h

Specify which channels to filter, by default all available channels are filtered.

invert, i

Enable inverted filtering, by default is disabled. This inverts interpretation of **threshold** option. When enabled any half-cycle of samples with their local peak value below or same as **threshold** option will be expanded otherwise it will be compressed.

link, l

Link channels when calculating gain applied to each filtered channel sample, by default is disabled. When disabled each filtered channel gain calculation is independent, otherwise when this option is enabled the minimum of all possible gains for each filtered channel is used.

rms, m

Set the expansion target RMS value. This specifies the highest allowed RMS level for the normalized audio input. Default value is 0.0, thus disabled. Allowed range is from 0.0 to 1.0.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Weak and slow amplification:

```
speechnorm=e=3:r=0.00001:l=1
```

- ⊕ Moderate and slow amplification:

```
speechnorm=e=6.25:r=0.00001:l=1
```

- ⊕ Strong and fast amplification:

```
speechnorm=e=12.5:r=0.0001:l=1
```

- ⊕ Very strong and fast amplification:

```
speechnorm=e=25:r=0.0001:l=1
```

- ⊕ Extreme and fast amplification:

```
speechnorm=e=50:r=0.0001:l=1
```

stereotools

This filter has some handy utilities to manage stereo signals, for converting M/S stereo recordings to L/R signal while having control over the parameters or spreading the stereo image of master track.

The filter accepts the following options:

level_in

Set input level before filtering for both channels. Defaults is 1. Allowed range is from 0.015625 to 64.

level_out

Set output level after filtering for both channels. Defaults is 1. Allowed range is from 0.015625 to 64.

balance_in

Set input balance between both channels. Default is 0. Allowed range is from -1 to 1.

balance_out

Set output balance between both channels. Default is 0. Allowed range is from -1 to 1.

softclip

Enable softclipping. Results in analog distortion instead of harsh digital 0dB clipping. Disabled by default.

mutel

Mute the left channel. Disabled by default.

muter

Mute the right channel. Disabled by default.

phasel

Change the phase of the left channel. Disabled by default.

phaser

Change the phase of the right channel. Disabled by default.

mode

Set stereo mode. Available values are:

lr>lr

Left/Right to Left/Right, this is default.

lr>ms

Left/Right to Mid/Side.

ms>lr

Mid/Side to Left/Right.

lr>ll Left/Right to Left/Left.

lr>rr

Left/Right to Right/Right.

lr>l+r

Left/Right to Left + Right.

lr>rl

Left/Right to Right/Left.

ms>ll

Mid/Side to Left/Left.

ms>rr

Mid/Side to Right/Right.

ms>rl

Mid/Side to Right/Left.

lr>l-r

Left/Right to Left - Right.

slev Set level of side signal. Default is 1. Allowed range is from 0.015625 to 64.

sbal Set balance of side signal. Default is 0. Allowed range is from -1 to 1.

mlev

Set level of the middle signal. Default is 1. Allowed range is from 0.015625 to 64.

mpan

Set middle signal pan. Default is 0. Allowed range is from -1 to 1.

base

Set stereo base between mono and inversed channels. Default is 0. Allowed range is from -1 to 1.

delay

Set delay in milliseconds how much to delay left from right channel and vice versa. Default is 0. Allowed range is from -20 to 20.

slevel

Set S/C level. Default is 1. Allowed range is from 1 to 100.

phase

Set the stereo phase in degrees. Default is 0. Allowed range is from 0 to 360.

bmode_in, bmode_out

Set balance mode for balance_in/balance_out option.

Can be one of the following:

balance

Classic balance mode. Attenuate one channel at time. Gain is raised up to 1.

amplitude

Similar as classic mode above but gain is raised up to 2.

power

Equal power distribution, from -6dB to +6dB range.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Apply karaoke like effect:

```
stereotools=mlev=0.015625
```

- ⊕ Convert M/S signal to L/R:

```
"stereotools=mode=ms>lr"
```


stereowiden

This filter enhance the stereo effect by suppressing signal common to both channels and by delaying the signal of left into right and vice versa, thereby widening the stereo effect.

The filter accepts the following options:

delay

Time in milliseconds of the delay of left signal into right and vice versa. Default is 20 milliseconds.

feedback

Amount of gain in delayed signal into right and vice versa. Gives a delay effect of left signal in right output and vice versa which gives widening effect. Default is 0.3.

crossfeed

Cross feed of left into right with inverted phase. This helps in suppressing the mono. If the value is 1 it will cancel all the signal common to both channels. Default is 0.3.

drymix

Set level of input signal of original channel. Default is 0.8.

Commands

This filter supports the all above options except "delay" as **commands**.

superequalizer

Apply 18 band equalizer.

The filter accepts the following options:

1b Set 65Hz band gain.

2b Set 92Hz band gain.

3b Set 131Hz band gain.

4b Set 185Hz band gain.

5b Set 262Hz band gain.

6b Set 370Hz band gain.

7b Set 523Hz band gain.

8b Set 740Hz band gain.

9b Set 1047Hz band gain.

10b Set 1480Hz band gain.

11b Set 2093Hz band gain.

12b Set 2960Hz band gain.

13b Set 4186Hz band gain.

14b Set 5920Hz band gain.

15b Set 8372Hz band gain.

16b Set 11840Hz band gain.

17b Set 16744Hz band gain.

18b Set 20000Hz band gain.

surround

Apply audio surround upmix filter.

This filter allows to produce multichannel output from audio stream.

The filter accepts the following options:

chl_out

Set output channel layout. By default, this is *5.1*.

See **the Channel Layout section in the ffmpeg-utils(1) manual** for the required syntax.

chl_in

Set input channel layout. By default, this is *stereo*.

See **the Channel Layout section in the ffmpeg-utils(1) manual** for the required syntax.

level_in

Set input volume level. By default, this is *1*.

level_out

Set output volume level. By default, this is *1*.

lfe Enable LFE channel output if output channel layout has it. By default, this is enabled.

lfe_low

Set LFE low cut off frequency. By default, this is *128* Hz.

lfe_high

Set LFE high cut off frequency. By default, this is *256* Hz.

lfe_mode

Set LFE mode, can be *add* or *sub*. Default is *add*. In *add* mode, LFE channel is created from input audio and added to output. In *sub* mode, LFE channel is created from input audio and added to output but also all non-LFE output channels are subtracted with output LFE channel.

smooth

Set temporal smoothness strength, used to gradually change factors when transforming stereo sound in time. Allowed range is from *0.0* to *1.0*. Useful to improve output quality with *focus* option values greater than *0.0*. Default is *0.0*. Only values inside this range and without edges are effective.

angle

Set angle of stereo surround transform, Allowed range is from *0* to *360*. Default is *90*.

focus

Set focus of stereo surround transform, Allowed range is from *-1* to *1*. Default is *0*.

fc_in

Set front center input volume. By default, this is *1*.

fc_out

Set front center output volume. By default, this is *1*.

fl_in

Set front left input volume. By default, this is *1*.

fl_out

Set front left output volume. By default, this is *1*.

fr_in

Set front right input volume. By default, this is *1*.

fr_out

Set front right output volume. By default, this is *1*.

sl_in

Set side left input volume. By default, this is *1*.

sl_out

Set side left output volume. By default, this is *1*.

sr_in

Set side right input volume. By default, this is *1*.

sr_out

Set side right output volume. By default, this is *1*.

bl_in

Set back left input volume. By default, this is *1*.

bl_out

Set back left output volume. By default, this is *1*.

br_in

Set back right input volume. By default, this is *1*.

br_out

Set back right output volume. By default, this is *1*.

bc_in

Set back center input volume. By default, this is *1*.

bc_out

Set back center output volume. By default, this is *1*.

lfe_in

Set LFE input volume. By default, this is *1*.

lfe_out

Set LFE output volume. By default, this is *1*.

allx Set spread usage of stereo image across X axis for all channels. Allowed range is from *-1* to *15*.
By default this value is negative *-1*, and thus unused.

ally Set spread usage of stereo image across Y axis for all channels. Allowed range is from *-1* to *15*.
By default this value is negative *-1*, and thus unused.

fcx, flx, frx, blx, brx, slx, srx, bcx

Set spread usage of stereo image across X axis for each channel. Allowed range is from *0.06* to *15*. By default this value is *0.5*.

fcy, fly, fry, bly, bry, sly, sry, bcy

Set spread usage of stereo image across Y axis for each channel. Allowed range is from *0.06* to *15*. By default this value is *0.5*.

win_size

Set window size. Allowed range is from *1024* to *65536*. Default size is *4096*.

win_func

Set window function.

It accepts the following values:

rect

bartlett

hann, hanning

hamming

blackman

welch

flattop

bharris

bnutall

bhann

sine

nutall

lanczos
gauss
tukey
dolph
cauchy
parzen
poisson
bohman
kaiser

Default is "hann".

overlap

Set window overlap. If set to 1, the recommended overlap for selected window function will be picked. Default is 0.5.

tiltshelf

Boost or cut the lower frequencies and cut or boost higher frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

gain, g

Give the gain at 0 Hz. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

frequency, f

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 3000 Hz.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Determine how steep is the filter's shelf transition.

poles, p

Set number of poles. Default is 2.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports some options as **commands**.

treble, highshelf

Boost or cut treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

gain, g

Give the gain at whichever is the lower of ~22 kHz and the Nyquist frequency. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

frequency, f

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 3000 Hz.

width_type, t

Set method to specify band-width of filter.

h Hz

q Q-Factor

o octave

s slope

k kHz

width, w

Determine how steep is the filter's shelf transition.

poles, p

Set number of poles. Default is 2.

mix, m

How much to use filtered signal in output. Default is 1. Range is between 0 and 1.

channels, c

Specify which channels to filter, by default all available are filtered.

normalize, n

Normalize biquad coefficients, by default is disabled. Enabling it will normalize magnitude response at DC to 0dB.

transform, a

Set transform type of IIR filter.

di

dii

tdi

tdii

latt

svf

zdf

precision, r

Set precision of filtering.

auto Pick automatic sample format depending on surround filters.

s16 Always use signed 16-bit.

s32 Always use signed 32-bit.

f32 Always use float 32-bit.

f64 Always use float 64-bit.

block_size, b

Set block size used for reverse IIR processing. If this value is set to high enough value (higher than impulse response length truncated when reaches near zero values) filtering will become linear phase otherwise if not big enough it will just produce nasty artifacts.

Note that filter delay will be exactly this many samples when set to non-zero value.

Commands

This filter supports the following commands:

frequency, f

Change treble frequency. Syntax for the command is : "*frequency*"

width_type, t

Change treble width_type. Syntax for the command is : "*width_type*"

width, w

Change treble width. Syntax for the command is : "*width*"

gain, g

Change treble gain. Syntax for the command is : "*gain*"

mix, m

Change treble mix. Syntax for the command is : "*mix*"

tremolo

Sinusoidal amplitude modulation.

The filter accepts the following options:

f Modulation frequency in Hertz. Modulation frequencies in the subharmonic range (20 Hz or lower) will result in a tremolo effect. This filter may also be used as a ring modulator by specifying a modulation frequency higher than 20 Hz. Range is 0.1 - 20000.0. Default value is 5.0 Hz.

d Depth of modulation as a percentage. Range is 0.0 - 1.0. Default value is 0.5.

vibrato

Sinusoidal phase modulation.

The filter accepts the following options:

f Modulation frequency in Hertz. Range is 0.1 - 20000.0. Default value is 5.0 Hz.

d Depth of modulation as a percentage. Range is 0.0 - 1.0. Default value is 0.5.

virtualbass

Apply audio Virtual Bass filter.

This filter accepts stereo input and produce stereo with LFE (2.1) channels output. The newly produced LFE channel have enhanced virtual bass originally obtained from both stereo channels. This filter outputs front left and front right channels unchanged as available in stereo input.

The filter accepts the following options:

cutoff

Set the virtual bass cutoff frequency. Default value is 250 Hz. Allowed range is from 100 to 500 Hz.

strength

Set the virtual bass strength. Allowed range is from 0.5 to 3. Default value is 3.

volume

Adjust the input audio volume.

It accepts the following parameters:

volume

Set audio volume expression.

Output values are clipped to the maximum value.

The output audio volume is given by the relation:

$$\langle \text{output_volume} \rangle = \langle \text{volume} \rangle * \langle \text{input_volume} \rangle$$

The default value for *volume* is "1.0".

precision

This parameter represents the mathematical precision.

It determines which input sample formats will be allowed, which affects the precision of the volume scaling.

fixed

8-bit fixed-point; this limits input sample format to U8, S16, and S32.

float

32-bit floating-point; this limits input sample format to FLT. (default)

double

64-bit floating-point; this limits input sample format to DBL.

replaygain

Choose the behaviour on encountering ReplayGain side data in input frames.

drop

Remove ReplayGain side data, ignoring its contents (the default).

ignore

Ignore ReplayGain side data, but leave it in the frame.

track

Prefer the track gain, if present.

album

Prefer the album gain, if present.

replaygain_preamp

Pre-amplification gain in dB to apply to the selected replaygain gain.

Default value for *replaygain_preamp* is 0.0.

replaygain_noclip

Prevent clipping by limiting the gain applied.

Default value for *replaygain_noclip* is 1.

eval Set when the volume expression is evaluated.

It accepts the following values:

once

only evaluate expression once during the filter initialization, or when the **volume** command is sent

frame

evaluate expression for each incoming frame

Default value is **once**.

The volume expression can contain the following parameters.

n frame number (starting at zero)

nb_channels

number of channels

nb_consumed_samples

number of samples consumed by the filter

nb_samples

number of samples in the current frame

pos original frame position in the file

pts frame PTS

sample_rate

sample rate

startpts

PTS at start of stream

startt

time at start of stream

t frame time

tb timestamp timebase

volume

last set volume value

Note that when **eval** is set to **once** only the *sample_rate* and *tb* variables are available, all other variables will evaluate to NAN.

Commands

This filter supports the following commands:

volume

Modify the volume expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

Examples

- ⊕ Halve the input audio volume:

```
volume=volume=0.5
volume=volume=1/2
volume=volume=-6.0206dB
```

In all the above example the named key for **volume** can be omitted, for example like in:

```
volume=0.5
```

- ⊕ Increase input audio power by 6 decibels using fixed-point precision:

```
volume=volume=6dB:precision=fixed
```

- ⊕ Fade volume after time 10 with an annihilation period of 5 seconds:

```
volume='if(lt(t,10),1,max(1-(t-10)/5,0))':eval=frame
```

volumedetect

Detect the volume of the input video.

The filter has no parameters. It supports only 16-bit signed integer samples, so the input will be converted when needed. Statistics about the volume will be printed in the log when the input stream

end is reached.

In particular it will show the mean volume (root mean square), maximum volume (on a per-sample basis), and the beginning of a histogram of the registered volume values (from the maximum value to a cumulated 1/1000 of the samples).

All volumes are in decibels relative to the maximum PCM value.

Examples

Here is an excerpt of the output:

```
[Parsed_volumedetect_0 0xa23120] mean_volume: -27 dB
[Parsed_volumedetect_0 0xa23120] max_volume: -4 dB
[Parsed_volumedetect_0 0xa23120] histogram_4db: 6
[Parsed_volumedetect_0 0xa23120] histogram_5db: 62
[Parsed_volumedetect_0 0xa23120] histogram_6db: 286
[Parsed_volumedetect_0 0xa23120] histogram_7db: 1042
[Parsed_volumedetect_0 0xa23120] histogram_8db: 2551
[Parsed_volumedetect_0 0xa23120] histogram_9db: 4609
[Parsed_volumedetect_0 0xa23120] histogram_10db: 8409
```

It means that:

- ⊕ The mean square energy is approximately -27 dB, or $10^{-2.7}$.
- ⊕ The largest sample is at -4 dB, or more precisely between -4 dB and -5 dB.
- ⊕ There are 6 samples at -4 dB, 62 at -5 dB, 286 at -6 dB, etc.

In other words, raising the volume by +4 dB does not cause any clipping, raising it by +5 dB causes clipping for 6 samples, etc.

AUDIO SOURCES

Below is a description of the currently available audio sources.

abuffer

Buffer audio frames, and make them available to the filter chain.

This source is mainly intended for a programmatic use, in particular through the interface defined in

libavfilter/buffersrc.h.

It accepts the following parameters:

time_base

The timebase which will be used for timestamps of submitted frames. It must be either a floating-point number or in *numerator/denominator* form.

sample_rate

The sample rate of the incoming audio buffers.

sample_fmt

The sample format of the incoming audio buffers. Either a sample format name or its corresponding integer representation from the enum AVSampleFormat in *libavutil/samplefmt.h*

channel_layout

The channel layout of the incoming audio buffers. Either a channel layout name from *channel_layout_map* in *libavutil/channel_layout.c* or its corresponding integer representation from the *AV_CH_LAYOUT_** macros in *libavutil/channel_layout.h*

channels

The number of channels of the incoming audio buffers. If both *channels* and *channel_layout* are specified, then they must be consistent.

Examples

```
abuffer=sample_rate=44100:sample_fmt=s16p:channel_layout=stereo
```

will instruct the source to accept planar 16bit signed stereo at 44100Hz. Since the sample format with name "s16p" corresponds to the number 6 and the "stereo" channel layout corresponds to the value 0x3, this is equivalent to:

```
abuffer=sample_rate=44100:sample_fmt=6:channel_layout=0x3
```

aevalsrc

Generate an audio signal specified by an expression.

This source accepts in input one or more expressions (one for each channel), which are evaluated and used to generate a corresponding audio signal.

This source accepts the following options:

exprs

Set the '|'-separated expressions list for each separate channel. In case the **channel_layout** option is not specified, the selected channel layout depends on the number of provided expressions. Otherwise the last specified expression is applied to the remaining output channels.

channel_layout, c

Set the channel layout. The number of channels in the specified layout must be equal to the number of specified expressions.

duration, d

Set the minimum duration of the sourced audio. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame.

If not specified, or the expressed duration is negative, the audio is supposed to be generated forever.

nb_samples, n

Set the number of samples per channel per each output frame, default to 1024.

sample_rate, s

Specify the sample rate, default to 44100.

Each expression in *exprs* can contain the following constants:

- n** number of the evaluated sample, starting from 0
- t** time of the evaluated sample expressed in seconds, starting from 0
- s** sample rate

Examples

- ⊕ Generate silence:

```
aevalsrc=0
```

- ⊕ Generate a sin signal with frequency of 440 Hz, set sample rate to 8000 Hz:

```
aevalsrc="sin(440*2*PI*t):s=8000"
```

- ⊕ Generate a two channels signal, specify the channel layout (Front Center + Back Center) explicitly:

```
aevalsrc="sin(420*2*PI*t)|cos(430*2*PI*t):c=FC|BC"
```

- ⊕ Generate white noise:

```
aevalsrc="-2+random(0)"
```

- ⊕ Generate an amplitude modulated signal:

```
aevalsrc="sin(10*2*PI*t)*sin(880*2*PI*t)"
```

- ⊕ Generate 2.5 Hz binaural beats on a 360 Hz carrier:

```
aevalsrc="0.1*sin(2*PI*(360-2.5/2)*t) | 0.1*sin(2*PI*(360+2.5/2)*t)"
```

afdelaysrc

Generate a fractional delay FIR coefficients.

The resulting stream can be used with **afir** filter for filtering the audio signal.

The filter accepts the following options:

delay, d

Set the fractional delay. Default is 0.

sample_rate, r

Set the sample rate, default is 44100.

nb_samples, n

Set the number of samples per each frame. Default is 1024.

taps, t

Set the number of filter coefficients in output audio stream. Default value is 0.

channel_layout, c

Specifies the channel layout, and can be a string representing a channel layout. The default value

of *channel_layout* is "stereo".

afirsrc

Generate a FIR coefficients using frequency sampling method.

The resulting stream can be used with **afir** filter for filtering the audio signal.

The filter accepts the following options:

taps, t

Set number of filter coefficients in output audio stream. Default value is 1025.

frequency, f

Set frequency points from where magnitude and phase are set. This must be in non decreasing order, and first element must be 0, while last element must be 1. Elements are separated by white spaces.

magnitude, m

Set magnitude value for every frequency point set by **frequency**. Number of values must be same as number of frequency points. Values are separated by white spaces.

phase, p

Set phase value for every frequency point set by **frequency**. Number of values must be same as number of frequency points. Values are separated by white spaces.

sample_rate, r

Set sample rate, default is 44100.

nb_samples, n

Set number of samples per each frame. Default is 1024.

win_func, w

Set window function. Default is blackman.

anullsrc

The null audio source, return unprocessed audio frames. It is mainly useful as a template and to be employed in analysis / debugging tools, or as the source for filters which ignore the input data (for example the sox synth filter).

This source accepts the following options:

channel_layout, cl

Specifies the channel layout, and can be either an integer or a string representing a channel layout. The default value of *channel_layout* is "stereo".

Check the *channel_layout_map* definition in *libavutil/channel_layout.c* for the mapping between strings and channel layout values.

sample_rate, r

Specifies the sample rate, and defaults to 44100.

nb_samples, n

Set the number of samples per requested frames.

duration, d

Set the duration of the sourced audio. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

If not specified, or the expressed duration is negative, the audio is supposed to be generated forever.

Examples

- ⊕ Set the sample rate to 48000 Hz and the channel layout to AV_CH_LAYOUT_MONO.

```
anullsrc=r=48000:cl=4
```

- ⊕ Do the same operation with a more obvious syntax:

```
anullsrc=r=48000:cl=mono
```

All the parameters need to be explicitly defined.

flite

Synthesize a voice utterance using the libflite library.

To enable compilation of this filter you need to configure FFmpeg with "--enable-libflite".

Note that versions of the flite library prior to 2.0 are not thread-safe.

The filter accepts the following options:

list_voices

If set to 1, list the names of the available voices and exit immediately. Default value is 0.

nb_samples, n

Set the maximum number of samples per frame. Default value is 512.

textfile

Set the filename containing the text to speak.

text Set the text to speak.

voice, v

Set the voice to use for the speech synthesis. Default value is "kal". See also the *list_voices* option.

Examples

- ⊕ Read from file *speech.txt*, and synthesize the text using the standard flite voice:

```
flite=textfile=speech.txt
```

- ⊕ Read the specified text selecting the "slt" voice:

```
flite=text='So fare thee well, poor devil of a Sub-Sub, whose commentator I am':voice=slt
```

- ⊕ Input text to ffmpeg:

```
ffmpeg -f lavfi -i flite=text='So fare thee well, poor devil of a Sub-Sub, whose commentator I am':voice=slt
```

- ⊕ Make *ffplay* speak the specified text, using "flite" and the "lavfi" device:

```
ffplay -f lavfi flite=text='No more be grieved for which that thou hast done.'
```

For more information about libflite, check: <<http://www.festvox.org/flite/>>

anoisesrc

Generate a noise audio signal.

The filter accepts the following options:

sample_rate, r

Specify the sample rate. Default value is 48000 Hz.

amplitude, a

Specify the amplitude (0.0 - 1.0) of the generated audio stream. Default value is 1.0.

duration, d

Specify the duration of the generated audio stream. Not specifying this option results in noise with an infinite length.

color, colour, c

Specify the color of noise. Available noise colors are white, pink, brown, blue, violet and velvet. Default color is white.

seed, s

Specify a value used to seed the PRNG.

nb_samples, n

Set the number of samples per each output frame, default is 1024.

Examples

- ⊕ Generate 60 seconds of pink noise, with a 44.1 kHz sampling rate and an amplitude of 0.5:

```
anoisesrc=d=60:c=pink:r=44100:a=0.5
```

hilbert

Generate odd-tap Hilbert transform FIR coefficients.

The resulting stream can be used with **afir** filter for phase-shifting the signal by 90 degrees.

This is used in many matrix coding schemes and for analytic signal generation. The process is often written as a multiplication by *i* (or *j*), the imaginary unit.

The filter accepts the following options:

sample_rate, s

Set sample rate, default is 44100.

taps, t

Set length of FIR filter, default is 22051.

nb_samples, n

Set number of samples per each frame.

win_func, w

Set window function to be used when generating FIR coefficients.

sinc

Generate a sinc kaiser-windowed low-pass, high-pass, band-pass, or band-reject FIR coefficients.

The resulting stream can be used with **afir** filter for filtering the audio signal.

The filter accepts the following options:

sample_rate, r

Set sample rate, default is 44100.

nb_samples, n

Set number of samples per each frame. Default is 1024.

hp Set high-pass frequency. Default is 0.

lp Set low-pass frequency. Default is 0. If high-pass frequency is lower than low-pass frequency and low-pass frequency is higher than 0 then filter will create band-pass filter coefficients, otherwise band-reject filter coefficients.

phase

Set filter phase response. Default is 50. Allowed range is from 0 to 100.

beta Set Kaiser window beta.

att Set stop-band attenuation. Default is 120dB, allowed range is from 40 to 180 dB.

round

Enable rounding, by default is disabled.

hptaps

Set number of taps for high-pass filter.

lptaps

Set number of taps for low-pass filter.

sine

Generate an audio signal made of a sine wave with amplitude 1/8.

The audio signal is bit-exact.

The filter accepts the following options:

frequency, f

Set the carrier frequency. Default is 440 Hz.

beep_factor, b

Enable a periodic beep every second with frequency *beep_factor* times the carrier frequency. Default is 0, meaning the beep is disabled.

sample_rate, r

Specify the sample rate, default is 44100.

duration, d

Specify the duration of the generated audio stream.

samples_per_frame

Set the number of samples per output frame.

The expression can contain the following constants:

n The (sequential) number of the output audio frame, starting from 0.

pts The PTS (Presentation TimeStamp) of the output audio frame, expressed in *TB* units.

t The PTS of the output audio frame, expressed in seconds.

TB The timebase of the output audio frames.

Default is 1024.

Examples

⊕ Generate a simple 440 Hz sine wave:

```
sine
```


- ⊕ Generate a 220 Hz sine wave with a 880 Hz beep each second, for 5 seconds:

```
sine=220:4:d=5
sine=f=220:b=4:d=5
sine=frequency=220:beep_factor=4:duration=5
```

- ⊕ Generate a 1 kHz sine wave following "1602,1601,1602,1601,1602" NTSC pattern:

```
sine=1000:samples_per_frame='st(0,mod(n,5)); 1602-not(not(eq(ld(0),1)+eq(ld(0),3)))'
```

AUDIO SINKS

Below is a description of the currently available audio sinks.

abuffersink

Buffer audio frames, and make them available to the end of filter chain.

This sink is mainly intended for programmatic use, in particular through the interface defined in *libavfilter/buffersink.h* or the options system.

It accepts a pointer to an AVABufferSinkContext structure, which defines the incoming buffers' formats, to be passed as the opaque parameter to "avfilter_init_filter" for initialization.

anullsink

Null audio sink; do absolutely nothing with the input audio. It is mainly useful as a template and for use in analysis / debugging tools.

VIDEO FILTERS

When you configure your FFmpeg build, you can disable any of the existing filters using "--disable-filters". The configure output will show the video filters included in your build.

Below is a description of the currently available video filters.

addroi

Mark a region of interest in a video frame.

The frame data is passed through unchanged, but metadata is attached to the frame indicating regions of interest which can affect the behaviour of later encoding. Multiple regions can be marked by applying the filter multiple times.

- x** Region distance in pixels from the left edge of the frame.

y Region distance in pixels from the top edge of the frame.

w Region width in pixels.

h Region height in pixels.

The parameters x , y , w and h are expressions, and may contain the following variables:

iw Width of the input frame.

ih Height of the input frame.

qoffset

Quantisation offset to apply within the region.

This must be a real value in the range -1 to +1. A value of zero indicates no quality change. A negative value asks for better quality (less quantisation), while a positive value asks for worse quality (greater quantisation).

The range is calibrated so that the extreme values indicate the largest possible offset - if the rest of the frame is encoded with the worst possible quality, an offset of -1 indicates that this region should be encoded with the best possible quality anyway. Intermediate values are then interpolated in some codec-dependent way.

For example, in 10-bit H.264 the quantisation parameter varies between -12 and 51. A typical `qoffset` value of -1/10 therefore indicates that this region should be encoded with a QP around one-tenth of the full range better than the rest of the frame. So, if most of the frame were to be encoded with a QP of around 30, this region would get a QP of around 24 (an offset of approximately $-1/10 * (51 - -12) = -6.3$). An extreme value of -1 would indicate that this region should be encoded with the best possible quality regardless of the treatment of the rest of the frame - that is, should be encoded at a QP of -12.

clear

If set to true, remove any existing regions of interest marked on the frame before adding the new one.

Examples

⊕ Mark the centre quarter of the frame as interesting.

```
addroi=iw/4:ih/4:iw/2:ih/2:-1/10
```

- ⊕ Mark the 100-pixel-wide region on the left edge of the frame as very uninteresting (to be encoded at much lower quality than the rest of the frame).

```
addroi=0:0:100:ih:+1/5
```

alphaextract

Extract the alpha component from the input as a grayscale video. This is especially useful with the *alphamerge* filter.

alphamerge

Add or replace the alpha component of the primary input with the grayscale value of a second input. This is intended for use with *alphaextract* to allow the transmission or storage of frame sequences that have alpha in a format that doesn't support an alpha channel.

For example, to reconstruct full frames from a normal YUV-encoded video and a separate video created with *alphaextract*, you might use:

```
movie=in_alpha.mkv [alpha]; [in][alpha] alphamerge [out]
```

amplify

Amplify differences between current pixel and pixels of adjacent frames in same pixel location.

This filter accepts the following options:

radius

Set frame radius. Default is 2. Allowed range is from 1 to 63. For example radius of 3 will instruct filter to calculate average of 7 frames.

factor

Set factor to amplify difference. Default is 2. Allowed range is from 0 to 65535.

threshold

Set threshold for difference amplification. Any difference greater or equal to this value will not alter source pixel. Default is 10. Allowed range is from 0 to 65535.

tolerance

Set tolerance for difference amplification. Any difference lower to this value will not alter source pixel. Default is 0. Allowed range is from 0 to 65535.

low Set lower limit for changing source pixel. Default is 65535. Allowed range is from 0 to 65535. This option controls maximum possible value that will decrease source pixel value.

high

Set high limit for changing source pixel. Default is 65535. Allowed range is from 0 to 65535. This option controls maximum possible value that will increase source pixel value.

planes

Set which planes to filter. Default is all. Allowed range is from 0 to 15.

Commands

This filter supports the following **commands** that corresponds to option of same name:

factor

threshold

tolerance

low

high

planes

ass

Same as the **subtitles** filter, except that it doesn't require libavcodec and libavformat to work. On the other hand, it is limited to ASS (Advanced Substation Alpha) subtitles files.

This filter accepts the following option in addition to the common options from the **subtitles** filter:

shaping

Set the shaping engine

Available values are:

auto The default libass shaping engine, which is the best available.

simple

Fast, font-agnostic shaper that can do only substitutions

complex

Slower shaper using OpenType for substitutions and positioning

The default is "auto".

atadenoise

Apply an Adaptive Temporal Averaging Denoiser to the video input.

The filter accepts the following options:

0a Set threshold A for 1st plane. Default is 0.02. Valid range is 0 to 0.3.

0b Set threshold B for 1st plane. Default is 0.04. Valid range is 0 to 5.

1a Set threshold A for 2nd plane. Default is 0.02. Valid range is 0 to 0.3.

1b Set threshold B for 2nd plane. Default is 0.04. Valid range is 0 to 5.

2a Set threshold A for 3rd plane. Default is 0.02. Valid range is 0 to 0.3.

2b Set threshold B for 3rd plane. Default is 0.04. Valid range is 0 to 5.

Threshold A is designed to react on abrupt changes in the input signal and threshold B is designed to react on continuous changes in the input signal.

s Set number of frames filter will use for averaging. Default is 9. Must be odd number in range [5, 129].

p Set what planes of frame filter will use for averaging. Default is all.

a Set what variant of algorithm filter will use for averaging. Default is "p" parallel. Alternatively can be set to "s" serial.

Parallel can be faster than serial, while other way around is never true. Parallel will abort early on first change being greater than thresholds, while serial will continue processing other side of frames if they are equal or below thresholds.

0s

1s

2s Set sigma for 1st plane, 2nd plane or 3rd plane. Default is 32767. Valid range is from 0 to 32767. This option controls weight for each pixel in radius defined by size. Default value means every pixel have same weight. Setting this option to 0 effectively disables filtering.

Commands

This filter supports same **commands** as options except option "s". The command accepts the same syntax of the corresponding option.

avgblur

Apply average blur filter.

The filter accepts the following options:

sizeX

Set horizontal radius size.

planes

Set which planes to filter. By default all planes are filtered.

sizeY

Set vertical radius size, if zero it will be same as "sizeX". Default is 0.

Commands

This filter supports same commands as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

backgroundkey

Turns a static background into transparency.

The filter accepts the following option:

threshold

Threshold for scene change detection.

similarity

Similarity percentage with the background.

blend

Set the blend amount for pixels that are not similar.

Commands

This filter supports the all above options as **commands**.

bbox

Compute the bounding box for the non-black pixels in the input frame luminance plane.

This filter computes the bounding box containing all the pixels with a luminance value greater than the minimum allowed value. The parameters describing the bounding box are printed on the filter log.

The filter accepts the following option:

min_val

Set the minimal luminance value. Default is 16.

Commands

This filter supports the all above options as **commands**.

bilateral

Apply bilateral filter, spatial smoothing while preserving edges.

The filter accepts the following options:

sigmaS

Set sigma of gaussian function to calculate spatial weight. Allowed range is 0 to 512. Default is 0.1.

sigmaR

Set sigma of gaussian function to calculate range weight. Allowed range is 0 to 1. Default is 0.1.

planes

Set planes to filter. Default is first only.

Commands

This filter supports the all above options as **commands**.

bilateral_cuda

CUDA accelerated bilateral filter, an edge preserving filter. This filter is mathematically accurate

thanks to the use of GPU acceleration. For best output quality, use one to one chroma subsampling, i.e. yuv444p format.

The filter accepts the following options:

sigmaS

Set sigma of gaussian function to calculate spatial weight, also called sigma space. Allowed range is 0.1 to 512. Default is 0.1.

sigmaR

Set sigma of gaussian function to calculate color range weight, also called sigma color. Allowed range is 0.1 to 512. Default is 0.1.

window_size

Set window size of the bilateral function to determine the number of neighbours to loop on. If the number entered is even, one will be added automatically. Allowed range is 1 to 255. Default is 1.

Examples

- ⊕ Apply the bilateral filter on a video.

```
./ffmpeg -v verbose \
-hwaccel cuda -hwaccel_output_format cuda -i input.mp4 \
-init_hw_device cuda \
-filter_complex \
" \
[0:v]scale_cuda=format=yuv444p[scaled_video];
[scaled_video]bilateral_cuda=window_size=9:sigmaS=3.0:sigmaR=50.0" \
-an -sn -c:v h264_nvenc -cq 20 out.mp4
```

bitplanenoise

Show and measure bit plane noise.

The filter accepts the following options:

bitplane

Set which plane to analyze. Default is 1.

filter

Filter out noisy pixels from "bitplane" set above. Default is disabled.

blackdetect

Detect video intervals that are (almost) completely black. Can be useful to detect chapter transitions, commercials, or invalid recordings.

The filter outputs its detection analysis to both the log as well as frame metadata. If a black segment of at least the specified minimum duration is found, a line with the start and end timestamps as well as duration is printed to the log with level "info". In addition, a log line with level "debug" is printed per frame showing the black amount detected for that frame.

The filter also attaches metadata to the first frame of a black segment with key "lavfi.black_start" and to the first frame after the black segment ends with key "lavfi.black_end". The value is the frame's timestamp. This metadata is added regardless of the minimum duration specified.

The filter accepts the following options:

black_min_duration, d

Set the minimum detected black duration expressed in seconds. It must be a non-negative floating point number.

Default value is 2.0.

picture_black_ratio_th, pic_th

Set the threshold for considering a picture "black". Express the minimum value for the ratio:

$$\langle \text{nb_black_pixels} \rangle / \langle \text{nb_pixels} \rangle$$

for which a picture is considered black. Default value is 0.98.

pixel_black_th, pix_th

Set the threshold for considering a pixel "black".

The threshold expresses the maximum pixel luminance value for which a pixel is considered "black". The provided value is scaled according to the following equation:

$$\langle \text{absolute_threshold} \rangle = \langle \text{luminance_minimum_value} \rangle + \langle \text{pixel_black_th} \rangle * \langle \text{luminance_range_size} \rangle$$

luminance_range_size and *luminance_minimum_value* depend on the input video format, the range is [0-255] for YUV full-range formats and [16-235] for YUV non full-range formats.

Default value is 0.10.

The following example sets the maximum pixel threshold to the minimum value, and detects only black intervals of 2 or more seconds:

```
blackdetect=d=2:pix_th=0.00
```

blackframe

Detect frames that are (almost) completely black. Can be useful to detect chapter transitions or commercials. Output lines consist of the frame number of the detected frame, the percentage of blackness, the position in the file if known or -1 and the timestamp in seconds.

In order to display the output lines, you need to set the loglevel at least to the AV_LOG_INFO value.

This filter exports frame metadata "lavfi.blackframe.pblack". The value represents the percentage of pixels in the picture that are below the threshold value.

It accepts the following parameters:

amount

The percentage of the pixels that have to be below the threshold; it defaults to 98.

threshold, thresh

The threshold below which a pixel value is considered black; it defaults to 32.

blend

Blend two video frames into each other.

The "blend" filter takes two input streams and outputs one stream, the first input is the "top" layer and second input is "bottom" layer. By default, the output terminates when the longest input terminates.

The "tblend" (time blend) filter takes two consecutive frames from one single stream, and outputs the result obtained by blending the new frame on top of the old frame.

A description of the accepted options follows.

c0_mode

c1_mode

c2_mode

c3_mode

all_mode

Set blend mode for specific pixel component or all pixel components in case of *all_mode*. Default

value is "normal".

Available values for component modes are:

addition
and
average
bleach
burn
darken
difference
divide
dodge
exclusion
extremity
freeze
geometric
glow
grainextract
grainmerge
hardlight
hardmix
hardoverlay
harmonic
heat
interpolate
lighten
linearlight
multiply
multiply128
negation
normal
or
overlay
phoenix
pinlight
reflect
screen
softdifference
softlight

stain
subtract
vividlight
xor

c0_opacity
c1_opacity
c2_opacity
c3_opacity
all_opacity

Set blend opacity for specific pixel component or all pixel components in case of *all_opacity*.
Only used in combination with pixel component blend modes.

c0_expr
c1_expr
c2_expr
c3_expr
all_expr

Set blend expression for specific pixel component or all pixel components in case of *all_expr*.
Note that related mode options will be ignored if those are set.

The expressions can use the following variables:

N The sequential number of the filtered frame, starting from 0.

X
Y the coordinates of the current sample

W
H the width and height of currently filtered plane

SW
SH Width and height scale for the plane being filtered. It is the ratio between the dimensions of the current plane to the luma plane, e.g. for a "yuv420p" frame, the values are "1,1" for the luma plane and "0.5,0.5" for the chroma planes.

T Time of the current frame, expressed in seconds.

TOP, A
Value of pixel component at current location for first video frame (top layer).

BOTTOM, B

Value of pixel component at current location for second video frame (bottom layer).

The "blend" filter also supports the **framesync** options.

Examples

- ⊕ Apply transition from bottom layer to top layer in first 10 seconds:

```
blend=all_expr='A*(if(gte(T,10),1,T/10))+B*(1-(if(gte(T,10),1,T/10)))'
```

- ⊕ Apply linear horizontal transition from top layer to bottom layer:

```
blend=all_expr='A*(X/W)+B*(1-X/W)'
```

- ⊕ Apply 1x1 checkerboard effect:

```
blend=all_expr='if(eq(mod(X,2),mod(Y,2)),A,B)'
```

- ⊕ Apply uncover left effect:

```
blend=all_expr='if(gte(N*SW+X,W),A,B)'
```

- ⊕ Apply uncover down effect:

```
blend=all_expr='if(gte(Y-N*SH,0),A,B)'
```

- ⊕ Apply uncover up-left effect:

```
blend=all_expr='if(gte(T*SH*40+Y,H)*gte((T*40*SW+X)*W/H,W),A,B)'
```

- ⊕ Split diagonally video and shows top and bottom layer on each side:

```
blend=all_expr='if(gt(X,Y*(W/H)),A,B)'
```

- ⊕ Display differences between the current and the previous frame:

```
tblend=all_mode=grainextract
```

Commands

This filter supports same **commands** as options.

blockdetect

Determines blockiness of frames without altering the input frames.

Based on Remco Muijs and Ihor Kirenko: "A no-reference blocking artifact measure for adaptive video processing." 2005 13th European signal processing conference.

The filter accepts the following options:

period_min

period_max

Set minimum and maximum values for determining pixel grids (periods). Default values are [3,24].

planes

Set planes to filter. Default is first only.

Examples

- ⊕ Determine blockiness for the first plane and search for periods within [8,32]:

```
blockdetect=period_min=8:period_max=32:planes=1
```

blurdetect

Determines blurriness of frames without altering the input frames.

Based on Marziliano, Pina, et al. "A no-reference perceptual blur metric." Allows for a block-based abbreviation.

The filter accepts the following options:

low

high

Set low and high threshold values used by the Canny thresholding algorithm.

The high threshold selects the "strong" edge pixels, which are then connected through 8-connectivity with the "weak" edge pixels selected by the low threshold.

low and *high* threshold values must be chosen in the range [0,1], and *low* should be lesser or equal

to *high*.

Default value for *low* is "20/255", and default value for *high* is "50/255".

radius

Define the radius to search around an edge pixel for local maxima.

block_pct

Determine blurriness only for the most significant blocks, given in percentage.

block_width

Determine blurriness for blocks of width *block_width*. If set to any value smaller 1, no blocks are used and the whole image is processed as one no matter of *block_height*.

block_height

Determine blurriness for blocks of height *block_height*. If set to any value smaller 1, no blocks are used and the whole image is processed as one no matter of *block_width*.

planes

Set planes to filter. Default is first only.

Examples

- ⊕ Determine blur for 80% of most significant 32x32 blocks:

```
blurdetect=block_width=32:block_height=32:block_pct=80
```

bm3d

Denoise frames using Block-Matching 3D algorithm.

The filter accepts the following options.

sigma

Set denoising strength. Default value is 1. Allowed range is from 0 to 999.9. The denoising algorithm is very sensitive to sigma, so adjust it according to the source.

block

Set local patch size. This sets dimensions in 2D.

bstep

Set sliding step for processing blocks. Default value is 4. Allowed range is from 1 to 64. Smaller values allows processing more reference blocks and is slower.

group

Set maximal number of similar blocks for 3rd dimension. Default value is 1. When set to 1, no block matching is done. Larger values allows more blocks in single group. Allowed range is from 1 to 256.

range

Set radius for search block matching. Default is 9. Allowed range is from 1 to INT32_MAX.

mstep

Set step between two search locations for block matching. Default is 1. Allowed range is from 1 to 64. Smaller is slower.

thmse

Set threshold of mean square error for block matching. Valid range is 0 to INT32_MAX.

hdthr

Set thresholding parameter for hard thresholding in 3D transformed domain. Larger values results in stronger hard-thresholding filtering in frequency domain.

estim

Set filtering estimation mode. Can be "basic" or "final". Default is "basic".

ref If enabled, filter will use 2nd stream for block matching. Default is disabled for "basic" value of *estim* option, and always enabled if value of *estim* is "final".

planes

Set planes to filter. Default is all available except alpha.

Examples

- ⊕ Basic filtering with bm3d:

```
bm3d=sigma=3:block=4:bstep=2:group=1:estim=basic
```

- ⊕ Same as above, but filtering only luma:

```
bm3d=sigma=3:block=4:bstep=2:group=1:estim=basic:planes=1
```


- ⊕ Same as above, but with both estimation modes:

```
split[a][b],[a]bm3d=sigma=3:block=4:bstep=2:group=1:estim=basic[a],[b][a]bm3d=sigma=3:block=4:bstep=
```

- ⊕ Same as above, but prefilter with **nlmeans** filter instead:

```
split[a][b],[a]nlmeans=s=3:r=7:p=3[a],[b][a]bm3d=sigma=3:block=4:bstep=2:group=16:estim=final:ref=1
```

boxblur

Apply a boxblur algorithm to the input video.

It accepts the following parameters:

luma_radius, lr
luma_power, lp
chroma_radius, cr
chroma_power, cp
alpha_radius, ar
alpha_power, ap

A description of the accepted options follows.

luma_radius, lr
chroma_radius, cr
alpha_radius, ar

Set an expression for the box radius in pixels used for blurring the corresponding input plane.

The radius value must be a non-negative number, and must not be greater than the value of the expression "min(w,h)/2" for the luma and alpha planes, and of "min(cw,ch)/2" for the chroma planes.

Default value for **luma_radius** is "2". If not specified, **chroma_radius** and **alpha_radius** default to the corresponding value set for **luma_radius**.

The expressions can contain the following constants:

w
h The input width and height in pixels.
cw

ch The input chroma image width and height in pixels.

hsub

vsub

The horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p", *hsub* is 2 and *vsub* is 1.

luma_power, lp

chroma_power, cp

alpha_power, ap

Specify how many times the boxblur filter is applied to the corresponding plane.

Default value for **luma_power** is 2. If not specified, **chroma_power** and **alpha_power** default to the corresponding value set for **luma_power**.

A value of 0 will disable the effect.

Examples

- ⊕ Apply a boxblur filter with the luma, chroma, and alpha radii set to 2:

```
boxblur=luma_radius=2:luma_power=1
boxblur=2:1
```

- ⊕ Set the luma radius to 2, and alpha and chroma radius to 0:

```
boxblur=2:1:cr=0:ar=0
```

- ⊕ Set the luma and chroma radii to a fraction of the video dimension:

```
boxblur=luma_radius=min(h,w)/10:luma_power=1:chroma_radius=min(cw,ch)/10:chroma_power=1
```

bwdif

Deinterlace the input video ("bwdif" stands for "Bob Weaver Deinterlacing Filter").

Motion adaptive deinterlacing based on yadif with the use of w3fdif and cubic interpolation algorithms. It accepts the following parameters:

mode

The interlacing mode to adopt. It accepts one of the following values:

0, send_frame

Output one frame for each frame.

1, send_field

Output one frame for each field.

The default value is "send_field".

parity

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

0, tff

Assume the top field is first.

1, bff

Assume the bottom field is first.

-1, auto

Enable automatic detection of field parity.

The default value is "auto". If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

deint

Specify which frames to deinterlace. Accepts one of the following values:

0, all

Deinterlace all frames.

1, interlaced

Only deinterlace frames marked as interlaced.

The default value is "all".

cas

Apply Contrast Adaptive Sharpen filter to video stream.

The filter accepts the following options:

strength

Set the sharpening strength. Default value is 0.

planes

Set planes to filter. Default value is to filter all planes except alpha plane.

Commands

This filter supports same **commands** as options.

chromahold

Remove all color information for all colors except for certain one.

The filter accepts the following options:

color

The color which will not be replaced with neutral chroma.

similarity

Similarity percentage with the above color. 0.01 matches only the exact key color, while 1.0 matches everything.

blend

Blend percentage. 0.0 makes pixels either fully gray, or not gray at all. Higher values result in more preserved color.

yuv Signals that the color passed is already in YUV instead of RGB.

Literal colors like "green" or "red" don't make sense with this enabled anymore. This can be used to pass exact YUV values as hexadecimal numbers.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

chromakey

YUV colorspace color/chroma keying.

The filter accepts the following options:

color

The color which will be replaced with transparency.

similarity

Similarity percentage with the key color.

0.01 matches only the exact key color, while 1.0 matches everything.

blend

Blend percentage.

0.0 makes pixels either fully transparent, or not transparent at all.

Higher values result in semi-transparent pixels, with a higher transparency the more similar the pixels color is to the key color.

yuv Signals that the color passed is already in YUV instead of RGB.

Literal colors like "green" or "red" don't make sense with this enabled anymore. This can be used to pass exact YUV values as hexadecimal numbers.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

Examples

- ⊕ Make every green pixel in the input image transparent:

```
ffmpeg -i input.png -vf chromakey=green out.png
```

- ⊕ Overlay a greenscreen-video on top of a static black background.

```
ffmpeg -f lavfi -i color=c=black:s=1280x720 -i video.mp4 -shortest -filter_complex "[1:v]chromakey=0x70d
```

chromakey_cuda

CUDA accelerated YUV colorspace color/chroma keying.

This filter works like normal chromakey filter but operates on CUDA frames. for more details and parameters see **chromakey**.

Examples

- ⊕ Make all the green pixels in the input video transparent and use it as an overlay for another video:

```
./ffmpeg \
-hwaccel cuda -hwaccel_output_format cuda -i input_green.mp4 \
-hwaccel cuda -hwaccel_output_format cuda -i base_video.mp4 \
-init_hw_device cuda \
-filter_complex \
" \
[0:v]chromakey_cuda=0x25302D:0.1:0.12:1[overlay_video]; \
[1:v]scale_cuda=format=yuv420p[base]; \
[base][overlay_video]overlay_cuda" \
-an -sn -c:v h264_nvenc -cq 20 output.mp4
```

- ⊕ Process two software sources, explicitly uploading the frames:

```
./ffmpeg -init_hw_device cuda=cuda -filter_hw_device cuda \
-f lavfi -i color=size=800x600:color=white,format=yuv420p \
-f lavfi -i yuvtestsrc=size=200x200,format=yuv420p \
-filter_complex \
" \
[0]hwupload[under]; \
[1]hwupload,chromakey_cuda=green:0.1:0.12[over]; \
[under][over]overlay_cuda" \
-c:v hevc_nvenc -cq 18 -preset slow output.mp4
```

chromanr

Reduce chrominance noise.

The filter accepts the following options:

thres

Set threshold for averaging chrominance values. Sum of absolute difference of Y, U and V pixel

components of current pixel and neighbour pixels lower than this threshold will be used in averaging. Luma component is left unchanged and is copied to output. Default value is 30. Allowed range is from 1 to 200.

sizew

Set horizontal radius of rectangle used for averaging. Allowed range is from 1 to 100. Default value is 5.

sizeh

Set vertical radius of rectangle used for averaging. Allowed range is from 1 to 100. Default value is 5.

stepw

Set horizontal step when averaging. Default value is 1. Allowed range is from 1 to 50. Mostly useful to speed-up filtering.

steph

Set vertical step when averaging. Default value is 1. Allowed range is from 1 to 50. Mostly useful to speed-up filtering.

they

Set Y threshold for averaging chrominance values. Set finer control for max allowed difference between Y components of current pixel and neighbour pixels. Default value is 200. Allowed range is from 1 to 200.

threu

Set U threshold for averaging chrominance values. Set finer control for max allowed difference between U components of current pixel and neighbour pixels. Default value is 200. Allowed range is from 1 to 200.

threv

Set V threshold for averaging chrominance values. Set finer control for max allowed difference between V components of current pixel and neighbour pixels. Default value is 200. Allowed range is from 1 to 200.

distance

Set distance type used in calculations.

manhattan

Absolute difference.

euclidean

Difference squared.

Default distance type is manhattan.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

chromashift

Shift chroma pixels horizontally and/or vertically.

The filter accepts the following options:

cbh Set amount to shift chroma-blue horizontally.

cbv Set amount to shift chroma-blue vertically.

crh Set amount to shift chroma-red horizontally.

crv Set amount to shift chroma-red vertically.

edge

Set edge mode, can be *smear*, *default*, or *warp*.

Commands

This filter supports the all above options as **commands**.

ciescope

Display CIE color diagram with pixels overlaid onto it.

The filter accepts the following options:

system

Set color system.

ntsc, 470m

ebu, 470bg

smpte

240m

apple

widergb

cie1931

rec709, hdtv

uhdtv, rec2020

dcip3

cie Set CIE system.

xyy

ucs

luv

gamuts

Set what gamuts to draw.

See "system" option for available values.

size, s

Set ciescope size, by default set to 512.

intensity, i

Set intensity used to map input pixel values to CIE diagram.

contrast

Set contrast used to draw tongue colors that are out of active color system gamut.

corrgamma

Correct gamma displayed on scope, by default enabled.

showwhite

Show white point on CIE diagram, by default disabled.

gamma

Set input gamma. Used only with XYZ input color space.

fill Fill with CIE colors. By default is enabled.

codeview

Visualize information exported by some codecs.

Some codecs can export information through frames using side-data or other means. For example, some MPEG based codecs export motion vectors through the *export_mvs* flag in the codec **flags2** option.

The filter accepts the following option:

block

Display block partition structure using the luma plane.

mv Set motion vectors to visualize.

Available flags for *mv* are:

pf forward predicted MVs of P-frames

bf forward predicted MVs of B-frames

bb backward predicted MVs of B-frames

qp Display quantization parameters using the chroma planes.

mv_type, mvt

Set motion vectors type to visualize. Includes MVs from all frames unless specified by *frame_type* option.

Available flags for *mv_type* are:

fp forward predicted MVs

bp backward predicted MVs

frame_type, ft

Set frame type to visualize motion vectors of.

Available flags for *frame_type* are:

if intra-coded frames (I-frames)

pf predicted frames (P-frames)

bf bi-directionally predicted frames (B-frames)

Examples

- ⊕ Visualize forward predicted MVs of all frames using **ffplay**:

```
ffplay -flags2 +export_mvs input.mp4 -vf codecview=mv_type=fp
```

- ⊕ Visualize multi-directionals MVs of P and B-Frames using **ffplay**:

```
ffplay -flags2 +export_mvs input.mp4 -vf codecview=mv=pf+bf+bb
```

colorbalance

Modify intensity of primary colors (red, green and blue) of input frames.

The filter allows an input frame to be adjusted in the shadows, midtones or highlights regions for the red-cyan, green-magenta or blue-yellow balance.

A positive adjustment value shifts the balance towards the primary color, a negative value towards the complementary color.

The filter accepts the following options:

rs

gs

bs Adjust red, green and blue shadows (darkest pixels).

rm

gm

bm Adjust red, green and blue midtones (medium pixels).

rh

gh

bh Adjust red, green and blue highlights (brightest pixels).

Allowed ranges for options are "[-1.0, 1.0]". Defaults are 0.

pl Preserve lightness when changing color balance. Default is disabled.

Examples

- ⊕ Add red color cast to shadows:

```
colorbalance=rs=.3
```

Commands

This filter supports the all above options as **commands**.

colorcontrast

Adjust color contrast between RGB components.

The filter accepts the following options:

rc Set the red-cyan contrast. Defaults is 0.0. Allowed range is from -1.0 to 1.0.

gm Set the green-magenta contrast. Defaults is 0.0. Allowed range is from -1.0 to 1.0.

by Set the blue-yellow contrast. Defaults is 0.0. Allowed range is from -1.0 to 1.0.

rcw

gmw

byw Set the weight of each "rc", "gm", "by" option value. Default value is 0.0. Allowed range is from 0.0 to 1.0. If all weights are 0.0 filtering is disabled.

pl Set the amount of preserving lightness. Default value is 0.0. Allowed range is from 0.0 to 1.0.

Commands

This filter supports the all above options as **commands**.

colorcorrect

Adjust color white balance selectively for blacks and whites. This filter operates in YUV colorspace.

The filter accepts the following options:

rl Set the red shadow spot. Allowed range is from -1.0 to 1.0. Default value is 0.

bl Set the blue shadow spot. Allowed range is from -1.0 to 1.0. Default value is 0.

rh Set the red highlight spot. Allowed range is from -1.0 to 1.0. Default value is 0.

bh Set the red highlight spot. Allowed range is from -1.0 to 1.0. Default value is 0.

saturation

Set the amount of saturation. Allowed range is from -3.0 to 3.0. Default value is 1.

analyze

If set to anything other than "manual" it will analyze every frame and use derived parameters for filtering output frame.

Possible values are:

manual

average

minmax

median

Default value is "manual".

Commands

This filter supports the all above options as **commands**.

colorchannelmixer

Adjust video input frames by re-mixing color channels.

This filter modifies a color channel by adding the values associated to the other channels of the same pixels. For example if the value to modify is red, the output value will be:

$$\langle \text{red} \rangle = \langle \text{red} \rangle * \langle \text{rr} \rangle + \langle \text{blue} \rangle * \langle \text{rb} \rangle + \langle \text{green} \rangle * \langle \text{rg} \rangle + \langle \text{alpha} \rangle * \langle \text{ra} \rangle$$

The filter accepts the following options:

rr

rg

rb

ra Adjust contribution of input red, green, blue and alpha channels for output red channel. Default is 1 for *rr*, and 0 for *rg*, *rb* and *ra*.

gr

gg

gb

ga Adjust contribution of input red, green, blue and alpha channels for output green channel. Default is 1 for *gg*, and 0 for *gr*, *gb* and *ga*.

br**bg****bb**

ba Adjust contribution of input red, green, blue and alpha channels for output blue channel. Default is 1 for *bb*, and 0 for *br*, *bg* and *ba*.

ar**ag****ab**

aa Adjust contribution of input red, green, blue and alpha channels for output alpha channel. Default is 1 for *aa*, and 0 for *ar*, *ag* and *ab*.

Allowed ranges for options are "[-2.0, 2.0]".

pc Set preserve color mode. The accepted values are:

none

Disable color preserving, this is default.

lum Preserve luminance.

max Preserve max value of RGB triplet.

avg Preserve average value of RGB triplet.

sum Preserve sum value of RGB triplet.

nrm Preserve normalized value of RGB triplet.

pwr Preserve power value of RGB triplet.

pa Set the preserve color amount when changing colors. Allowed range is from "[0.0, 1.0]". Default is 0.0, thus disabled.

Examples

- ⊕ Convert source to grayscale:

```
colorchannelmixer=.3:.4:.3:0:.3:.4:.3:0:.3:.4:.3
```

- ⊕ Simulate sepia tones:

```
colorchannelmixer=.393:.769:.189:0:.349:.686:.168:0:.272:.534:.131
```

Commands

This filter supports the all above options as **commands**.

colorize

Overlay a solid color on the video stream.

The filter accepts the following options:

hue Set the color hue. Allowed range is from 0 to 360. Default value is 0.

saturation

Set the color saturation. Allowed range is from 0 to 1. Default value is 0.5.

lightness

Set the color lightness. Allowed range is from 0 to 1. Default value is 0.5.

mix Set the mix of source lightness. By default is set to 1.0. Allowed range is from 0.0 to 1.0.

Commands

This filter supports the all above options as **commands**.

colorkey

RGB colorspace color keying. This filter operates on 8-bit RGB format frames by setting the alpha component of each pixel which falls within the similarity radius of the key color to 0. The alpha value for pixels outside the similarity radius depends on the value of the blend option.

The filter accepts the following options:

color

Set the color for which alpha will be set to 0 (full transparency). See "**Color**" section in the

ffmpeg-utils manual. Default is "black".

similarity

Set the radius from the key color within which other colors also have full transparency. The computed distance is related to the unit fractional distance in 3D space between the RGB values of the key color and the pixel's color. Range is 0.01 to 1.0. 0.01 matches within a very small radius around the exact key color, while 1.0 matches everything. Default is 0.01.

blend

Set how the alpha value for pixels that fall outside the similarity radius is computed. 0.0 makes pixels either fully transparent or fully opaque. Higher values result in semi-transparent pixels, with greater transparency the more similar the pixel color is to the key color. Range is 0.0 to 1.0. Default is 0.0.

Examples

- ⊕ Make every green pixel in the input image transparent:

```
ffmpeg -i input.png -vf colorkey=green out.png
```

- ⊕ Overlay a greenscreen-video on top of a static background image.

```
ffmpeg -i background.png -i video.mp4 -filter_complex "[1:v]colorkey=0x3BBD1E:0.3:0.2[ckout];[0:v][ckout]"
```

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

colorhold

Remove all color information for all RGB colors except for certain one.

The filter accepts the following options:

color

The color which will not be replaced with neutral gray.

similarity

Similarity percentage with the above color. 0.01 matches only the exact key color, while 1.0 matches everything.

blend

Blend percentage. 0.0 makes pixels fully gray. Higher values result in more preserved color.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

colorlevels

Adjust video input frames using levels.

The filter accepts the following options:

rimin**gimin****bimin****aimin**

Adjust red, green, blue and alpha input black point. Allowed ranges for options are "[-1.0, 1.0]". Defaults are 0.

rimax**gimax****bimax****aimax**

Adjust red, green, blue and alpha input white point. Allowed ranges for options are "[-1.0, 1.0]". Defaults are 1.

Input levels are used to lighten highlights (bright tones), darken shadows (dark tones), change the balance of bright and dark tones.

romin**gomin****bomin****aomin**

Adjust red, green, blue and alpha output black point. Allowed ranges for options are "[0, 1.0]".

Defaults are 0.

romax

gomax

bomax

aomax

Adjust red, green, blue and alpha output white point. Allowed ranges for options are "[0, 1.0]". Defaults are 1.

Output levels allows manual selection of a constrained output level range.

preserve

Set preserve color mode. The accepted values are:

none

Disable color preserving, this is default.

lum Preserve luminance.

max Preserve max value of RGB triplet.

avg Preserve average value of RGB triplet.

sum Preserve sum value of RGB triplet.

nrm Preserve normalized value of RGB triplet.

pwr Preserve power value of RGB triplet.

Examples

⊕ Make video output darker:

```
colorlevels=rimin=0.058:gimin=0.058:bimin=0.058
```

⊕ Increase contrast:

```
colorlevels=rimin=0.039:gimin=0.039:bimin=0.039:rimax=0.96:gimax=0.96:bimax=0.96
```

⊕ Make video output lighter:

```
colorlevels=rimax=0.902:gimax=0.902:bimax=0.902
```

⊕ Increase brightness:

```
colorlevels=romin=0.5:gomin=0.5:bomin=0.5
```

Commands

This filter supports the all above options as **commands**.

colormap

Apply custom color maps to video stream.

This filter needs three input video streams. First stream is video stream that is going to be filtered out. Second and third video stream specify color patches for source color to target color mapping.

The filter accepts the following options:

patch_size

Set the source and target video stream patch size in pixels.

nb_patches

Set the max number of used patches from source and target video stream. Default value is number of patches available in additional video streams. Max allowed number of patches is 64.

type Set the adjustments used for target colors. Can be "relative" or "absolute". Defaults is "absolute".

kernel

Set the kernel used to measure color differences between mapped colors.

The accepted values are:

euclidean

weuclidean

Default is "euclidean".

colormatrix

Convert color matrix.

The filter accepts the following options:

src

dst Specify the source and destination color matrix. Both values must be specified.

The accepted values are:

bt709

BT.709

fcc FCC

bt601

BT.601

bt470

BT.470

bt470bg

BT.470BG

smpte170m

SMPTE-170M

smpte240m

SMPTE-240M

bt2020

BT.2020

For example to convert from BT.601 to SMPTE-240M, use the command:

```
colormatrix=bt601:smpte240m
```

colorspace

Convert colorspace, transfer characteristics or color primaries. Input video needs to have an even size.

The filter accepts the following options:

all Specify all color properties at once.

The accepted values are:

bt470m

BT.470M

bt470bg

BT.470BG

bt601-6-525

BT.601-6 525

bt601-6-625

BT.601-6 625

bt709

BT.709

smpte170m

SMPTE-170M

smpte240m

SMPTE-240M

bt2020

BT.2020

space

Specify output colorspace.

The accepted values are:

bt709

BT.709

fcc FCC

bt470bg

BT.470BG or BT.601-6 625

smpte170m

SMPTE-170M or BT.601-6 525

smpte240m

SMPTE-240M

ycgco

YCgCo

bt2020ncl

BT.2020 with non-constant luminance

trc Specify output transfer characteristics.

The accepted values are:

bt709

BT.709

bt470m

BT.470M

bt470bg

BT.470BG

gamma22

Constant gamma of 2.2

gamma28

Constant gamma of 2.8

smpte170m

SMPTE-170M, BT.601-6 625 or BT.601-6 525

smpte240m

SMPTE-240M

srgb SRGB

iec61966-2-1

iec61966-2-1

iec61966-2-4

iec61966-2-4

xvcc

xvcc

bt2020-10

BT.2020 for 10-bits content

bt2020-12

BT.2020 for 12-bits content

primaries

Specify output color primaries.

The accepted values are:

bt709

BT.709

bt470m

BT.470M

bt470bg

BT.470BG or BT.601-6 625

smpte170m

SMPTE-170M or BT.601-6 525

smpte240m

SMPTE-240M

film film

smpte431

SMPTE-431

smpte432

SMPTE-432

bt2020

BT.2020

jedec-p22

JEDEC P22 phosphors

range

Specify output color range.

The accepted values are:

tv TV (restricted) range**mpeg**

MPEG (restricted) range

pc PC (full) range**jpeg** JPEG (full) range**format**

Specify output color format.

The accepted values are:

yuv420p

YUV 4:2:0 planar 8-bits

yuv420p10

YUV 4:2:0 planar 10-bits

yuv420p12

YUV 4:2:0 planar 12-bits

yuv422p

YUV 4:2:2 planar 8-bits

yuv422p10

YUV 4:2:2 planar 10-bits

yuv422p12

YUV 4:2:2 planar 12-bits

yuv444p

YUV 4:4:4 planar 8-bits

yuv444p10

YUV 4:4:4 planar 10-bits

yuv444p12

YUV 4:4:4 planar 12-bits

fast Do a fast conversion, which skips gamma/primary correction. This will take significantly less CPU, but will be mathematically incorrect. To get output compatible with that produced by the colormatrix filter, use fast=1.

dither

Specify dithering mode.

The accepted values are:

none

No dithering

fsb Floyd-Steinberg dithering**wadapt**

Whitepoint adaptation mode.

The accepted values are:

bradford

Bradford whitepoint adaptation

vonkries

von Kries whitepoint adaptation

identity

identity whitepoint adaptation (i.e. no whitepoint adaptation)

iall Override all input properties at once. Same accepted values as **all**.

ispace

Override input colorspace. Same accepted values as **space**.

iprimaries

Override input color primaries. Same accepted values as **primaries**.

itrc Override input transfer characteristics. Same accepted values as **trc**.

irange

Override input color range. Same accepted values as **range**.

The filter converts the transfer characteristics, color space and color primaries to the specified user values. The output value, if not specified, is set to a default value based on the "all" property. If that property is also not specified, the filter will log an error. The output color range and format default to the same value as the input color range and format. The input transfer characteristics, color space, color primaries and color range should be set on the input data. If any of these are missing, the filter will log an error and no conversion will take place.

For example to convert the input to SMPTE-240M, use the command:

```
colorspace=smp240m
```

colorspace_cuda

CUDA accelerated implementation of the colorspace filter.

It is by no means feature complete compared to the software colorspace filter, and at the current time only supports color range conversion between jpeg/full and mpeg/limited range.

The filter accepts the following options:

range

Specify output color range.

The accepted values are:

tv TV (restricted) range

mpeg

MPEG (restricted) range

pc PC (full) range

jpeg JPEG (full) range

colortemperature

Adjust color temperature in video to simulate variations in ambient color temperature.

The filter accepts the following options:

temperature

Set the temperature in Kelvin. Allowed range is from 1000 to 40000. Default value is 6500 K.

mix Set mixing with filtered output. Allowed range is from 0 to 1. Default value is 1.

pl Set the amount of preserving lightness. Allowed range is from 0 to 1. Default value is 0.

Commands

This filter supports same **commands** as options.

convolution

Apply convolution of 3x3, 5x5, 7x7 or horizontal/vertical up to 49 elements.

The filter accepts the following options:

0m

1m

2m

3m Set matrix for each plane. Matrix is sequence of 9, 25 or 49 signed integers in *square* mode, and from 1 to 49 odd number of signed integers in *row* mode.

0rdiv

1rdiv

2rdiv

3rdiv

Set multiplier for calculated value for each plane. If unset or 0, it will be sum of all matrix elements.

0bias**1bias****2bias****3bias**

Set bias for each plane. This value is added to the result of the multiplication. Useful for making the overall image brighter or darker. Default is 0.0.

0mode**1mode****2mode****3mode**

Set matrix mode for each plane. Can be *square*, *row* or *column*. Default is *square*.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Apply sharpen:

```
convolution="0 -1 0 -1 5 -1 0 -1 0:0 -1 0 -1 5 -1 0 -1 0:0 -1 0 -1 5 -1 0 -1 0:0 -1 0 -1 5 -1 0 -1 0"
```

- ⊕ Apply blur:

```
convolution="1 1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1 1:1/9:1/9:1/9:1/9"
```

- ⊕ Apply edge enhance:

```
convolution="0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0:5:1:1:1:0:128:128:128"
```

- ⊕ Apply edge detect:

```
convolution="0 1 0 1 -4 1 0 1 0:0 1 0 1 -4 1 0 1 0:0 1 0 1 -4 1 0 1 0:0 1 0 1 -4 1 0 1 0:5:5:5:1:0:128:128:128"
```

- ⊕ Apply laplacian edge detector which includes diagonals:

```
convolution="1 1 1 1 -8 1 1 1 1:1 1 1 1 -8 1 1 1 1:1 1 1 1 -8 1 1 1 1:1 1 1 1 -8 1 1 1 1:5:5:5:1:0:128:128:0"
```

- ⊕ Apply emboss:

```
convolution="-2 -1 0 -1 1 1 0 1 2:-2 -1 0 -1 1 1 0 1 2:-2 -1 0 -1 1 1 0 1 2:-2 -1 0 -1 1 1 0 1 2"
```

convolve

Apply 2D convolution of video stream in frequency domain using second stream as impulse.

The filter accepts the following options:

planes

Set which planes to process.

impulse

Set which impulse video frames will be processed, can be *first* or *all*. Default is *all*.

The "convolve" filter also supports the **framesync** options.

copy

Copy the input video source unchanged to the output. This is mainly useful for testing purposes.

coreimage

Video filtering on GPU using Apple's CoreImage API on OSX.

Hardware acceleration is based on an OpenGL context. Usually, this means it is processed by video hardware. However, software-based OpenGL implementations exist which means there is no guarantee for hardware processing. It depends on the respective OSX.

There are many filters and image generators provided by Apple that come with a large variety of options. The filter has to be referenced by its name along with its options.

The coreimage filter accepts the following options:

list_filters

List all available filters and generators along with all their respective options as well as possible minimum and maximum values along with the default values.

```
list_filters=true
```

filter

Specify all filters by their respective name and options. Use *list_filters* to determine all valid filter names and options. Numerical options are specified by a float value and are automatically clamped to their respective value range. Vector and color options have to be specified by a list of

space separated float values. Character escaping has to be done. A special option name "default" is available to use default options for a filter.

It is required to specify either "default" or at least one of the filter options. All omitted options are used with their default values. The syntax of the filter string is as follows:

```
filter=<NAME>@<OPTION>=<VALUE>[@<OPTION>=<VALUE>][@...][#<NAME>@<OPTION>=<V
```

output_rect

Specify a rectangle where the output of the filter chain is copied into the input image. It is given by a list of space separated float values:

```
output_rect=x\ y\ width\ height
```

If not given, the output rectangle equals the dimensions of the input image. The output rectangle is automatically cropped at the borders of the input image. Negative values are valid for each component.

```
output_rect=25\ 25\ 100\ 100
```

Several filters can be chained for successive processing without GPU-HOST transfers allowing for fast processing of complex filter chains. Currently, only filters with zero (generators) or exactly one (filters) input image and one output image are supported. Also, transition filters are not yet usable as intended.

Some filters generate output images with additional padding depending on the respective filter kernel. The padding is automatically removed to ensure the filter output has the same size as the input image.

For image generators, the size of the output image is determined by the previous output image of the filter chain or the input image of the whole filterchain, respectively. The generators do not use the pixel information of this image to generate their output. However, the generated output is blended onto this image, resulting in partial or complete coverage of the output image.

The **coreimagesrc** video source can be used for generating input images which are directly fed into the filter chain. By using it, providing input images by another video source or an input video is not required.

Examples

- ⊕ List all filters available:

```
coreimage=list_filters=true
```

- ⊕ Use the CIBoxBlur filter with default options to blur an image:

```
coreimage=filter=CIBoxBlur@default
```

- ⊕ Use a filter chain with CISepiaTone at default values and CIVignetteEffect with its center at 100x100 and a radius of 50 pixels:

```
coreimage=filter=CIBoxBlur@default#CIVignetteEffect@inputCenter=100\ 100@inputRadius=50
```

- ⊕ Use nullsrc and CIQRCodeGenerator to create a QR code for the FFmpeg homepage, given as complete and escaped command-line for Apple's standard bash shell:

```
ffmpeg -f lavfi -i nullsrc=s=100x100,coreimage=filter=CIQRCodeGenerator@inputMessage=https\\\\\\\\://FFm
```

corr

Obtain the correlation between two input videos.

This filter takes two input videos.

Both input videos must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained per component, average, min and max correlation is printed through the logging system.

The filter stores the calculated correlation of each frame in frame metadata.

This filter also supports the **framesync** options.

In the below example the input file *main.mpg* being processed is compared with the reference file *ref.mpg*.

```
ffmpeg -i main.mpg -i ref.mpg -lavfi corr -f null -
```

cover_rect

Cover a rectangular object

It accepts the following options:

cover

Filepath of the optional cover image, needs to be in yuv420.

mode

Set covering mode.

It accepts the following values:

cover

cover it by the supplied image

blur cover it by interpolating the surrounding pixels

Default value is *blur*.

Examples

- ⊕ Cover a rectangular object by the supplied image of a given video using **ffmpeg**:

```
ffmpeg -i file.ts -vf find_rect=newref.pgm,cover_rect=cover.jpg:mode=cover new.mkv
```

crop

Crop the input video to given dimensions.

It accepts the following parameters:

w, out_w

The width of the output video. It defaults to "iw". This expression is evaluated only once during the filter configuration, or when the **w** or **out_w** command is sent.

h, out_h

The height of the output video. It defaults to "ih". This expression is evaluated only once during the filter configuration, or when the **h** or **out_h** command is sent.

x The horizontal position, in the input video, of the left edge of the output video. It defaults to "(in_w-out_w)/2". This expression is evaluated per-frame.

y The vertical position, in the input video, of the top edge of the output video. It defaults to "(in_h-out_h)/2". This expression is evaluated per-frame.

keep_aspect

If set to 1 will force the output display aspect ratio to be the same of the input, by changing the output sample aspect ratio. It defaults to 0.

exact

Enable exact cropping. If enabled, subsampled videos will be cropped at exact width/height/x/y as specified and will not be rounded to nearest smaller value. It defaults to 0.

The *out_w*, *out_h*, *x*, *y* parameters are expressions containing the following constants:

x

y The computed values for *x* and *y*. They are evaluated for each new frame.

in_w**in_h**

The input width and height.

iw

ih These are the same as *in_w* and *in_h*.

out_w**out_h**

The output (cropped) width and height.

ow

oh These are the same as *out_w* and *out_h*.

a same as iw / ih

sar input sample aspect ratio

dar input display aspect ratio, it is the same as $(iw / ih) * sar$

hsub**vsub**

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

n The number of the input frame, starting from 0.

pos the position in the file of the input frame, NAN if unknown

t The timestamp expressed in seconds. It's NAN if the input timestamp is unknown.

The expression for *out_w* may depend on the value of *out_h*, and the expression for *out_h* may depend on *out_w*, but they cannot depend on *x* and *y*, as *x* and *y* are evaluated after *out_w* and *out_h*.

The *x* and *y* parameters specify the expressions for the position of the top-left corner of the output (non-cropped) area. They are evaluated for each frame. If the evaluated value is not valid, it is approximated to the nearest valid value.

The expression for *x* may depend on *y*, and the expression for *y* may depend on *x*.

Examples

- ⊕ Crop area with size 100x100 at position (12,34).

```
crop=100:100:12:34
```

Using named options, the example above becomes:

```
crop=w=100:h=100:x=12:y=34
```

- ⊕ Crop the central input area with size 100x100:

```
crop=100:100
```

- ⊕ Crop the central input area with size 2/3 of the input video:

```
crop=2/3*in_w:2/3*in_h
```

- ⊕ Crop the input video central square:

```
crop=out_w=in_h
crop=in_h
```

- ⊕ Delimit the rectangle with the top-left corner placed at position 100:100 and the right-bottom corner corresponding to the right-bottom corner of the input image.

```
crop=in_w-100:in_h-100:100:100
```

- ⊕ Crop 10 pixels from the left and right borders, and 20 pixels from the top and bottom borders

```
crop=in_w-2*10:in_h-2*20
```

- ⊕ Keep only the bottom right quarter of the input image:

```
crop=in_w/2:in_h/2:in_w/2:in_h/2
```

- ⊕ Crop height for getting Greek harmony:

```
crop=in_w:1/PHI*in_w
```

- ⊕ Apply trembling effect:

```
crop=in_w/2:in_h/2:(in_w-out_w)/2+((in_w-out_w)/2)*sin(n/10):(in_h-out_h)/2+((in_h-out_h)/2)*sin(n/7)
```

- ⊕ Apply erratic camera effect depending on timestamp:

```
crop=in_w/2:in_h/2:(in_w-out_w)/2+((in_w-out_w)/2)*sin(t*10):(in_h-out_h)/2+((in_h-out_h)/2)*sin(t*13)
```

- ⊕ Set x depending on the value of y:

```
crop=in_w/2:in_h/2:y:10+10*sin(n/10)
```

Commands

This filter supports the following commands:

w, out_w

h, out_h

x

y Set width/height of the output video and the horizontal/vertical position in the input video. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

cropdetect

Auto-detect the crop size.

It calculates the necessary cropping parameters and prints the recommended parameters via the logging

system. The detected dimensions correspond to the non-black or video area of the input video according to *mode*.

It accepts the following parameters:

mode

Depending on *mode* crop detection is based on either the mere black value of surrounding pixels or a combination of motion vectors and edge pixels.

black

Detect black pixels surrounding the playing video. For fine control use option *limit*.

mvedges

Detect the playing video by the motion vectors inside the video and scanning for edge pixels typically forming the border of a playing video.

limit

Set higher black value threshold, which can be optionally specified from nothing (0) to everything (255 for 8-bit based formats). An intensity value greater to the set value is considered non-black. It defaults to 24. You can also specify a value between 0.0 and 1.0 which will be scaled depending on the bitdepth of the pixel format.

round

The value which the width/height should be divisible by. It defaults to 16. The offset is automatically adjusted to center the video. Use 2 to get only even dimensions (needed for 4:2:2 video). 16 is best when encoding to most video codecs.

skip Set the number of initial frames for which evaluation is skipped. Default is 2. Range is 0 to INT_MAX.

reset_count, reset

Set the counter that determines after how many frames cropdetect will reset the previously detected largest video area and start over to detect the current optimal crop area. Default value is 0.

This can be useful when channel logos distort the video area. 0 indicates 'never reset', and returns the largest area encountered during playback.

mv_threshold

Set motion in pixel units as threshold for motion detection. It defaults to 8.

low**high**

Set low and high threshold values used by the Canny thresholding algorithm.

The high threshold selects the "strong" edge pixels, which are then connected through 8-connectivity with the "weak" edge pixels selected by the low threshold.

low and *high* threshold values must be chosen in the range [0,1], and *low* should be lesser or equal to *high*.

Default value for *low* is "5/255", and default value for *high* is "15/255".

Examples

- ⊕ Find video area surrounded by black borders:

```
ffmpeg -i file.mp4 -vf cropdetect,metadata=mode=print -f null -
```

- ⊕ Find an embedded video area, generate motion vectors beforehand:

```
ffmpeg -i file.mp4 -vf mestimate,cropdetect=mode=medges,metadata=mode=print -f null -
```

- ⊕ Find an embedded video area, use motion vectors from decoder:

```
ffmpeg -flags2 +export_mvs -i file.mp4 -vf cropdetect=mode=medges,metadata=mode=print -f null -
```

Commands

This filter supports the following commands:

limit

The command accepts the same syntax of the corresponding option. If the specified expression is not valid, it is kept at its current value.

cue

Delay video filtering until a given wallclock timestamp. The filter first passes on **preroll** amount of frames, then it buffers at most **buffer** amount of frames and waits for the cue. After reaching the cue it forwards the buffered frames and also any subsequent frames coming in its input.

The filter can be used synchronize the output of multiple ffmpeg processes for realtime output devices

like decklink. By putting the delay in the filtering chain and pre-buffering frames the process can pass on data to output almost immediately after the target wallclock timestamp is reached.

Perfect frame accuracy cannot be guaranteed, but the result is good enough for some use cases.

cue The cue timestamp expressed in a UNIX timestamp in microseconds. Default is 0.

preroll

The duration of content to pass on as preroll expressed in seconds. Default is 0.

buffer

The maximum duration of content to buffer before waiting for the cue expressed in seconds. Default is 0.

curves

Apply color adjustments using curves.

This filter is similar to the Adobe Photoshop and GIMP curves tools. Each component (red, green and blue) has its values defined by N key points tied from each other using a smooth curve. The x-axis represents the pixel values from the input frame, and the y-axis the new pixel values to be set for the output frame.

By default, a component curve is defined by the two points $(0;0)$ and $(1;1)$. This creates a straight line where each original pixel value is "adjusted" to its own value, which means no change to the image.

The filter allows you to redefine these two points and add some more. A new curve will be defined to pass smoothly through all these new coordinates. The new defined points need to be strictly increasing over the x-axis, and their x and y values must be in the $[0;1]$ interval. The curve is formed by using a natural or monotonic cubic spline interpolation, depending on the *interp* option (default: "natural"). The "natural" spline produces a smoother curve in general while the monotonic ("pchip") spline guarantees the transitions between the specified points to be monotonic. If the computed curves happened to go outside the vector spaces, the values will be clipped accordingly.

The filter accepts the following options:

preset

Select one of the available color presets. This option can be used in addition to the **r**, **g**, **b** parameters; in this case, the later options take priority on the preset values. Available presets are:

none

color_negative
cross_process
darker
increase_contrast
lighter
linear_contrast
medium_contrast
negative
strong_contrast
vintage

Default is "none".

master, m

Set the master key points. These points will define a second pass mapping. It is sometimes called a "luminance" or "value" mapping. It can be used with **r**, **g**, **b** or **all** since it acts like a post-processing LUT.

red, r

Set the key points for the red component.

green, g

Set the key points for the green component.

blue, b

Set the key points for the blue component.

all Set the key points for all components (not including master). Can be used in addition to the other key points component options. In this case, the unset component(s) will fallback on this **all** setting.

psfile

Specify a Photoshop curves file (".acv") to import the settings from.

plot Save Gnuplot script of the curves in specified file.

interp

Specify the kind of interpolation. Available algorithms are:

natural

Natural cubic spline using a piece-wise cubic polynomial that is twice continuously

differentiable.

pchip

Monotonic cubic spline using a piecewise cubic Hermite interpolating polynomial (PCHIP).

To avoid some filtergraph syntax conflicts, each key points list need to be defined using the following syntax: "x0/y0 x1/y1 x2/y2 ...".

Commands

This filter supports same **commands** as options.

Examples

- ⊕ Increase slightly the middle level of blue:

```
curves=blue='0/0 0.5/0.58 1/1'
```

- ⊕ Vintage effect:

```
curves=r='0/0.11 .42/.51 1/0.95':g='0/0 0.50/0.48 1/1':b='0/0.22 .49/.44 1/0.8'
```

Here we obtain the following coordinates for each components:

```
red "(0;0.11) (0.42;0.51) (1;0.95)"
```

green

```
"(0;0) (0.50;0.48) (1;1)"
```

```
blue "(0;0.22) (0.49;0.44) (1;0.80)"
```

- ⊕ The previous example can also be achieved with the associated built-in preset:

```
curves=preset=vintage
```

- ⊕ Or simply:

```
curves=vintage
```

- ⊕ Use a Photoshop preset and redefine the points of the green component:


```
curves=psfile='MyCurvesPresets/purple.acv':green='0/0 0.45/0.53 1/1'
```

- ⊕ Check out the curves of the "cross_process" profile using **ffmpeg** and **gnuplot**:

```
ffmpeg -f lavfi -i color -vf curves=cross_process:plot=/tmp/curves.plt -frames:v 1 -f null -  
gnuplot -p /tmp/curves.plt
```

datascope

Video data analysis filter.

This filter shows hexadecimal pixel values of part of video.

The filter accepts the following options:

size, s

Set output video size.

x Set x offset from where to pick pixels.

y Set y offset from where to pick pixels.

mode

Set scope mode, can be one of the following:

mono

Draw hexadecimal pixel values with white color on black background.

color

Draw hexadecimal pixel values with input video pixel color on black background.

color2

Draw hexadecimal pixel values on color background picked from input video, the text color is picked in such way so its always visible.

axis Draw rows and columns numbers on left and top of video.

opacity

Set background opacity.

format

Set display number format. Can be "hex", or "dec". Default is "hex".

components

Set pixel components to display. By default all pixel components are displayed.

Commands

This filter supports same **commands** as options excluding "size" option.

dblur

Apply Directional blur filter.

The filter accepts the following options:

angle

Set angle of directional blur. Default is 45.

radius

Set radius of directional blur. Default is 5.

planes

Set which planes to filter. By default all planes are filtered.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

dctdnoiz

Denoise frames using 2D DCT (frequency domain filtering).

This filter is not designed for real time.

The filter accepts the following options:

sigma, s

Set the noise sigma constant.

This *sigma* defines a hard threshold of " $3 * \text{sigma}$ "; every DCT coefficient (absolute value) below this threshold will be dropped.

If you need a more advanced filtering, see **expr**.

Default is 0.

overlap

Set number overlapping pixels for each block. Since the filter can be slow, you may want to reduce this value, at the cost of a less effective filter and the risk of various artefacts.

If the overlapping value doesn't permit processing the whole input width or height, a warning will be displayed and according borders won't be denoised.

Default value is *blocksize-1*, which is the best possible setting.

expr, e

Set the coefficient factor expression.

For each coefficient of a DCT block, this expression will be evaluated as a multiplier value for the coefficient.

If this option is set, the **sigma** option will be ignored.

The absolute value of the coefficient can be accessed through the *c* variable.

- n** Set the *blocksize* using the number of bits. " $1 \ll n$ " defines the *blocksize*, which is the width and height of the processed blocks.

The default value is 3 (8x8) and can be raised to 4 for a *blocksize* of 16x16. Note that changing this setting has huge consequences on the speed processing. Also, a larger block size does not necessarily mean a better de-noising.

Examples

Apply a denoise with a **sigma** of 4.5:

```
dctdnoiz=4.5
```

The same operation can be achieved using the expression system:

```
dctdnoiz=e='gte(c, 4.5*3)'
```

Violent denoise using a block size of "16x16":

```
dctdnoiz=15:n=4
```

deband

Remove banding artifacts from input video. It works by replacing banded pixels with average value of referenced pixels.

The filter accepts the following options:

1thr

2thr

3thr

4thr Set banding detection threshold for each plane. Default is 0.02. Valid range is 0.00003 to 0.5. If difference between current pixel and reference pixel is less than threshold, it will be considered as banded.

range, r

Banding detection range in pixels. Default is 16. If positive, random number in range 0 to set value will be used. If negative, exact absolute value will be used. The range defines square of four pixels around current pixel.

direction, d

Set direction in radians from which four pixel will be compared. If positive, random direction from 0 to set direction will be picked. If negative, exact of absolute value will be picked. For example direction 0, -PI or -2*PI radians will pick only pixels on same row and -PI/2 will pick only pixels on same column.

blur, b

If enabled, current pixel is compared with average value of all four surrounding pixels. The default is enabled. If disabled current pixel is compared with all four surrounding pixels. The pixel is considered banded if only all four differences with surrounding pixels are less than threshold.

coupling, c

If enabled, current pixel is changed if and only if all pixel components are banded, e.g. banding detection threshold is triggered for all color components. The default is disabled.

Commands

This filter supports the all above options as **commands**.

deblock

Remove blocking artifacts from input video.

The filter accepts the following options:

filter

Set filter type, can be *weak* or *strong*. Default is *strong*. This controls what kind of deblocking is applied.

block

Set size of block, allowed range is from 4 to 512. Default is 8.

alpha

beta

gamma

delta

Set blocking detection thresholds. Allowed range is 0 to 1. Defaults are: *0.098* for *alpha* and *0.05* for the rest. Using higher threshold gives more deblocking strength. Setting *alpha* controls threshold detection at exact edge of block. Remaining options controls threshold detection near the edge. Each one for below/above or left/right. Setting any of those to *0* disables deblocking.

planes

Set planes to filter. Default is to filter all available planes.

Examples

- ⊕ Deblock using weak filter and block size of 4 pixels.

```
deblock=filter=weak:block=4
```

- ⊕ Deblock using strong filter, block size of 4 pixels and custom thresholds for deblocking more edges.

```
deblock=filter=strong:block=4:alpha=0.12:beta=0.07:gamma=0.06:delta=0.05
```

- ⊕ Similar as above, but filter only first plane.

```
deblock=filter=strong:block=4:alpha=0.12:beta=0.07:gamma=0.06:delta=0.05:planes=1
```

- ⊕ Similar as above, but filter only second and third plane.

```
deblock=filter=strong:block=4:alpha=0.12:beta=0.07:gamma=0.06:delta=0.05:planes=6
```

Commands

This filter supports the all above options as **commands**.

decimate

Drop duplicated frames at regular intervals.

The filter accepts the following options:

cycle

Set the number of frames from which one will be dropped. Setting this to N means one frame in every batch of N frames will be dropped. Default is 5.

dupthresh

Set the threshold for duplicate detection. If the difference metric for a frame is less than or equal to this value, then it is declared as duplicate. Default is 1.1

scthresh

Set scene change threshold. Default is 15.

blockx

blocky

Set the size of the x and y-axis blocks used during metric calculations. Larger blocks give better noise suppression, but also give worse detection of small movements. Must be a power of two. Default is 32.

ppsrc

Mark main input as a pre-processed input and activate clean source input stream. This allows the input to be pre-processed with various filters to help the metrics calculation while keeping the frame selection lossless. When set to 1, the first stream is for the pre-processed input, and the second stream is the clean source from where the kept frames are chosen. Default is 0.

chroma

Set whether or not chroma is considered in the metric calculations. Default is 1.

mixed

Set whether or not the input only partially contains content to be decimated. Default is "false". If enabled video output stream will be in variable frame rate.

deconvolve

Apply 2D deconvolution of video stream in frequency domain using second stream as impulse.

The filter accepts the following options:

planes

Set which planes to process.

impulse

Set which impulse video frames will be processed, can be *first* or *all*. Default is *all*.

noise

Set noise when doing divisions. Default is *0.0000001*. Useful when width and height are not same and not power of 2 or if stream prior to convolving had noise.

The "deconvolve" filter also supports the **framesync** options.

dedot

Reduce cross-luminance (dot-crawl) and cross-color (rainbows) from video.

It accepts the following options:

m Set mode of operation. Can be combination of *dotcrawl* for cross-luminance reduction and/or *rainbows* for cross-color reduction.

lt Set spatial luma threshold. Lower values increases reduction of cross-luminance.

tl Set tolerance for temporal luma. Higher values increases reduction of cross-luminance.

tc Set tolerance for chroma temporal variation. Higher values increases reduction of cross-color.

ct Set temporal chroma threshold. Lower values increases reduction of cross-color.

deflate

Apply deflate effect to the video.

This filter replaces the pixel by the local(3x3) average by taking into account only values lower than

the pixel.

It accepts the following options:

threshold0

threshold1

threshold2

threshold3

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

Commands

This filter supports the all above options as **commands**.

deflicker

Remove temporal frame luminance variations.

It accepts the following options:

size, s

Set moving-average filter size in frames. Default is 5. Allowed range is 2 - 129.

mode, m

Set averaging mode to smooth temporal luminance variations.

Available values are:

am Arithmetic mean

gm Geometric mean

hm Harmonic mean

qm Quadratic mean

cm Cubic mean

pm Power mean

median

Median

bypass

Do not actually modify frame. Useful when one only wants metadata.

dejudder

Remove judder produced by partially interlaced telecined content.

Judder can be introduced, for instance, by **pullup** filter. If the original source was partially telecined content then the output of "pullup,dejudder" will have a variable frame rate. May change the recorded frame rate of the container. Aside from that change, this filter will not affect constant frame rate video.

The option available in this filter is:

cycle

Specify the length of the window over which the judder repeats.

Accepts any integer greater than 1. Useful values are:

- 4** If the original was telecined from 24 to 30 fps (Film to NTSC).
- 5** If the original was telecined from 25 to 30 fps (PAL to NTSC).
- 20** If a mixture of the two.

The default is **4**.

delogo

Suppress a TV station logo by a simple interpolation of the surrounding pixels. Just set a rectangle covering the logo and watch it disappear (and sometimes something even uglier appear - your mileage may vary).

It accepts the following parameters:

x

y Specify the top left corner coordinates of the logo. They must be specified.

w

h Specify the width and height of the logo to clear. They must be specified.

show

When set to 1, a green rectangle is drawn on the screen to simplify finding the right x , y , w , and h parameters. The default value is 0.

The rectangle is drawn on the outermost pixels which will be (partly) replaced with interpolated values. The values of the next pixels immediately outside this rectangle in each direction will be used to compute the interpolated pixel values inside the rectangle.

Examples

- ⊕ Set a rectangle covering the area with top left corner coordinates 0,0 and size 100x77:

```
delogo=x=0:y=0:w=100:h=77
```

derain

Remove the rain in the input image/video by applying the derain methods based on convolutional neural networks. Supported models:

- ⊕ Recurrent Squeeze-and-Excitation Context Aggregation Net (RESCAN). See http://openaccess.thecvf.com/content_ECCV_2018/papers/Xia_Li_Recurrent_Squeeze-and-Excitation_Co

Training as well as model generation scripts are provided in the repository at https://github.com/XuweiMeng/derain_filter.git.

Native model files (.model) can be generated from TensorFlow model files (.pb) by using `tools/python/convert.py`

The filter accepts the following options:

filter_type

Specify which filter to use. This option accepts the following values:

derain

Derain filter. To conduct derain filter, you need to use a derain model.

dehaze

Dehaze filter. To conduct dehaze filter, you need to use a dehaze model.

Default value is **derain**.

dnn_backend

Specify which DNN backend to use for model loading and execution. This option accepts the following values:

native

Native implementation of DNN loading and execution.

tensorflow

TensorFlow backend. To enable this backend you need to install the TensorFlow for C library (see <https://www.tensorflow.org/install/lang_c>) and configure FFmpeg with "--enable-libtensorflow"

Default value is **native**.

model

Set path to model file specifying network architecture and its parameters. Note that different backends use different file formats. TensorFlow and native backend can load files for only its format.

To get full functionality (such as async execution), please use the **dnn_processing** filter.

deshake

Attempt to fix small changes in horizontal and/or vertical shift. This filter helps remove camera shake from hand-holding a camera, bumping a tripod, moving on a vehicle, etc.

The filter accepts the following options:

x

y

w

h Specify a rectangular area where to limit the search for motion vectors. If desired the search for motion vectors can be limited to a rectangular area of the frame defined by its top left corner, width and height. These parameters have the same meaning as the drawbox filter which can be used to visualise the position of the bounding box.

This is useful when simultaneous movement of subjects within the frame might be confused for camera motion by the motion vector search.

If any or all of x , y , w and h are set to -1 then the full frame is used. This allows later options to be set without specifying the bounding box for the motion vector search.

Default - search the whole frame.

rx

ry Specify the maximum extent of movement in x and y directions in the range 0-64 pixels. Default 16.

edge

Specify how to generate pixels to fill blanks at the edge of the frame. Available values are:

blank, 0

Fill zeroes at blank locations

original, 1

Original image at blank locations

clamp, 2

Extruded edge value at blank locations

mirror, 3

Mirrored edge at blank locations

Default value is **mirror**.

blocksize

Specify the blocksize to use for motion search. Range 4-128 pixels, default 8.

contrast

Specify the contrast threshold for blocks. Only blocks with more than the specified contrast (difference between darkest and lightest pixels) will be considered. Range 1-255, default 125.

search

Specify the search strategy. Available values are:

exhaustive, 0

Set exhaustive search

less, 1

Set less exhaustive search.

Default value is **exhaustive**.

filename

If set then a detailed log of the motion search is written to the specified file.

despill

Remove unwanted contamination of foreground colors, caused by reflected color of greenscreen or bluescreen.

This filter accepts the following options:

type Set what type of despill to use.

mix Set how spillmap will be generated.

expand

Set how much to get rid of still remaining spill.

red Controls amount of red in spill area.

green

Controls amount of green in spill area. Should be -1 for greenscreen.

blue Controls amount of blue in spill area. Should be -1 for bluescreen.

brightness

Controls brightness of spill area, preserving colors.

alpha

Modify alpha from generated spillmap.

Commands

This filter supports the all above options as **commands**.

detelecine

Apply an exact inverse of the telecine operation. It requires a predefined pattern specified using the pattern option which must be the same as that passed to the telecine filter.

This filter accepts the following options:

first_field

top, t

top field first

bottom, b

bottom field first The default value is "top".

pattern

A string of numbers representing the pulldown pattern you wish to apply. The default value is 23.

start_frame

A number representing position of the first frame with respect to the telecine pattern. This is to be used if the stream is cut. The default value is 0.

dilation

Apply dilation effect to the video.

This filter replaces the pixel by the local(3x3) maximum.

It accepts the following options:

threshold0**threshold1****threshold2****threshold3**

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

coordinates

Flag which specifies the pixel to refer to. Default is 255 i.e. all eight pixels are used.

Flags to local 3x3 coordinates maps like this:

1 2 3

4 5

6 7 8

Commands

This filter supports the all above options as **commands**.

displace

Displace pixels as indicated by second and third input stream.

It takes three input streams and outputs one stream, the first input is the source, and second and third input are displacement maps.

The second input specifies how much to displace pixels along the x-axis, while the third input specifies how much to displace pixels along the y-axis. If one of displacement map streams terminates, last frame from that displacement map will be used.

Note that once generated, displacements maps can be reused over and over again.

A description of the accepted options follows.

edge

Set displace behavior for pixels that are out of range.

Available values are:

blank

Missing pixels are replaced by black pixels.

smear

Adjacent pixels will spread out to replace missing pixels.

wrap

Out of range pixels are wrapped so they point to pixels of other side.

mirror

Out of range pixels will be replaced with mirrored pixels.

Default is **smear**.

Examples

- ⊕ Add ripple effect to rgb input of video size hd720:

```
ffmpeg -i INPUT -f lavfi -i nullsrc=s=hd720,lutrgb=128:128:128 -f lavfi -i nullsrc=s=hd720,geq='r=128+30'
```

- ⊕ Add wave effect to rgb input of video size hd720:

```
ffmpeg -i INPUT -f lavfi -i nullsrc=hd720,geq='r=128+80*(sin(sqrt((X-W/2)*(X-W/2)+(Y-H/2)*(Y-H/2)))/2
```

dnn_classify

Do classification with deep neural networks based on bounding boxes.

The filter accepts the following options:

dnn_backend

Specify which DNN backend to use for model loading and execution. This option accepts only `openvino` now, `tensorflow` backends will be added.

model

Set path to model file specifying network architecture and its parameters. Note that different backends use different file formats.

input

Set the input name of the dnn network.

output

Set the output name of the dnn network.

confidence

Set the confidence threshold (default: 0.5).

labels

Set path to label file specifying the mapping between label id and name. Each label name is written in one line, trailing spaces and empty lines are skipped. The first line is the name of label id 0, and the second line is the name of label id 1, etc. The label id is considered as name if the label file is not provided.

backend_configs

Set the configs to be passed into backend

For `tensorflow` backend, you can set its configs with `sess_config` options, please use `tools/python/tf_sess_config.py` to get the configs for your system.

dnn_detect

Do object detection with deep neural networks.

The filter accepts the following options:

dnn_backend

Specify which DNN backend to use for model loading and execution. This option accepts only openvino now, tensorflow backends will be added.

model

Set path to model file specifying network architecture and its parameters. Note that different backends use different file formats.

input

Set the input name of the dnn network.

output

Set the output name of the dnn network.

confidence

Set the confidence threshold (default: 0.5).

labels

Set path to label file specifying the mapping between label id and name. Each label name is written in one line, trailing spaces and empty lines are skipped. The first line is the name of label id 0 (usually it is 'background'), and the second line is the name of label id 1, etc. The label id is considered as name if the label file is not provided.

backend_configs

Set the configs to be passed into backend. To use async execution, set async (default: set). Roll back to sync execution if the backend does not support async.

dnn_processing

Do image processing with deep neural networks. It works together with another filter which converts the pixel format of the Frame to what the dnn network requires.

The filter accepts the following options:

dnn_backend

Specify which DNN backend to use for model loading and execution. This option accepts the following values:

native

Native implementation of DNN loading and execution.

tensorflow

TensorFlow backend. To enable this backend you need to install the TensorFlow for C library (see <https://www.tensorflow.org/install/lang_c>) and configure FFmpeg with "--enable-libtensorflow"

openvino

OpenVINO backend. To enable this backend you need to build and install the OpenVINO for C library (see <<https://github.com/openvinotoolkit/openvino/blob/master/build-instruction.md>>) and configure FFmpeg with "--enable-libopenvino" (--extra-cflags=-I... --extra-ldflags=-L... might be needed if the header files and libraries are not installed into system path)

Default value is **native**.

model

Set path to model file specifying network architecture and its parameters. Note that different backends use different file formats. TensorFlow, OpenVINO and native backend can load files for only its format.

Native model file (.model) can be generated from TensorFlow model file (.pb) by using tools/python/convert.py

input

Set the input name of the dnn network.

output

Set the output name of the dnn network.

backend_configs

Set the configs to be passed into backend. To use async execution, set async (default: set). Roll back to sync execution if the backend does not support async.

For tensorflow backend, you can set its configs with **sess_config** options, please use tools/python/tf_sess_config.py to get the configs of TensorFlow backend for your system.

Examples

- ⊕ Remove rain in rgb24 frame with can.pb (see **derain** filter):

```
./ffmpeg -i rain.jpg -vf format=rgb24,dnn_processing=dnn_backend=tensorflow:model=can.pb:input=x:output=y
```

- ⊕ Halve the pixel value of the frame with format gray32f:

```
ffmpeg -i input.jpg -vf format=grayf32,dnn_processing=model=halve_gray_float.model:input=dnn_in:output=
```

- ⊕ Handle the Y channel with srcnn.pb (see **sr** filter) for frame with yuv420p (planar YUV formats supported):

```
./ffmpeg -i 480p.jpg -vf format=yuv420p,scale=w=iw*2:h=ih*2,dnn_processing=dnn_backend=tensorflow:m
```

- ⊕ Handle the Y channel with espcn.pb (see **sr** filter), which changes frame size, for format yuv420p (planar YUV formats supported), please use tools/python/tf_sess_config.py to get the configs of TensorFlow backend for your system.

```
./ffmpeg -i 480p.jpg -vf format=yuv420p,dnn_processing=dnn_backend=tensorflow:model=espcn.pb:input=
```

drawbox

Draw a colored box on the input image.

It accepts the following parameters:

x

y The expressions which specify the top left corner coordinates of the box. It defaults to 0.

width, w

height, h

The expressions which specify the width and height of the box; if 0 they are interpreted as the input width and height. It defaults to 0.

color, c

Specify the color of the box to write. For the general syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. If the special value "invert" is used, the box edge color is the same as the video with inverted luma.

thickness, t

The expression which sets the thickness of the box edge. A value of "fill" will create a filled box. Default value is 3.

See below for the list of accepted constants.

replace

Applicable if the input has alpha. With value 1, the pixels of the painted box will overwrite the video's color and alpha pixels. Default is 0, which composites the box onto the input, leaving the video's alpha intact.

The parameters for x , y , w and h and t are expressions containing the following constants:

dar The input display aspect ratio, it is the same as $(w / h) * sar$.

hsub

vsub

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" $hsub$ is 2 and $vsub$ is 1.

in_h, ih

in_w, iw

The input width and height.

sar The input sample aspect ratio.

x

y The x and y offset coordinates where the box is drawn.

w

h The width and height of the drawn box.

box_source

Box source can be set as `side_data_detection_bboxes` if you want to use box data in detection bboxes of side data.

If `box_source` is set, the x , y , $width$ and $height$ will be ignored and still use box data in detection bboxes of side data. So please do not use this parameter if you were not sure about the box source.

t The thickness of the drawn box.

These constants allow the x , y , w , h and t expressions to refer to each other, so you may for example specify " $y=x/dar$ " or " $h=w/dar$ ".

Examples

⊕ Draw a black box around the edge of the input image:

drawbox

- ⊕ Draw a box with color red and an opacity of 50%:

```
drawbox=10:20:200:60:red@0.5
```

The previous example can be specified as:

```
drawbox=x=10:y=20:w=200:h=60:color=red@0.5
```

- ⊕ Fill the box with pink color:

```
drawbox=x=10:y=10:w=100:h=100:color=pink@0.5:t=fill
```

- ⊕ Draw a 2-pixel red 2.40:1 mask:

```
drawbox=x=t:y=0.5*(ih-iw/2.4)-t:w=iw+t*2:h=iw/2.4+t*2:t=2:c=red
```

Commands

This filter supports same commands as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

drawgraph

Draw a graph using input video metadata.

It accepts the following parameters:

m1 Set 1st frame metadata key from which metadata values will be used to draw a graph.

fg1 Set 1st foreground color expression.

m2 Set 2nd frame metadata key from which metadata values will be used to draw a graph.

fg2 Set 2nd foreground color expression.

m3 Set 3rd frame metadata key from which metadata values will be used to draw a graph.

fg3 Set 3rd foreground color expression.

m4 Set 4th frame metadata key from which metadata values will be used to draw a graph.

fg4 Set 4th foreground color expression.

min Set minimal value of metadata value.

max Set maximal value of metadata value.

bg Set graph background color. Default is white.

mode

Set graph mode.

Available values for mode is:

bar

dot

line

Default is "line".

slide

Set slide mode.

Available values for slide is:

frame

Draw new frame when right border is reached.

replace

Replace old columns with new ones.

scroll

Scroll from right to left.

rscroll

Scroll from left to right.

picture

Draw single picture.

Default is "frame".

size Set size of graph video. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. The default value is "900x256".

rate, r

Set the output frame rate. Default value is 25.

The foreground color expressions can use the following variables:

MIN

Minimal value of metadata value.

MAX

Maximal value of metadata value.

VAL

Current metadata key value.

The color is defined as 0xAABBGGRR.

Example using metadata from **signalstats** filter:

```
signalstats,drawgraph=lavfi.signalstats.YAVG:min=0:max=255
```

Example using metadata from **ebur128** filter:

```
ebur128=metadata=1,adrawgraph=lavfi.r128.M:min=-120:max=5
```

drawgrid

Draw a grid on the input image.

It accepts the following parameters:

x

y The expressions which specify the coordinates of some point of grid intersection (meant to configure offset). Both default to 0.

width, w**height, h**

The expressions which specify the width and height of the grid cell, if 0 they are interpreted as the input width and height, respectively, minus "thickness", so image gets framed. Default to 0.

color, c

Specify the color of the grid. For the general syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. If the special value "invert" is used, the grid color is the same as the video with inverted luma.

thickness, t

The expression which sets the thickness of the grid line. Default value is 1.

See below for the list of accepted constants.

replace

Applicable if the input has alpha. With 1 the pixels of the painted grid will overwrite the video's color and alpha pixels. Default is 0, which composites the grid onto the input, leaving the video's alpha intact.

The parameters for *x*, *y*, *w* and *h* and *t* are expressions containing the following constants:

dar The input display aspect ratio, it is the same as $(w / h) * sar$.

hsub**vsub**

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

in_h, ih**in_w, iw**

The input grid cell width and height.

sar The input sample aspect ratio.

x

y The x and y coordinates of some point of grid intersection (meant to configure offset).

w

h The width and height of the drawn cell.

t The thickness of the drawn cell.

These constants allow the *x*, *y*, *w*, *h* and *t* expressions to refer to each other, so you may for example specify "*y=x/dar*" or "*h=w/dar*".

Examples

⊕ Draw a grid with cell 100x100 pixels, thickness 2 pixels, with color red and an opacity of 50%:

```
drawgrid=width=100:height=100:thickness=2:color=red@0.5
```

⊕ Draw a white 3x3 grid with an opacity of 50%:

```
drawgrid=w=iw/3:h=ih/3:t=2:c=white@0.5
```

Commands

This filter supports same commands as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

drawtext

Draw a text string or text from a specified file on top of a video, using the libfreetype library.

To enable compilation of this filter, you need to configure FFmpeg with "--enable-libfreetype". To enable default font fallback and the *font* option you need to configure FFmpeg with "--enable-libfontconfig". To enable the *text_shaping* option, you need to configure FFmpeg with "--enable-libfribidi".

Syntax

It accepts the following parameters:

box Used to draw a box around text using the background color. The value must be either 1 (enable) or 0 (disable). The default value of *box* is 0.

boxborderw

Set the width of the border to be drawn around the box using *boxcolor*. The default value of *boxborderw* is 0.

boxcolor

The color to be used for drawing box around text. For the syntax of this option, check the **"Color" section in the ffmpeg-utils manual**.

The default value of *boxcolor* is "white".

line_spacing

Set the line spacing in pixels of the border to be drawn around the box using *box*. The default value of *line_spacing* is 0.

borderw

Set the width of the border to be drawn around the text using *bordercolor*. The default value of *borderw* is 0.

bordercolor

Set the color to be used for drawing border around text. For the syntax of this option, check the **"Color" section in the ffmpeg-utils manual**.

The default value of *bordercolor* is "black".

expansion

Select how the *text* is expanded. Can be either "none", "strftime" (deprecated) or "normal" (default). See the **drawtext_expansion, Text expansion** section below for details.

basetime

Set a start time for the count. Value is in microseconds. Only applied in the deprecated strftime expansion mode. To emulate in normal expansion mode use the "pts" function, supplying the start time (in seconds) as the second argument.

fix_bounds

If true, check and fix text coords to avoid clipping.

fontcolor

The color to be used for drawing fonts. For the syntax of this option, check the **"Color" section in the ffmpeg-utils manual**.

The default value of *fontcolor* is "black".

fontcolor_expr

String which is expanded the same way as *text* to obtain dynamic *fontcolor* value. By default this

option has empty value and is not processed. When this option is set, it overrides *fontcolor* option.

font The font family to be used for drawing text. By default Sans.

fontfile

The font file to be used for drawing text. The path must be included. This parameter is mandatory if the fontconfig support is disabled.

alpha

Draw the text applying alpha blending. The value can be a number between 0.0 and 1.0. The expression accepts the same variables *x*, *y* as well. The default value is 1. Please see *fontcolor_expr*.

fontsize

The font size to be used for drawing text. The default value of *fontsize* is 16.

text_shaping

If set to 1, attempt to shape the text (for example, reverse the order of right-to-left text and join Arabic characters) before drawing it. Otherwise, just draw the text exactly as given. By default 1 (if supported).

ft_load_flags

The flags to be used for loading the fonts.

The flags map the corresponding flags supported by libfreetype, and are a combination of the following values:

default
no_scale
no_hinting
render
no_bitmap
vertical_layout
force_autohint
crop_bitmap
pedantic
ignore_global_advance_width
no_recurse
ignore_transform
monochrome

linear_design
no_outhint

Default value is "default".

For more information consult the documentation for the FT_LOAD_* libfreetype flags.

shadowcolor

The color to be used for drawing a shadow behind the drawn text. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**.

The default value of *shadowcolor* is "black".

shadowx

shadowy

The x and y offsets for the text shadow position with respect to the position of the text. They can be either positive or negative values. The default value for both is "0".

start_number

The starting frame number for the n/frame_num variable. The default value is "0".

tabsize

The size in number of spaces to use for rendering the tab. Default value is 4.

timecode

Set the initial timecode representation in "hh:mm:ss[;:].ff" format. It can be used with or without text parameter. *timecode_rate* option must be specified.

timecode_rate, rate, r

Set the timecode frame rate (timecode only). Value will be rounded to nearest integer. Minimum value is "1". Drop-frame timecode is supported for frame rates 30 & 60.

tc24hmax

If set to 1, the output of the timecode option will wrap around at 24 hours. Default is 0 (disabled).

text The text string to be drawn. The text must be a sequence of UTF-8 encoded characters. This parameter is mandatory if no file is specified with the parameter *textfile*.

textfile

A text file containing text to be drawn. The text must be a sequence of UTF-8 encoded characters.

This parameter is mandatory if no text string is specified with the parameter *text*.

If both *text* and *textfile* are specified, an error is thrown.

text_source

Text source should be set as *side_data_detection_bboxes* if you want to use text data in detection bboxes of side data.

If text source is set, *text* and *textfile* will be ignored and still use text data in detection bboxes of side data. So please do not use this parameter if you are not sure about the text source.

reload

The *textfile* will be reloaded at specified frame interval. Be sure to update *textfile* atomically, or it may be read partially, or even fail. Range is 0 to INT_MAX. Default is 0.

x

y The expressions which specify the offsets where text will be drawn within the video frame. They are relative to the top/left border of the output image.

The default value of *x* and *y* is "0".

See below for the list of accepted constants and functions.

The parameters for *x* and *y* are expressions containing the following constants and functions:

dar input display aspect ratio, it is the same as $(w / h) * sar$

hsub

vsub

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

line_h, lh

the height of each text line

main_h, h, H

the input height

main_w, w, W

the input width

max_glyph_a, ascent

the maximum distance from the baseline to the highest/upper grid coordinate used to place a glyph outline point, for all the rendered glyphs. It is a positive value, due to the grid's orientation with the Y axis upwards.

max_glyph_d, descent

the maximum distance from the baseline to the lowest grid coordinate used to place a glyph outline point, for all the rendered glyphs. This is a negative value, due to the grid's orientation, with the Y axis upwards.

max_glyph_h

maximum glyph height, that is the maximum height for all the glyphs contained in the rendered text, it is equivalent to *ascent - descent*.

max_glyph_w

maximum glyph width, that is the maximum width for all the glyphs contained in the rendered text

n the number of input frame, starting from 0

rand(min, max)

return a random number included between *min* and *max*

sar The input sample aspect ratio.

t timestamp expressed in seconds, NAN if the input timestamp is unknown

text_h, th

the height of the rendered text

text_w, tw

the width of the rendered text

x

y the x and y offset coordinates where the text is drawn.

These parameters allow the *x* and *y* expressions to refer to each other, so you can for example specify "y=x/dar".

pict_type

A one character description of the current frame's picture type.

pkt_pos

The current packet's position in the input file or stream (in bytes, from the start of the input). A value of -1 indicates this info is not available.

duration

The current packet's duration, in seconds.

pkt_size

The current packet's size (in bytes).

Text expansion

If **expansion** is set to "strftime", the filter recognizes **strftime()** sequences in the provided text and expands them accordingly. Check the documentation of **strftime()**. This feature is deprecated.

If **expansion** is set to "none", the text is printed verbatim.

If **expansion** is set to "normal" (which is the default), the following expansion mechanism is used.

The backslash character \, followed by any character, always expands to the second character.

Sequences of the form "% {...}" are expanded. The text between the braces is a function name, possibly followed by arguments separated by ':'. If the arguments contain special characters or delimiters (':' or '}''), they should be escaped.

Note that they probably must also be escaped as the value for the **text** option in the filter argument string and as the filter argument in the filtergraph description, and possibly also for the shell, that makes up to four levels of escaping; using a text file avoids these problems.

The following functions are available:

expr, e

The expression evaluation result.

It must take one argument specifying the expression to be evaluated, which accepts the same constants and functions as the *x* and *y* values. Note that not all constants should be used, for example the text size is not known when evaluating the expression, so the constants *text_w* and *text_h* will have an undefined value.

expr_int_format, eif

Evaluate the expression's value and output as formatted integer.

The first argument is the expression to be evaluated, just as for the *expr* function. The second argument specifies the output format. Allowed values are **x**, **X**, **d** and **u**. They are treated exactly as in the "printf" function. The third parameter is optional and sets the number of positions taken by the output. It can be used to add padding with zeros from the left.

gmtime

The time at which the filter is running, expressed in UTC. It can accept an argument: a **strftime()** format string. The format string is extended to support the variable *%[1-6]N* which prints fractions of the second with optionally specified number of digits.

localtime

The time at which the filter is running, expressed in the local time zone. It can accept an argument: a **strftime()** format string. The format string is extended to support the variable *%[1-6]N* which prints fractions of the second with optionally specified number of digits.

metadata

Frame metadata. Takes one or two arguments.

The first argument is mandatory and specifies the metadata key.

The second argument is optional and specifies a default value, used when the metadata key is not found or empty.

Available metadata can be identified by inspecting entries starting with TAG included within each frame section printed by running "ffprobe -show_frames".

String metadata generated in filters leading to the drawtext filter are also available.

n, frame_num

The frame number, starting from 0.

pict_type

A one character description of the current picture type.

pts The timestamp of the current frame. It can take up to three arguments.

The first argument is the format of the timestamp; it defaults to "flt" for seconds as a decimal number with microsecond accuracy; "hms" stands for a formatted *[-]HH:MM:SS.mmm* timestamp

with millisecond accuracy. "gmtime" stands for the timestamp of the frame formatted as UTC time; "localtime" stands for the timestamp of the frame formatted as local time zone time.

The second argument is an offset added to the timestamp.

If the format is set to "hms", a third argument "24HH" may be supplied to present the hour part of the formatted timestamp in 24h format (00-23).

If the format is set to "localtime" or "gmtime", a third argument may be supplied: a **strftime()** format string. By default, *YYYY-MM-DD HH:MM:SS* format will be used.

Commands

This filter supports altering parameters via commands:

reinit

Alter existing filter parameters.

Syntax for the argument is the same as for filter invocation, e.g.

```
fontsize=56:fontcolor=green:text='Hello World'
```

Full filter invocation with sendcmd would look like this:

```
sendcmd=c='56.0 drawtext reinit fontsize=56\;fontcolor=green\;text=Hello\ World'
```

If the entire argument can't be parsed or applied as valid values then the filter will continue with its existing parameters.

Examples

- ⊕ Draw "Test Text" with font FreeSerif, using the default values for the optional parameters.

```
drawtext="fontfile=/usr/share/fonts/truetype/freefont/FreeSerif.ttf: text='Test Text'"
```

- ⊕ Draw 'Test Text' with font FreeSerif of size 24 at position x=100 and y=50 (counting from the top-left corner of the screen), text is yellow with a red box around it. Both the text and the box have an opacity of 20%.

```
drawtext="fontfile=/usr/share/fonts/truetype/freefont/FreeSerif.ttf: text='Test Text':\
```

```
x=100: y=50: fontsize=24: fontcolor=yellow@0.2: box=1: boxcolor=red@0.2"
```

Note that the double quotes are not necessary if spaces are not used within the parameter list.

- ⊕ Show the text at the center of the video frame:

```
drawtext="fontsize=30:fontfile=FreeSerif.ttf:text='hello world':x=(w-text_w)/2:y=(h-text_h)/2"
```

- ⊕ Show the text at a random position, switching to a new position every 30 seconds:

```
drawtext="fontsize=30:fontfile=FreeSerif.ttf:text='hello world':x=if(eq(mod(t,30)\,0)\,rand(0\,(w-text_w))\,)
```

- ⊕ Show a text line sliding from right to left in the last row of the video frame. The file *LONG_LINE* is assumed to contain a single line with no newlines.

```
drawtext="fontsize=15:fontfile=FreeSerif.ttf:text=LONG_LINE:y=h-line_h:x=-50*t"
```

- ⊕ Show the content of file *CREDITS* off the bottom of the frame and scroll up.

```
drawtext="fontsize=20:fontfile=FreeSerif.ttf:textfile=CREDITS:y=h-20*t"
```

- ⊕ Draw a single green letter "g", at the center of the input video. The glyph baseline is placed at half screen height.

```
drawtext="fontsize=60:fontfile=FreeSerif.ttf:fontcolor=green:text=g:x=(w-max_glyph_w)/2:y=h/2-ascent"
```

- ⊕ Show text for 1 second every 3 seconds:

```
drawtext="fontfile=FreeSerif.ttf:fontcolor=white:x=100:y=x/dar:enable=lt(mod(t,3)\,1):text='blink'"
```

- ⊕ Use `fontconfig` to set the font. Note that the colons need to be escaped.

```
drawtext='fontfile=Linux Libertine O-40\:style=Semibold:text=FFmpeg'
```

- ⊕ Draw "Test Text" with font size dependent on height of the video.

```
drawtext="text='Test Text': fontsize=h/30: x=(w-text_w)/2: y=(h-text_h*2)"
```

- ⊕ Print the date of a real-time encoding (see `strftime(3)`):

```
drawtext='fontfile=FreeSans.ttf:text=%{localtime\:%a %b %d %Y}'
```

- ⊕ Show text fading in and out (appearing/disappearing):

```
#!/bin/sh
DS=1.0 # display start
DE=10.0 # display end
FID=1.5 # fade in duration
FOD=5 # fade out duration
ffplay -f lavfi "color,drawtext=text=TEST:fontsize=50:fontfile=FreeSerif.ttf:fontcolor_expr=ff0000%{eif\\|\\|};"
```

- ⊕ Horizontally align multiple separate texts. Note that **max_glyph_a** and the **fontsize** value are included in the **y** offset.

```
drawtext=fontfile=FreeSans.ttf:text=DOG:fontsize=24:x=10:y=20+24-max_glyph_a,
drawtext=fontfile=FreeSans.ttf:text=cow:fontsize=24:x=80:y=20+24-max_glyph_a
```

- ⊕ Plot special *lavfi.image2dec.source_basename* metadata onto each frame if such metadata exists. Otherwise, plot the string "NA". Note that image2 demuxer must have option **-export_path_metadata 1** for the special metadata fields to be available for filters.

```
drawtext="fontsize=20:fontcolor=white:fontfile=FreeSans.ttf:text='%{metadata\:\lavfi.image2dec.source_base"
```

For more information about libfreetype, check: <<http://www.freetype.org/>>.

For more information about fontconfig, check:

<<http://freedesktop.org/software/fontconfig/fontconfig-user.html>>.

For more information about libfribidi, check: <<http://fribidi.org/>>.

edgedetect

Detect and draw edges. The filter uses the Canny Edge Detection algorithm.

The filter accepts the following options:

low

high

Set low and high threshold values used by the Canny thresholding algorithm.

The high threshold selects the "strong" edge pixels, which are then connected through

8-connectivity with the "weak" edge pixels selected by the low threshold.

low and *high* threshold values must be chosen in the range [0,1], and *low* should be lesser or equal to *high*.

Default value for *low* is "20/255", and default value for *high* is "50/255".

mode

Define the drawing mode.

wires

Draw white/gray wires on black background.

colormix

Mix the colors to create a paint/cartoon effect.

canny

Apply Canny edge detector on all selected planes.

Default value is *wires*.

planes

Select planes for filtering. By default all available planes are filtered.

Examples

- ⊕ Standard edge detection with custom values for the hysteresis thresholding:

```
edgedetect=low=0.1:high=0.4
```

- ⊕ Painting effect without thresholding:

```
edgedetect=mode=colormix:high=0
```

elbg

Apply a posterize effect using the ELBG (Enhanced LBG) algorithm.

For each input image, the filter will compute the optimal mapping from the input to the output given the codebook length, that is the number of distinct output colors.

This filter accepts the following options.

codebook_length, l

Set codebook length. The value must be a positive integer, and represents the number of distinct output colors. Default value is 256.

nb_steps, n

Set the maximum number of iterations to apply for computing the optimal mapping. The higher the value the better the result and the higher the computation time. Default value is 1.

seed, s

Set a random seed, must be an integer included between 0 and UIN32_MAX. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

pal8 Set pal8 output pixel format. This option does not work with codebook length greater than 256. Default is disabled.

use_alpha

Include alpha values in the quantization calculation. Allows creating palettized output images (e.g. PNG8) with multiple alpha smooth blending.

entropy

Measure graylevel entropy in histogram of color channels of video frames.

It accepts the following parameters:

mode

Can be either *normal* or *diff*. Default is *normal*.

diff mode measures entropy of histogram delta values, absolute differences between neighbour histogram values.

epx

Apply the EPX magnification filter which is designed for pixel art.

It accepts the following option:

n Set the scaling dimension: 2 for "2xEPX", 3 for "3xEPX". Default is 3.

eq

Set brightness, contrast, saturation and approximate gamma adjustment.

The filter accepts the following options:

contrast

Set the contrast expression. The value must be a float value in range "-1000.0" to 1000.0. The default value is "1".

brightness

Set the brightness expression. The value must be a float value in range "-1.0" to 1.0. The default value is "0".

saturation

Set the saturation expression. The value must be a float in range 0.0 to 3.0. The default value is "1".

gamma

Set the gamma expression. The value must be a float in range 0.1 to 10.0. The default value is "1".

gamma_r

Set the gamma expression for red. The value must be a float in range 0.1 to 10.0. The default value is "1".

gamma_g

Set the gamma expression for green. The value must be a float in range 0.1 to 10.0. The default value is "1".

gamma_b

Set the gamma expression for blue. The value must be a float in range 0.1 to 10.0. The default value is "1".

gamma_weight

Set the gamma weight expression. It can be used to reduce the effect of a high gamma value on bright image areas, e.g. keep them from getting overamplified and just plain white. The value must be a float in range 0.0 to 1.0. A value of 0.0 turns the gamma correction all the way down while 1.0 leaves it at its full strength. Default is "1".

eval Set when the expressions for brightness, contrast, saturation and gamma expressions are evaluated.

It accepts the following values:

init only evaluate expressions once during the filter initialization or when a command is processed

frame
evaluate expressions for each incoming frame

Default value is **init**.

The expressions accept the following parameters:

n frame count of the input frame starting from 0

pos byte position of the corresponding packet in the input file, NAN if unspecified

r frame rate of the input video, NAN if the input frame rate is unknown

t timestamp expressed in seconds, NAN if the input timestamp is unknown

Commands

The filter supports the following commands:

contrast
Set the contrast expression.

brightness
Set the brightness expression.

saturation
Set the saturation expression.

gamma
Set the gamma expression.

gamma_r
Set the gamma_r expression.

gamma_g

Set gamma_g expression.

gamma_b

Set gamma_b expression.

gamma_weight

Set gamma_weight expression.

The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

erosion

Apply erosion effect to the video.

This filter replaces the pixel by the local(3x3) minimum.

It accepts the following options:

threshold0**threshold1****threshold2****threshold3**

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

coordinates

Flag which specifies the pixel to refer to. Default is 255 i.e. all eight pixels are used.

Flags to local 3x3 coordinates maps like this:

```
1 2 3
4 5
6 7 8
```

Commands

This filter supports the all above options as **commands**.

estdif

Deinterlace the input video ("estdif" stands for "Edge Slope Tracing Deinterlacing Filter").

Spatial only filter that uses edge slope tracing algorithm to interpolate missing lines. It accepts the following parameters:

mode

The interlacing mode to adopt. It accepts one of the following values:

frame

Output one frame for each frame.

field

Output one frame for each field.

The default value is "field".

parity

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

tff Assume the top field is first.

bff Assume the bottom field is first.

auto Enable automatic detection of field parity.

The default value is "auto". If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

deint

Specify which frames to deinterlace. Accepts one of the following values:

all Deinterlace all frames.

interlaced

Only deinterlace frames marked as interlaced.

The default value is "all".

rslope

Specify the search radius for edge slope tracing. Default value is 1. Allowed range is from 1 to 15.

redge

Specify the search radius for best edge matching. Default value is 2. Allowed range is from 0 to 15.

ecost

Specify the edge cost for edge matching. Default value is 1.0. Allowed range is from 0 to 9.

mcost

Specify the middle cost for edge matching. Default value is 0.5. Allowed range is from 0 to 1.

dcost

Specify the distance cost for edge matching. Default value is 0.5. Allowed range is from 0 to 1.

interp

Specify the interpolation used. Default is 4-point interpolation. It accepts one of the following values:

2p Two-point interpolation.

4p Four-point interpolation.

6p Six-point interpolation.

Commands

This filter supports same **commands** as options.

exposure

Adjust exposure of the video stream.

The filter accepts the following options:

exposure

Set the exposure correction in EV. Allowed range is from -3.0 to 3.0 EV Default value is 0 EV.

black

Set the black level correction. Allowed range is from -1.0 to 1.0. Default value is 0.

Commands

This filter supports same **commands** as options.

extractplanes

Extract color channel components from input video stream into separate grayscale video streams.

The filter accepts the following option:

planes

Set plane(s) to extract.

Available values for planes are:

y
u
v
a
r
g
b

Choosing planes not available in the input will result in an error. That means you cannot select "r", "g", "b" planes with "y", "u", "v" planes at same time.

Examples

⊕ Extract luma, u and v color channel component from input video frame into 3 grayscale outputs:

```
ffmpeg -i video.avi -filter_complex 'extractplanes=y+u+v[y][u][v]' -map '[y]' y.avi -map '[u]' u.avi -map '[v]' v.avi
```

fade

Apply a fade-in/out effect to the input video.

It accepts the following parameters:

type, t

The effect type can be either "in" for a fade-in, or "out" for a fade-out effect. Default is "in".

start_frame, s

Specify the number of the frame to start applying the fade effect at. Default is 0.

nb_frames, n

The number of frames that the fade effect lasts. At the end of the fade-in effect, the output video will have the same intensity as the input video. At the end of the fade-out transition, the output video will be filled with the selected **color**. Default is 25.

alpha

If set to 1, fade only alpha channel, if one exists on the input. Default value is 0.

start_time, st

Specify the timestamp (in seconds) of the frame to start to apply the fade effect. If both start_frame and start_time are specified, the fade will start at whichever comes last. Default is 0.

duration, d

The number of seconds for which the fade effect has to last. At the end of the fade-in effect the output video will have the same intensity as the input video, at the end of the fade-out transition the output video will be filled with the selected **color**. If both duration and nb_frames are specified, duration is used. Default is 0 (nb_frames is used by default).

color, c

Specify the color of the fade. Default is "black".

Examples

- ⊕ Fade in the first 30 frames of video:

```
fade=in:0:30
```

The command above is equivalent to:

```
fade=t=in:s=0:n=30
```

- ⊕ Fade out the last 45 frames of a 200-frame video:

```
fade=out:155:45  
fade=type=out:start_frame=155:nb_frames=45
```

- ⊕ Fade in the first 25 frames and fade out the last 25 frames of a 1000-frame video:

```
fade=in:0:25, fade=out:975:25
```

- ⊕ Make the first 5 frames yellow, then fade in from frame 5-24:

```
fade=in:5:20:color=yellow
```

- ⊕ Fade in alpha over first 25 frames of video:

```
fade=in:0:25:alpha=1
```

- ⊕ Make the first 5.5 seconds black, then fade in for 0.5 seconds:

```
fade=t=in:st=5.5:d=0.5
```

feedback

Apply feedback video filter.

This filter pass cropped input frames to 2nd output. From there it can be filtered with other video filters. After filter receives frame from 2nd input, that frame is combined on top of original frame from 1st input and passed to 1st output.

The typical usage is filter only part of frame.

The filter accepts the following options:

x

y Set the top left crop position.

w

h Set the crop size.

Examples

- ⊕ Blur only top left rectangular part of video frame size 100x100 with gblur filter.

```
[in][blurin]feedback=x=0:y=0:w=100:h=100[out][blurout];[blurout]gblur=8[blurin]
```

- ⊕ Draw black box on top left part of video frame of size 100x100 with drawbox filter.

```
[in][blurin]feedback=x=0:y=0:w=100:h=100[out][blurout];[blurout]drawbox=x=0:y=0:w=100:h=100:t=100[
```

fftdnoiz

Denoise frames using 3D FFT (frequency domain filtering).

The filter accepts the following options:

sigma

Set the noise sigma constant. This sets denoising strength. Default value is 1. Allowed range is from 0 to 30. Using very high sigma with low overlap may give blocking artifacts.

amount

Set amount of denoising. By default all detected noise is reduced. Default value is 1. Allowed range is from 0 to 1.

block

Set size of block in pixels, Default is 32, can be 8 to 256.

overlap

Set block overlap. Default is 0.5. Allowed range is from 0.2 to 0.8.

method

Set denoising method. Default is "wiener", can also be "hard".

prev

Set number of previous frames to use for denoising. By default is set to 0.

next Set number of next frames to to use for denoising. By default is set to 0.

planes

Set planes which will be filtered, by default are all available filtered except alpha.

fftfilt

Apply arbitrary expressions to samples in frequency domain

dc_Y

Adjust the dc value (gain) of the luma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

dc_U

Adjust the dc value (gain) of the 1st chroma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

dc_V

Adjust the dc value (gain) of the 2nd chroma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

weight_Y

Set the frequency domain weight expression for the luma plane.

weight_U

Set the frequency domain weight expression for the 1st chroma plane.

weight_V

Set the frequency domain weight expression for the 2nd chroma plane.

eval Set when the expressions are evaluated.

It accepts the following values:

init Only evaluate expressions once during the filter initialization.

frame

Evaluate expressions for each incoming frame.

Default value is **init**.

The filter accepts the following variables:

X

Y The coordinates of the current sample.

W

H The width and height of the image.

N The number of input frame, starting from 0.

WS

HS The size of FFT array for horizontal and vertical processing.

Examples

⊕ High-pass:

```
fftfilt=dc_Y=128:weight_Y='squish(1-(Y+X)/100)'
```

- ⊕ Low-pass:

```
fftfilt=dc_Y=0:weight_Y='squish((Y+X)/100-1)'
```

- ⊕ Sharpen:

```
fftfilt=dc_Y=0:weight_Y='1+squish(1-(Y+X)/100)'
```

- ⊕ Blur:

```
fftfilt=dc_Y=0:weight_Y='exp(-4 * ((Y+X)/(W+H)))'
```

field

Extract a single field from an interlaced image using stride arithmetic to avoid wasting CPU time. The output frames are marked as non-interlaced.

The filter accepts the following options:

type Specify whether to extract the top (if the value is 0 or "top") or the bottom field (if the value is 1 or "bottom").

fieldhint

Create new frames by copying the top and bottom fields from surrounding frames supplied as numbers by the hint file.

hint Set file containing hints: absolute/relative frame numbers.

There must be one line for each frame in a clip. Each line must contain two numbers separated by the comma, optionally followed by "-" or "+". Numbers supplied on each line of file can not be out of [N-1,N+1] where N is current frame number for "absolute" mode or out of [-1, 1] range for "relative" mode. First number tells from which frame to pick up top field and second number tells from which frame to pick up bottom field.

If optionally followed by "+" output frame will be marked as interlaced, else if followed by "-" output frame will be marked as progressive, else it will be marked same as input frame. If optionally followed by "t" output frame will use only top field, or in case of "b" it will use only bottom field. If line starts with "#" or ";" that line is skipped.

mode

Can be item "absolute" or "relative" or "pattern". Default is "absolute". The "pattern" mode is same as "relative" mode, except at last entry of file if there are more frames to process than "hint" file is seek back to start.

Example of first several lines of "hint" file for "relative" mode:

```
0,0 - # first frame
1,0 - # second frame, use third's frame top field and second's frame bottom field
1,0 - # third frame, use fourth's frame top field and third's frame bottom field
1,0 -
0,0 -
0,0 -
1,0 -
1,0 -
1,0 -
0,0 -
0,0 -
1,0 -
1,0 -
1,0 -
0,0 -
```

fieldmatch

Field matching filter for inverse telecine. It is meant to reconstruct the progressive frames from a telecined stream. The filter does not drop duplicated frames, so to achieve a complete inverse telecine "fieldmatch" needs to be followed by a decimation filter such as **decimate** in the filtergraph.

The separation of the field matching and the decimation is notably motivated by the possibility of inserting a de-interlacing filter fallback between the two. If the source has mixed telecined and real interlaced content, "fieldmatch" will not be able to match fields for the interlaced parts. But these remaining combed frames will be marked as interlaced, and thus can be de-interlaced by a later filter such as **yadif** before decimation.

In addition to the various configuration options, "fieldmatch" can take an optional second stream, activated through the **ppsrc** option. If enabled, the frames reconstruction will be based on the fields and frames from this second stream. This allows the first input to be pre-processed in order to help the various algorithms of the filter, while keeping the output lossless (assuming the fields are matched properly). Typically, a field-aware denoiser, or brightness/contrast adjustments can help.

Note that this filter uses the same algorithms as TIVTC/TFM (AviSynth project) and VIVTC/VFM (VapourSynth project). The later is a light clone of TFM from which "fieldmatch" is based on. While the semantic and usage are very close, some behaviour and options names can differ.

The **decimate** filter currently only works for constant frame rate input. If your input has mixed telecined (30fps) and progressive content with a lower framerate like 24fps use the following filterchain to produce the necessary cfr stream: "dejudder,fps=30000/1001,fieldmatch,decimate".

The filter accepts the following options:

order

Specify the assumed field order of the input stream. Available values are:

auto Auto detect parity (use FFmpeg's internal parity value).

bff Assume bottom field first.

tff Assume top field first.

Note that it is sometimes recommended not to trust the parity announced by the stream.

Default value is *auto*.

mode

Set the matching mode or strategy to use. **pc** mode is the safest in the sense that it won't risk creating jerkiness due to duplicate frames when possible, but if there are bad edits or blended fields it will end up outputting combed frames when a good match might actually exist. On the other hand, **pcn_ub** mode is the most risky in terms of creating jerkiness, but will almost always find a good frame if there is one. The other values are all somewhere in between **pc** and **pcn_ub** in terms of risking jerkiness and creating duplicate frames versus finding good matches in sections with bad edits, orphaned fields, blended fields, etc.

More details about p/c/n/u/b are available in **p/c/n/u/b meaning** section.

Available values are:

pc 2-way matching (p/c)

pc_n

2-way matching, and trying 3rd match if still combed (p/c + n)

pc_u

2-way matching, and trying 3rd match (same order) if still combed (p/c + u)

pc_n_ub

2-way matching, trying 3rd match if still combed, and trying 4th/5th matches if still combed (p/c + n + u/b)

pcn 3-way matching (p/c/n)

pcn_ub

3-way matching, and trying 4th/5th matches if all 3 of the original matches are detected as combed (p/c/n + u/b)

The parenthesis at the end indicate the matches that would be used for that mode assuming **order=fff** (and **field** on *auto* or *top*).

In terms of speed **pc** mode is by far the fastest and **pcn_ub** is the slowest.

Default value is *pc_n*.

ppsrc

Mark the main input stream as a pre-processed input, and enable the secondary input stream as the clean source to pick the fields from. See the filter introduction for more details. It is similar to the **clip2** feature from VFM/TFM.

Default value is 0 (disabled).

field

Set the field to match from. It is recommended to set this to the same value as **order** unless you experience matching failures with that setting. In certain circumstances changing the field that is used to match from can have a large impact on matching performance. Available values are:

auto Automatic (same value as **order**).

bottom

Match from the bottom field.

top Match from the top field.

Default value is *auto*.

mchroma

Set whether or not chroma is included during the match comparisons. In most cases it is recommended to leave this enabled. You should set this to 0 only if your clip has bad chroma problems such as heavy rainbowing or other artifacts. Setting this to 0 could also be used to speed things up at the cost of some accuracy.

Default value is 1.

y0

y1 These define an exclusion band which excludes the lines between **y0** and **y1** from being included in the field matching decision. An exclusion band can be used to ignore subtitles, a logo, or other things that may interfere with the matching. **y0** sets the starting scan line and **y1** sets the ending line; all lines in between **y0** and **y1** (including **y0** and **y1**) will be ignored. Setting **y0** and **y1** to the same value will disable the feature. **y0** and **y1** defaults to 0.

scthresh

Set the scene change detection threshold as a percentage of maximum change on the luma plane. Good values are in the "[8.0, 14.0]" range. Scene change detection is only relevant in case **combmatch=sc**. The range for **scthresh** is "[0.0, 100.0]".

Default value is 12.0.

combmatch

When **combmatch** is not *none*, "fieldmatch" will take into account the combed scores of matches when deciding what match to use as the final match. Available values are:

none

No final matching based on combed scores.

sc Combed scores are only used when a scene change is detected.

full Use combed scores all the time.

Default is *sc*.

combdbg

Force "fieldmatch" to calculate the combed metrics for certain matches and print them. This setting is known as **micout** in TFM/VFM vocabulary. Available values are:

none

No forced calculation.

pcn Force p/c/n calculations.

pcnub

Force p/c/n/u/b calculations.

Default value is *none*.

cthresh

This is the area combing threshold used for combed frame detection. This essentially controls how "strong" or "visible" combing must be to be detected. Larger values mean combing must be more visible and smaller values mean combing can be less visible or strong and still be detected. Valid settings are from "-1" (every pixel will be detected as combed) to 255 (no pixel will be detected as combed). This is basically a pixel difference value. A good range is "[8, 12]".

Default value is 9.

chroma

Sets whether or not chroma is considered in the combed frame decision. Only disable this if your source has chroma problems (rainbowing, etc.) that are causing problems for the combed frame detection with chroma enabled. Actually, using **chroma=0** is usually more reliable, except for the case where there is chroma only combing in the source.

Default value is 0.

blockx

blocky

Respectively set the x-axis and y-axis size of the window used during combed frame detection. This has to do with the size of the area in which **combpel** pixels are required to be detected as combed for a frame to be declared combed. See the **combpel** parameter description for more info. Possible values are any number that is a power of 2 starting at 4 and going up to 512.

Default value is 16.

combpel

The number of combed pixels inside any of the **blocky** by **blockx** size blocks on the frame for the frame to be detected as combed. While **cthresh** controls how "visible" the combing must be, this setting controls "how much" combing there must be in any localized area (a window defined by the **blockx** and **blocky** settings) on the frame. Minimum value is 0 and maximum is "blocky x

blockx" (at which point no frames will ever be detected as combed). This setting is known as **MI** in TFM/VFM vocabulary.

Default value is 80.

p/c/n/w/b meaning

p/c/n

We assume the following telecined stream:

Top fields: 1 2 2 3 4
Bottom fields: 1 2 3 4 4

The numbers correspond to the progressive frame the fields relate to. Here, the first two frames are progressive, the 3rd and 4th are combed, and so on.

When "fieldmatch" is configured to run a matching from bottom (**field=bottom**) this is how this input stream get transformed:

Input stream:
T 1 2 2 3 4
B 1 2 3 4 4 <-- matching reference

Matches: c c n n c

Output stream:
T 1 2 3 4 4
B 1 2 3 4 4

As a result of the field matching, we can see that some frames get duplicated. To perform a complete inverse telecine, you need to rely on a decimation filter after this operation. See for instance the **decimate** filter.

The same operation now matching from top fields (**field=top**) looks like this:

Input stream:
T 1 2 2 3 4 <-- matching reference
B 1 2 3 4 4

Matches: c c p p c

Output stream:

```

T  1 2 2 3 4
B  1 2 2 3 4

```

In these examples, we can see what p , c and n mean; basically, they refer to the frame and field of the opposite parity:

*< p matches the field of the opposite parity in the previous frame>

*< c matches the field of the opposite parity in the current frame>

*< n matches the field of the opposite parity in the next frame>

u/b

The u and b matching are a bit special in the sense that they match from the opposite parity flag. In the following examples, we assume that we are currently matching the 2nd frame (Top:2, bottom:2). According to the match, a 'x' is placed above and below each matched fields.

With bottom matching (**field=bottom**):

```

Match:      c      p      n      b      u

           x  x          x  x      x
Top        1 2 2  1 2 2  1 2 2  1 2 2  1 2 2
Bottom     1 2 3  1 2 3  1 2 3  1 2 3  1 2 3
           x  x          x  x      x

```

Output frames:

```

      2      1      2      2      2
      2      2      2      1      3

```

With top matching (**field=top**):

```

Match:      c      p      n      b      u

           x  x          x  x      x
Top        1 2 2  1 2 2  1 2 2  1 2 2  1 2 2
Bottom     1 2 3  1 2 3  1 2 3  1 2 3  1 2 3
           x  x          x  x      x

```

Output frames:

2	2	2	1	2
2	1	3	2	2

Examples

Simple IVTC of a top field first telecined stream:

```
fieldmatch=order=tff:combmach=none, decimate
```

Advanced IVTC, with fallback on **yadif** for still combed frames:

```
fieldmatch=order=tff:combmach=full, yadif=deint=interlaced, decimate
```

fieldorder

Transform the field order of the input video.

It accepts the following parameters:

order

The output field order. Valid values are *tff* for top field first or *bff* for bottom field first.

The default value is **tff**.

The transformation is done by shifting the picture content up or down by one line, and filling the remaining line with appropriate picture content. This method is consistent with most broadcast field order converters.

If the input video is not flagged as being interlaced, or it is already flagged as being of the required output field order, then this filter does not alter the incoming video.

It is very useful when converting to or from PAL DV material, which is bottom field first.

For example:

```
ffmpeg -i in.vob -vf "fieldorder=bff" out.dv
```

fifo, afifo

Buffer input images and send them when they are requested.

It is mainly useful when auto-inserted by the libavfilter framework.

It does not take parameters.

fillborders

Fill borders of the input video, without changing video stream dimensions. Sometimes video can have garbage at the four edges and you may not want to crop video input to keep size multiple of some number.

This filter accepts the following options:

left Number of pixels to fill from left border.

right
Number of pixels to fill from right border.

top Number of pixels to fill from top border.

bottom
Number of pixels to fill from bottom border.

mode
Set fill mode.

It accepts the following values:

smear
fill pixels using outermost pixels

mirror
fill pixels using mirroring (half sample symmetric)

fixed
fill pixels with constant value

reflect
fill pixels using reflecting (whole sample symmetric)

wrap
fill pixels using wrapping

fade fade pixels to constant value

margins

fill pixels at top and bottom with weighted averages pixels near borders

Default is *smear*.

color

Set color for pixels in fixed or fade mode. Default is *black*.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

find_rect

Find a rectangular object

It accepts the following options:

object

Filepath of the object image, needs to be in gray8.

threshold

Detection threshold, default is 0.5.

mipmaps

Number of mipmaps, default is 3.

xmin, ymin, xmax, ymax

Specifies the rectangle in which to search.

discard

Discard frames where object is not detected. Default is disabled.

Examples

⊕ Cover a rectangular object by the supplied image of a given video using **ffmpeg**:

```
ffmpeg -i file.ts -vf find_rect=newref.pgm,cover_rect=cover.jpg:mode=cover new.mkv
```

floodfill

Flood area with values of same pixel components with another values.

It accepts the following options:

- x** Set pixel x coordinate.
- y** Set pixel y coordinate.
- s0** Set source #0 component value.
- s1** Set source #1 component value.
- s2** Set source #2 component value.
- s3** Set source #3 component value.
- d0** Set destination #0 component value.
- d1** Set destination #1 component value.
- d2** Set destination #2 component value.
- d3** Set destination #3 component value.

format

Convert the input video to one of the specified pixel formats. Libavfilter will try to pick one that is suitable as input to the next filter.

It accepts the following parameters:

pix_fmts

A '|' -separated list of pixel format names, such as "pix_fmts=yuv420p|monow|rgb24".

Examples

- ⊕ Convert the input video to the *yuv420p* format

```
format=pix_fmts=yuv420p
```

Convert the input video to any of the formats in the list

```
format=pix_fmts=yuv420p|yuv444p|yuv410p
```

fps

Convert the video to specified constant frame rate by duplicating or dropping frames as necessary.

It accepts the following parameters:

fps The desired output frame rate. It accepts expressions containing the following constants:

source_fps

The input's frame rate

ntsc NTSC frame rate of "30000/1001"

pal PAL frame rate of 25.0

film Film frame rate of 24.0

ntsc_film

NTSC-film frame rate of "24000/1001"

The default is 25.

start_time

Assume the first PTS should be the given value, in seconds. This allows for padding/trimming at the start of stream. By default, no assumption is made about the first frame's expected PTS, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with duplicates of the first frame if a video stream starts after the audio stream or to trim any frames with a negative PTS.

round

Timestamp (PTS) rounding method.

Possible values are:

zero round towards 0

inf round away from 0

down

round towards -infinity

up round towards +infinity

near round to nearest

The default is "near".

eof_action

Action performed when reading the last frame.

Possible values are:

round

Use same timestamp rounding method as used for other frames.

pass Pass through last frame if input duration has not been reached yet.

The default is "round".

Alternatively, the options can be specified as a flat string: `fps[:start_time[:round]]`.

See also the **setpts** filter.

Examples

- ⊕ A typical usage in order to set the fps to 25:

```
fps=fps=25
```

- ⊕ Sets the fps to 24, using abbreviation and rounding method to round to nearest:

```
fps=fps=film:round=near
```

framepack

Pack two different video streams into a stereoscopic video, setting proper metadata on supported codecs. The two views should have the same size and framerate and processing will stop when the

shorter video ends. Please note that you may conveniently adjust view properties with the **scale** and **fps** filters.

It accepts the following parameters:

format

The desired packing format. Supported values are:

sbs The views are next to each other (default).

tab The views are on top of each other.

lines

The views are packed by line.

columns

The views are packed by column.

frameseq

The views are temporally interleaved.

Some examples:

```
# Convert left and right views into a frame-sequential video
```

```
ffmpeg -i LEFT -i RIGHT -filter_complex framepack=frameseq OUTPUT
```

```
# Convert views into a side-by-side video with the same output resolution as the input
```

```
ffmpeg -i LEFT -i RIGHT -filter_complex [0:v]scale=w=iw/2[left],[1:v]scale=w=iw/2[right],[left][right]framepack
```

framerate

Change the frame rate by interpolating new video output frames from the source frames.

This filter is not designed to function correctly with interlaced media. If you wish to change the frame rate of interlaced media then you are required to deinterlace before this filter and re-interlace after this filter.

A description of the accepted options follows.

fps Specify the output frames per second. This option can also be specified as a value alone. The default is 50.

interp_start

Specify the start of a range where the output frame will be created as a linear interpolation of two frames. The range is [0-255], the default is 15.

interp_end

Specify the end of a range where the output frame will be created as a linear interpolation of two frames. The range is [0-255], the default is 240.

scene

Specify the level at which a scene change is detected as a value between 0 and 100 to indicate a new scene; a low value reflects a low probability for the current frame to introduce a new scene, while a higher value means the current frame is more likely to be one. The default is 8.2.

flags

Specify flags influencing the filter process.

Available value for *flags* is:

scene_change_detect, scd

Enable scene change detection using the value of the option *scene*. This flag is enabled by default.

framestep

Select one frame every N-th frame.

This filter accepts the following option:

step Select frame after every "step" frames. Allowed values are positive integers higher than 0. Default value is 1.

freezedetect

Detect frozen video.

This filter logs a message and sets frame metadata when it detects that the input video has no significant change in content during a specified duration. Video freeze detection calculates the mean average absolute difference of all the components of video frames and compares it to a noise floor.

The printed times and duration are expressed in seconds. The "lavfi.freezedetect.freeze_start" metadata key is set on the first frame whose timestamp equals or exceeds the detection duration and it contains the timestamp of the first frame of the freeze. The "lavfi.freezedetect.freeze_duration" and

"lavfi.freeze detect.freeze_end" metadata keys are set on the first frame after the freeze.

The filter accepts the following options:

noise, n

Set noise tolerance. Can be specified in dB (in case "dB" is appended to the specified value) or as a difference ratio between 0 and 1. Default is -60dB, or 0.001.

duration, d

Set freeze duration until notification (default is 2 seconds).

freezeframes

Freeze video frames.

This filter freezes video frames using frame from 2nd input.

The filter accepts the following options:

first Set number of first frame from which to start freeze.

last Set number of last frame from which to end freeze.

replace

Set number of frame from 2nd input which will be used instead of replaced frames.

frei0r

Apply a frei0r effect to the input video.

To enable the compilation of this filter, you need to install the frei0r header and configure FFmpeg with "--enable-frei0r".

It accepts the following parameters:

filter_name

The name of the frei0r effect to load. If the environment variable **FREI0R_PATH** is defined, the frei0r effect is searched for in each of the directories specified by the colon-separated list in **FREI0R_PATH**. Otherwise, the standard frei0r paths are searched, in this order:
HOME/.frei0r-1/lib/, /usr/local/lib/frei0r-1/, /usr/lib/frei0r-1/.

filter_params

A '|' -separated list of parameters to pass to the frei0r effect.

A frei0r effect parameter can be a boolean (its value is either "y" or "n"), a double, a color (specified as *R/G/B*, where *R*, *G*, and *B* are floating point numbers between 0.0 and 1.0, inclusive) or a color description as specified in the "**Color**" section in the **ffmpeg-utils manual**, a position (specified as *X/Y*, where *X* and *Y* are floating point numbers) and/or a string.

The number and types of parameters depend on the loaded effect. If an effect parameter is not specified, the default value is set.

Examples

- ⊕ Apply the distort0r effect, setting the first two double parameters:

```
frei0r=filter_name=distort0r:filter_params=0.5|0.01
```

- ⊕ Apply the colordistance effect, taking a color as the first parameter:

```
frei0r=colordistance:0.2/0.3/0.4
```

```
frei0r=colordistance:violet
```

```
frei0r=colordistance:0x112233
```

- ⊕ Apply the perspective effect, specifying the top left and top right image positions:

```
frei0r=perspective:0.2/0.2|0.8/0.2
```

For more information, see <<http://frei0r.dyne.org>>

Commands

This filter supports the **filter_params** option as **commands**.

fspp

Apply fast and simple postprocessing. It is a faster version of **spp**.

It splits (I)DCT into horizontal/vertical passes. Unlike the simple post-processing filter, one of them is performed once per block, not per pixel. This allows for much higher speed.

The filter accepts the following options:

quality

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 4-5. Default value is 4.

qp Force a constant quantization parameter. It accepts an integer in range 0-63. If not set, the filter will use the QP from the video stream (if available).

strength

Set filter strength. It accepts an integer in range -15 to 32. Lower values mean more details but also more artifacts, while higher values make the image smoother but also blurrier. Default value is 0 X PSNR optimal.

use_bframe_qp

Enable the use of the QP from the B-Frames if set to 1. Using this option may cause flicker since the B-Frames have often larger QP. Default is 0 (not enabled).

gblur

Apply Gaussian blur filter.

The filter accepts the following options:

sigma

Set horizontal sigma, standard deviation of Gaussian blur. Default is 0.5.

steps

Set number of steps for Gaussian approximation. Default is 1.

planes

Set which planes to filter. By default all planes are filtered.

sigmaV

Set vertical sigma, if negative it will be same as "sigma". Default is "-1".

Commands

This filter supports same commands as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

geq

Apply generic equation to each pixel.

The filter accepts the following options:

lum_expr, lum

Set the luminance expression.

cb_expr, cb

Set the chrominance blue expression.

cr_expr, cr

Set the chrominance red expression.

alpha_expr, a

Set the alpha expression.

red_expr, r

Set the red expression.

green_expr, g

Set the green expression.

blue_expr, b

Set the blue expression.

The colorspace is selected according to the specified options. If one of the **lum_expr**, **cb_expr**, or **cr_expr** options is specified, the filter will automatically select a YCbCr colorspace. If one of the **red_expr**, **green_expr**, or **blue_expr** options is specified, it will select an RGB colorspace.

If one of the chrominance expression is not defined, it falls back on the other one. If no alpha expression is specified it will evaluate to opaque value. If none of chrominance expressions are specified, they will evaluate to the luminance expression.

The expressions can use the following variables and functions:

N The sequential number of the filtered frame, starting from 0.

X

Y The coordinates of the current sample.

W

H The width and height of the image.

SW

SH Width and height scale depending on the currently filtered plane. It is the ratio between the corresponding luma plane number of pixels and the current plane ones. E.g. for YUV4:2:0 the values are "1,1" for the luma plane, and "0.5,0.5" for chroma planes.

T Time of the current frame, expressed in seconds.

p(x, y)

Return the value of the pixel at location (x,y) of the current plane.

lum(x, y)

Return the value of the pixel at location (x,y) of the luminance plane.

cb(x, y)

Return the value of the pixel at location (x,y) of the blue-difference chroma plane. Return 0 if there is no such plane.

cr(x, y)

Return the value of the pixel at location (x,y) of the red-difference chroma plane. Return 0 if there is no such plane.

r(x, y)**g(x, y)****b(x, y)**

Return the value of the pixel at location (x,y) of the red/green/blue component. Return 0 if there is no such component.

alpha(x, y)

Return the value of the pixel at location (x,y) of the alpha plane. Return 0 if there is no such plane.

psum(x,y), lumsum(x, y), cbsum(x,y), crsum(x,y), rsum(x,y), gsum(x,y), bsum(x,y), alphasum(x,y)

Sum of sample values in the rectangle from (0,0) to (x,y), this allows obtaining sums of samples within a rectangle. See the functions without the sum postfix.

interpolation

Set one of interpolation methods:

nearest, n
bilinear, b

Default is bilinear.

For functions, if x and y are outside the area, the value will be automatically clipped to the closer edge.

Please note that this filter can use multiple threads in which case each slice will have its own expression state. If you want to use only a single expression state because your expressions depend on previous state then you should limit the number of filter threads to 1.

Examples

- ⊕ Flip the image horizontally:

```
geq=p(W-X,Y)
```

- ⊕ Generate a bidimensional sine wave, with angle "PI/3" and a wavelength of 100 pixels:

```
geq=128 + 100*sin(2*(PI/100)*(cos(PI/3)*(X-50*T) + sin(PI/3)*Y)):128:128
```

- ⊕ Generate a fancy enigmatic moving light:

```
nullsrc=s=256x256,geq=random(1)/hypot(X-cos(N*0.07)*W/2-W/2,Y-sin(N*0.09)*H/2-H/2)^2*1000000*s
```

- ⊕ Generate a quick emboss effect:

```
format=gray,geq=lum_expr='(p(X,Y)+(256-p(X-4,Y-4)))/2'
```

- ⊕ Modify RGB components depending on pixel position:

```
geq=r='X/W*r(X,Y)';g='(1-X/W)*g(X,Y)';b='(H-Y)/H*b(X,Y)'
```

- ⊕ Create a radial gradient that is the same size as the input (also see the **vignette** filter):

```
geq=lum=255*gauss((X/W-0.5)*3)*gauss((Y/H-0.5)*3)/gauss(0)/gauss(0),format=gray
```

gradfun

Fix the banding artifacts that are sometimes introduced into nearly flat regions by truncation to 8-bit color depth. Interpolate the gradients that should go where the bands are, and dither them.

It is designed for playback only. Do not use it prior to lossy compression, because compression tends to lose the dither and bring back the bands.

It accepts the following parameters:

strength

The maximum amount by which the filter will change any one pixel. This is also the threshold for detecting nearly flat regions. Acceptable values range from .51 to 64; the default value is 1.2. Out-of-range values will be clipped to the valid range.

radius

The neighborhood to fit the gradient to. A larger radius makes for smoother gradients, but also prevents the filter from modifying the pixels near detailed regions. Acceptable values are 8-32; the default value is 16. Out-of-range values will be clipped to the valid range.

Alternatively, the options can be specified as a flat string: *strength[:radius]*

Examples

- ⊕ Apply the filter with a 3.5 strength and radius of 8:

```
gradfun=3.5:8
```

- ⊕ Specify radius, omitting the strength (which will fall-back to the default value):

```
gradfun=radius=8
```

graphmonitor

Show various filtergraph stats.

With this filter one can debug complete filtergraph. Especially issues with links filling with queued frames.

The filter accepts the following options:

size, s

Set video output size. Default is *hd720*.

opacity, o

Set video opacity. Default is *0.9*. Allowed range is from *0* to *1*.

mode, m

Set output mode, can be *full* or *compact*. In *compact* mode only filters with some queued frames have displayed stats.

flags, f

Set flags which enable which stats are shown in video.

Available values for flags are:

queue

Display number of queued frames in each link.

frame_count_in

Display number of frames taken from filter.

frame_count_out

Display number of frames given out from filter.

frame_count_delta

Display delta number of frames between above two values.

pts Display current filtered frame pts.

pts_delta

Display pts delta between current and previous frame.

time

Display current filtered frame time.

time_delta

Display time delta between current and previous frame.

timebase

Display time base for filter link.

format

Display used format for filter link.

size Display video size or number of audio channels in case of audio used by filter link.

rate Display video frame rate or sample rate in case of audio used by filter link.

eof Display link output status.

sample_count_in

Display number of samples taken from filter.

sample_count_out

Display number of samples given out from filter.

sample_count_delta

Display delta number of samples between above two values.

rate, r

Set upper limit for video rate of output stream, Default value is 25. This guarantee that output video frame rate will not be higher than this value.

grayworld

A color constancy filter that applies color correction based on the grayworld assumption

See:

<https://www.researchgate.net/publication/275213614_A_New_Color_Correction_Method_for_Underwater_Im>

The algorithm uses linear light, so input data should be linearized beforehand (and possibly correctly tagged).

```
ffmpeg -i INPUT -vf zscale=transfer=linear,grayworld,zscale=transfer=bt709,format=yuv420p OUTPUT
```

greyedge

A color constancy variation filter which estimates scene illumination via grey edge algorithm and corrects the scene colors accordingly.

See: <<https://staff.science.uva.nl/th.gevers/pub/GeversTIP07.pdf>>

The filter accepts the following options:

difford

The order of differentiation to be applied on the scene. Must be chosen in the range [0,2] and default value is 1.

minknorm

The Minkowski parameter to be used for calculating the Minkowski distance. Must be chosen in the range [0,20] and default value is 1. Set to 0 for getting max value instead of calculating Minkowski distance.

sigma

The standard deviation of Gaussian blur to be applied on the scene. Must be chosen in the range [0,1024.0] and default value = 1. $\text{floor}(\text{sigma} * \text{break_off_sigma}(3))$ can't be equal to 0 if *difford* is greater than 0.

Examples

- ⊕ Grey Edge:

```
greyedge=difford=1:minknorm=5:sigma=2
```

- ⊕ Max Edge:

```
greyedge=difford=1:minknorm=0:sigma=2
```

guided

Apply guided filter for edge-preserving smoothing, dehazing and so on.

The filter accepts the following options:

radius

Set the box radius in pixels. Allowed range is 1 to 20. Default is 3.

eps Set regularization parameter (with square). Allowed range is 0 to 1. Default is 0.01.

mode

Set filter mode. Can be "basic" or "fast". Default is "basic".

sub Set subsampling ratio for "fast" mode. Range is 2 to 64. Default is 4. No subsampling occurs in "basic" mode.

guidance

Set guidance mode. Can be "off" or "on". Default is "off". If "off", single input is required. If "on", two inputs of the same resolution and pixel format are required. The second input serves as the guidance.

planes

Set planes to filter. Default is first only.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Edge-preserving smoothing with guided filter:

```
ffmpeg -i in.png -vf guided out.png
```

- ⊕ Dehazing, structure-transferring filtering, detail enhancement with guided filter. For the generation of guidance image, refer to paper "Guided Image Filtering". See: <http://kaiminghe.com/publications/pami12guidedfilter.pdf>.

```
ffmpeg -i in.png -i guidance.png -filter_complex guided=guidance=on out.png
```

haldclut

Apply a Hald CLUT to a video stream.

First input is the video stream to process, and second one is the Hald CLUT. The Hald CLUT input can be a simple picture or a complete video stream.

The filter accepts the following options:

- clut** Set which CLUT video frames will be processed from second input stream, can be *first* or *all*. Default is *all*.

shortest

Force termination when the shortest input terminates. Default is 0.

repeatlast

Continue applying the last CLUT after the end of the stream. A value of 0 disable the filter after the last frame of the CLUT is reached. Default is 1.

"haldclut" also has the same interpolation options as **lut3d** (both filters share the same internals).

This filter also supports the **framesync** options.

More information about the Hald CLUT can be found on Eskil Steenberg's website (Hald CLUT author) at <<http://www.quelsolaar.com/technology/clut.html>>.

Commands

This filter supports the "interp" option as **commands**.

Workflow examples

Hald CLUT video stream

Generate an identity Hald CLUT stream altered with various effects:

```
ffmpeg -f lavfi -i B<haldclutsrc>=8 -vf "hue=H=2*PI*t:s=sin(2*PI*t)+1, curves=cross_process" -t 10 -c:v ffv1 clut
```

Note: make sure you use a lossless codec.

Then use it with "haldclut" to apply it on some random stream:

```
ffmpeg -f lavfi -i mandelbrot -i clut.nut -filter_complex '[0][1] haldclut' -t 20 mandelclut.mkv
```

The Hald CLUT will be applied to the 10 first seconds (duration of *clut.nut*), then the latest picture of that CLUT stream will be applied to the remaining frames of the "mandelbrot" stream.

Hald CLUT with preview

A Hald CLUT is supposed to be a squared image of "Level*Level*Level" by "Level*Level*Level" pixels. For a given Hald CLUT, Ffmpeg will select the biggest possible square starting at the top left of the picture. The remaining padding pixels (bottom or right) will be ignored. This area can be used to add a preview of the Hald CLUT.

Typically, the following generated Hald CLUT will be supported by the "haldclut" filter:

```
ffmpeg -f lavfi -i B<haldclutsrc>=8 -vf "
  pad=iw+320 [padded_clut];
  smptebars=s=320x256, split [a][b];
  [padded_clut][a] overlay=W-320:h, curves=color_negative [main];
  [main][b] overlay=W-320" -frames:v 1 clut.png
```

It contains the original and a preview of the effect of the CLUT: SMPTE color bars are displayed on

the right-top, and below the same color bars processed by the color changes.

Then, the effect of this Hald CLUT can be visualized with:

```
ffplay input.mkv -vf "movie=clut.png, [in] haldclut"
```

hflip

Flip the input video horizontally.

For example, to horizontally flip the input video with **ffmpeg**:

```
ffmpeg -i in.avi -vf "hflip" out.avi
```

histeq

This filter applies a global color histogram equalization on a per-frame basis.

It can be used to correct video that has a compressed range of pixel intensities. The filter redistributes the pixel intensities to equalize their distribution across the intensity range. It may be viewed as an "automatically adjusting contrast filter". This filter is useful only for correcting degraded or poorly captured source video.

The filter accepts the following options:

strength

Determine the amount of equalization to be applied. As the strength is reduced, the distribution of pixel intensities more-and-more approaches that of the input frame. The value must be a float number in the range [0,1] and defaults to 0.200.

intensity

Set the maximum intensity that can be generated and scale the output values appropriately. The strength should be set as desired and then the intensity can be limited if needed to avoid washing-out. The value must be a float number in the range [0,1] and defaults to 0.210.

antibanding

Set the antibanding level. If enabled the filter will randomly vary the luminance of output pixels by a small amount to avoid banding of the histogram. Possible values are "none", "weak" or "strong". It defaults to "none".

histogram

Compute and draw a color distribution histogram for the input video.

The computed histogram is a representation of the color component distribution in an image.

Standard histogram displays the color components distribution in an image. Displays color graph for each color component. Shows distribution of the Y, U, V, A or R, G, B components, depending on input format, in the current frame. Below each graph a color component scale meter is shown.

The filter accepts the following options:

level_height

Set height of level. Default value is 200. Allowed range is [50, 2048].

scale_height

Set height of color scale. Default value is 12. Allowed range is [0, 40].

display_mode

Set display mode. It accepts the following values:

stack

Per color component graphs are placed below each other.

parade

Per color component graphs are placed side by side.

overlay

Presents information identical to that in the "parade", except that the graphs representing color components are superimposed directly over one another.

Default is "stack".

levels_mode

Set mode. Can be either "linear", or "logarithmic". Default is "linear".

components

Set what color components to display. Default is 7.

fgopacity

Set foreground opacity. Default is 0.7.

bgopacity

Set background opacity. Default is 0.5.

colors_mode

Set colors mode. It accepts the following values:

whiteonblack
blackonwhite
whiteongray
blackongray
coloronblack
coloronwhite
colorongray
blackoncolor
whiteoncolor
grayoncolor

Default is "whiteonblack".

Examples

- ⊕ Calculate and draw histogram:

```
ffplay -i input -vf histogram
```

hqdn3d

This is a high precision/quality 3d denoise filter. It aims to reduce image noise, producing smooth images and making still images really still. It should enhance compressibility.

It accepts the following optional parameters:

luma_spatial

A non-negative floating point number which specifies spatial luma strength. It defaults to 4.0.

chroma_spatial

A non-negative floating point number which specifies spatial chroma strength. It defaults to $3.0 * luma_spatial / 4.0$.

luma_tmp

A floating point number which specifies luma temporal strength. It defaults to $6.0 * luma_spatial / 4.0$.

chroma_tmp

A floating point number which specifies chroma temporal strength. It defaults to $luma_tmp * chroma_spatial / luma_spatial$.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

hwdownload

Download hardware frames to system memory.

The input must be in hardware frames, and the output a non-hardware format. Not all formats will be supported on the output - it may be necessary to insert an additional **format** filter immediately following in the graph to get the output in a supported format.

hwmap

Map hardware frames to system memory or to another device.

This filter has several different modes of operation; which one is used depends on the input and output formats:

- ⊕ Hardware frame input, normal frame output

Map the input frames to system memory and pass them to the output. If the original hardware frame is later required (for example, after overlaying something else on part of it), the **hwmap** filter can be used again in the next mode to retrieve it.

- ⊕ Normal frame input, hardware frame output

If the input is actually a software-mapped hardware frame, then unmap it - that is, return the original hardware frame.

Otherwise, a device must be provided. Create new hardware surfaces on that device for the output, then map them back to the software format at the input and give those frames to the preceding filter. This will then act like the **hwupload** filter, but may be able to avoid an additional copy when the input is already in a compatible format.

- ⊕ Hardware frame input and output

A device must be supplied for the output, either directly or with the **derive_device** option. The input and output devices must be of different types and compatible - the exact meaning of this is system-dependent, but typically it means that they must refer to the same underlying hardware context (for example, refer to the same graphics card).

If the input frames were originally created on the output device, then unmap to retrieve the original frames.

Otherwise, map the frames to the output device - create new hardware frames on the output corresponding to the frames on the input.

The following additional parameters are accepted:

mode

Set the frame mapping mode. Some combination of:

read The mapped frame should be readable.

write

The mapped frame should be writeable.

overwrite

The mapping will always overwrite the entire frame.

This may improve performance in some cases, as the original contents of the frame need not be loaded.

direct

The mapping must not involve any copying.

Indirect mappings to copies of frames are created in some cases where either direct mapping is not possible or it would have unexpected properties. Setting this flag ensures that the mapping is direct and will fail if that is not possible.

Defaults to *read+write* if not specified.

derive_device type

Rather than using the device supplied at initialisation, instead derive a new device of type *type* from the device the input frames exist on.

reverse

In a hardware to hardware mapping, map in reverse - create frames in the sink and map them back to the source. This may be necessary in some cases where a mapping in one direction is required but only the opposite direction is supported by the devices being used.

This option is dangerous - it may break the preceding filter in undefined ways if there are any additional constraints on that filter's output. Do not use it without fully understanding the implications of its use.

hwupload

Upload system memory frames to hardware surfaces.

The device to upload to must be supplied when the filter is initialised. If using ffmpeg, select the appropriate device with the **-filter_hw_device** option or with the **derive_device** option. The input and output devices must be of different types and compatible - the exact meaning of this is system-dependent, but typically it means that they must refer to the same underlying hardware context (for example, refer to the same graphics card).

The following additional parameters are accepted:

derive_device *type*

Rather than using the device supplied at initialisation, instead derive a new device of type *type* from the device the input frames exist on.

hwupload_cuda

Upload system memory frames to a CUDA device.

It accepts the following optional parameters:

device

The number of the CUDA device to use

hqx

Apply a high-quality magnification filter designed for pixel art. This filter was originally created by Maxim Stepin.

It accepts the following option:

n Set the scaling dimension: 2 for "hq2x", 3 for "hq3x" and 4 for "hq4x". Default is 3.

hstack

Stack input videos horizontally.

All streams must be of same pixel format and of same height.

Note that this filter is faster than using **overlay** and **pad** filter to create same output.

The filter accepts the following option:

inputs

Set number of input streams. Default is 2.

shortest

If set to 1, force the output to terminate when the shortest input terminates. Default value is 0.

hsvhold

Turns a certain HSV range into gray values.

This filter measures color difference between set HSV color in options and ones measured in video stream. Depending on options, output colors can be changed to be gray or not.

The filter accepts the following options:

hue Set the hue value which will be used in color difference calculation. Allowed range is from -360 to 360. Default value is 0.

sat Set the saturation value which will be used in color difference calculation. Allowed range is from -1 to 1. Default value is 0.

val Set the value which will be used in color difference calculation. Allowed range is from -1 to 1. Default value is 0.

similarity

Set similarity percentage with the key color. Allowed range is from 0 to 1. Default value is 0.01.

0.00001 matches only the exact key color, while 1.0 matches everything.

blend

Blend percentage. Allowed range is from 0 to 1. Default value is 0.

0.0 makes pixels either fully gray, or not gray at all.

Higher values result in more gray pixels, with a higher gray pixel the more similar the pixels color is to the key color.

hsvkey

Turns a certain HSV range into transparency.

This filter measures color difference between set HSV color in options and ones measured in video stream. Depending on options, output colors can be changed to transparent by adding alpha channel.

The filter accepts the following options:

hue Set the hue value which will be used in color difference calculation. Allowed range is from -360 to 360. Default value is 0.

sat Set the saturation value which will be used in color difference calculation. Allowed range is from -1 to 1. Default value is 0.

val Set the value which will be used in color difference calculation. Allowed range is from -1 to 1. Default value is 0.

similarity

Set similarity percentage with the key color. Allowed range is from 0 to 1. Default value is 0.01.

0.00001 matches only the exact key color, while 1.0 matches everything.

blend

Blend percentage. Allowed range is from 0 to 1. Default value is 0.

0.0 makes pixels either fully transparent, or not transparent at all.

Higher values result in semi-transparent pixels, with a higher transparency the more similar the pixels color is to the key color.

hue

Modify the hue and/or the saturation of the input.

It accepts the following parameters:

- h** Specify the hue angle as a number of degrees. It accepts an expression, and defaults to "0".
- s** Specify the saturation in the [-10,10] range. It accepts an expression and defaults to "1".
- H** Specify the hue angle as a number of radians. It accepts an expression, and defaults to "0".
- b** Specify the brightness in the [-10,10] range. It accepts an expression and defaults to "0".

h and **H** are mutually exclusive, and can't be specified at the same time.

The **b**, **h**, **H** and **s** option values are expressions containing the following constants:

- n** frame count of the input frame starting from 0
- pts** presentation timestamp of the input frame expressed in time base units
- r** frame rate of the input video, NAN if the input frame rate is unknown
- t** timestamp expressed in seconds, NAN if the input timestamp is unknown
- tb** time base of the input video

Examples

- ⊕ Set the hue to 90 degrees and the saturation to 1.0:

```
hue=h=90:s=1
```

- ⊕ Same command but expressing the hue in radians:

```
hue=H=PI/2:s=1
```

- ⊕ Rotate hue and make the saturation swing between 0 and 2 over a period of 1 second:

```
hue="H=2*PI*t: s=sin(2*PI*t)+1"
```

- ⊕ Apply a 3 seconds saturation fade-in effect starting at 0:

```
hue="s=min(t/3,1)"
```

The general fade-in expression can be written as:

$$\text{hue}=\text{"s}=\text{min}(0\backslash, \text{max}((\text{t}-\text{START})/\text{DURATION}\backslash, 1))\text{"}$$

- ⊕ Apply a 3 seconds saturation fade-out effect starting at 5 seconds:

$$\text{hue}=\text{"s}=\text{max}(0\backslash, \text{min}(1\backslash, (8-\text{t})/3))\text{"}$$

The general fade-out expression can be written as:

$$\text{hue}=\text{"s}=\text{max}(0\backslash, \text{min}(1\backslash, (\text{START}+\text{DURATION}-\text{t})/\text{DURATION}))\text{"}$$

Commands

This filter supports the following commands:

b

s

h

- H** Modify the hue and/or the saturation and/or brightness of the input video. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

huesaturation

Apply hue-saturation-intensity adjustments to input video stream.

This filter operates in RGB colorspace.

This filter accepts the following options:

hue Set the hue shift in degrees to apply. Default is 0. Allowed range is from -180 to 180.

saturation

Set the saturation shift. Default is 0. Allowed range is from -1 to 1.

intensity

Set the intensity shift. Default is 0. Allowed range is from -1 to 1.

colors

Set which primary and complementary colors are going to be adjusted. This options is set by providing one or multiple values. This can select multiple colors at once. By default all colors are selected.

r Adjust reds.

y Adjust yellows.

g Adjust greens.

c Adjust cyans.

b Adjust blues.

m Adjust magentas.

a Adjust all colors.

strength

Set strength of filtering. Allowed range is from 0 to 100. Default value is 1.

rw, gw, bw

Set weight for each RGB component. Allowed range is from 0 to 1. By default is set to 0.333, 0.334, 0.333. Those options are used in saturation and lightness processing.

lightness

Set preserving lightness, by default is disabled. Adjusting hues can change lightness from original RGB triplet, with this option enabled lightness is kept at same value.

hysteresis

Grow first stream into second stream by connecting components. This makes it possible to build more robust edge masks.

This filter accepts the following options:

planes

Set which planes will be processed as bitmap, unprocessed planes will be copied from first stream. By default value 0xf, all planes will be processed.

threshold

Set threshold which is used in filtering. If pixel component value is higher than this value filter algorithm for connecting components is activated. By default value is 0.

The "hysteresis" filter also supports the **framesync** options.

iccdetect

Detect the colorspace from an embedded ICC profile (if present), and update the frame's tags accordingly.

This filter accepts the following options:

force

If true, the frame's existing colorspace tags will always be overridden by values detected from an ICC profile. Otherwise, they will only be assigned if they contain "unknown". Enabled by default.

iccgen

Generate ICC profiles and attach them to frames.

This filter accepts the following options:

color primaries

color_trc

Configure the colorspace that the ICC profile will be generated for. The default value of "auto" infers the value from the input frame's metadata, defaulting to BT.709/sRGB as appropriate.

See the **setparams** filter for a list of possible values, but note that "unknown" are not valid values for this filter.

force

If true, an ICC profile will be generated even if it would overwrite an already existing ICC profile. Disabled by default.

identity

Obtain the identity score between two input videos.

This filter takes two input videos.

Both input videos must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained per component, average, min and max identity score is printed through the logging system.

The filter stores the calculated identity scores of each frame in frame metadata.

This filter also supports the **framesync** options.

In the below example the input file *main.mpg* being processed is compared with the reference file *ref.mpg*.

```
ffmpeg -i main.mpg -i ref.mpg -lavfi identity -f null -
```

idet

Detect video interlacing type.

This filter tries to detect if the input frames are interlaced, progressive, top or bottom field first. It will also try to detect fields that are repeated between adjacent frames (a sign of telecine).

Single frame detection considers only immediately adjacent frames when classifying each frame. Multiple frame detection incorporates the classification history of previous frames.

The filter will log these metadata values:

single.current_frame

Detected type of current frame using single-frame detection. One of: “tff” (top field first), “bff” (bottom field first), “progressive”, or “undetermined”

single.tff

Cumulative number of frames detected as top field first using single-frame detection.

multiple.tff

Cumulative number of frames detected as top field first using multiple-frame detection.

single.bff

Cumulative number of frames detected as bottom field first using single-frame detection.

multiple.current_frame

Detected type of current frame using multiple-frame detection. One of: “tff” (top field first), “bff” (bottom field first), “progressive”, or “undetermined”

multiple.bff

Cumulative number of frames detected as bottom field first using multiple-frame detection.

single.progressive

Cumulative number of frames detected as progressive using single-frame detection.

multiple.progressive

Cumulative number of frames detected as progressive using multiple-frame detection.

single.undetermined

Cumulative number of frames that could not be classified using single-frame detection.

multiple.undetermined

Cumulative number of frames that could not be classified using multiple-frame detection.

repeated.current_frame

Which field in the current frame is repeated from the last. One of “neither”, “top”, or “bottom”.

repeated.neither

Cumulative number of frames with no repeated field.

repeated.top

Cumulative number of frames with the top field repeated from the previous frame’s top field.

repeated.bottom

Cumulative number of frames with the bottom field repeated from the previous frame’s bottom field.

The filter accepts the following options:

intl_thres

Set interlacing threshold.

prog_thres

Set progressive threshold.

rep_thres

Threshold for repeated field detection.

half_life

Number of frames after which a given frame's contribution to the statistics is halved (i.e., it contributes only 0.5 to its classification). The default of 0 means that all frames seen are given full weight of 1.0 forever.

analyze_interlaced_flag

When this is not 0 then idet will use the specified number of frames to determine if the interlaced flag is accurate, it will not count undetermined frames. If the flag is found to be accurate it will be used without any further computations, if it is found to be inaccurate it will be cleared without any further computations. This allows inserting the idet filter as a low computational method to clean up the interlaced flag

il

Deinterleave or interleave fields.

This filter allows one to process interlaced images fields without deinterlacing them. Deinterleaving splits the input frame into 2 fields (so called half pictures). Odd lines are moved to the top half of the output image, even lines to the bottom half. You can process (filter) them independently and then re-interleave them.

The filter accepts the following options:

luma_mode, l**chroma_mode, c****alpha_mode, a**

Available values for *luma_mode*, *chroma_mode* and *alpha_mode* are:

none

Do nothing.

deinterleave, d

Deinterleave fields, placing one above the other.

interleave, i

Interleave fields. Reverse the effect of deinterleaving.

Default value is "none".

luma_swap, ls**chroma_swap, cs****alpha_swap, as**

Swap luma/chroma/alpha fields. Exchange even & odd lines. Default value is 0.

Commands

This filter supports the all above options as **commands**.

inflate

Apply inflate effect to the video.

This filter replaces the pixel by the local(3x3) average by taking into account only values higher than the pixel.

It accepts the following options:

threshold0

threshold1

threshold2

threshold3

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

Commands

This filter supports the all above options as **commands**.

interlace

Simple interlacing filter from progressive contents. This interleaves upper (or lower) lines from odd frames with lower (or upper) lines from even frames, halving the frame rate and preserving image height.

Original Frame 'j'	Original Frame 'j+1'	New Frame (tff)
=====	=====	=====
Line 0 ----->		Frame 'j' Line 0
Line 1	Line 1 ---->	Frame 'j+1' Line 1
Line 2 ----->		Frame 'j' Line 2
Line 3	Line 3 ---->	Frame 'j+1' Line 3
...

New Frame + 1 will be generated by Frame 'j+2' and Frame 'j+3' and so on

It accepts the following optional parameters:

scan

This determines whether the interlaced frame is taken from the even (tff - default) or odd (bff) lines of the progressive frame.

lowpass

Vertical lowpass filter to avoid twitter interlacing and reduce moire patterns.

0, off

Disable vertical lowpass filter

1, linear

Enable linear filter (default)

2, complex

Enable complex filter. This will slightly less reduce twitter and moire but better retain detail and subjective sharpness impression.

kerndeint

Deinterlace input video by applying Donald Graft's adaptive kernel deinterling. Work on interlaced parts of a video to produce progressive frames.

The description of the accepted parameters follows.

thresh

Set the threshold which affects the filter's tolerance when determining if a pixel line must be processed. It must be an integer in the range [0,255] and defaults to 10. A value of 0 will result in applying the process on every pixels.

map Paint pixels exceeding the threshold value to white if set to 1. Default is 0.

order

Set the fields order. Swap fields if set to 1, leave fields alone if 0. Default is 0.

sharp

Enable additional sharpening if set to 1. Default is 0.

twoway

Enable twoway sharpening if set to 1. Default is 0.

Examples

- ⊕ Apply default values:

kerndeint=thresh=10:map=0:order=0:sharp=0:twoway=0

- ⊕ Enable additional sharpening:

kerndeint=sharp=1

- ⊕ Paint processed pixels in white:

kerndeint=map=1

kirsch

Apply kirsch operator to input video stream.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

scale

Set value which will be multiplied with filtered result.

delta

Set value which will be added to filtered result.

Commands

This filter supports the all above options as **commands**.

lagfun

Slowly update darker pixels.

This filter makes short flashes of light appear longer. This filter accepts the following options:

decay

Set factor for decaying. Default is .95. Allowed range is from 0 to 1.

planes

Set which planes to filter. Default is all. Allowed range is from 0 to 15.

Commands

This filter supports the all above options as **commands**.

lenscorrection

Correct radial lens distortion

This filter can be used to correct for radial distortion as can result from the use of wide angle lenses, and thereby re-rectify the image. To find the right parameters one can use tools available for example as part of opencv or simply trial-and-error. To use opencv use the calibration sample (under samples/cpp) from the opencv sources and extract the k1 and k2 coefficients from the resulting matrix.

Note that effectively the same filter is available in the open-source tools Krita and Digikam from the KDE project.

In contrast to the **vignette** filter, which can also be used to compensate lens errors, this filter corrects the distortion of the image, whereas **vignette** corrects the brightness distribution, so you may want to use both filters together in certain cases, though you will have to take care of ordering, i.e. whether vignetting should be applied before or after lens correction.

Options

The filter accepts the following options:

- cx** Relative x-coordinate of the focal point of the image, and thereby the center of the distortion. This value has a range [0,1] and is expressed as fractions of the image width. Default is 0.5.
- cy** Relative y-coordinate of the focal point of the image, and thereby the center of the distortion. This value has a range [0,1] and is expressed as fractions of the image height. Default is 0.5.
- k1** Coefficient of the quadratic correction term. This value has a range [-1,1]. 0 means no correction. Default is 0.
- k2** Coefficient of the double quadratic correction term. This value has a range [-1,1]. 0 means no correction. Default is 0.
- i** Set interpolation type. Can be "nearest" or "bilinear". Default is "nearest".

fc Specify the color of the unmapped pixels. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. Default color is "black@0".

The formula that generates the correction is:

$$r_{src} = r_{tgt} * (1 + k1 * (r_{tgt} / r_0)^2 + k2 * (r_{tgt} / r_0)^4)$$

where r_0 is halve of the image diagonal and r_{src} and r_{tgt} are the distances from the focal point in the source and target images, respectively.

Commands

This filter supports the all above options as **commands**.

lensfun

Apply lens correction via the lensfun library (<<http://lensfun.sourceforge.net/>>).

The "lensfun" filter requires the camera make, camera model, and lens model to apply the lens correction. The filter will load the lensfun database and query it to find the corresponding camera and lens entries in the database. As long as these entries can be found with the given options, the filter can perform corrections on frames. Note that incomplete strings will result in the filter choosing the best match with the given options, and the filter will output the chosen camera and lens models (logged with level "info"). You must provide the make, camera model, and lens model as they are required.

To obtain a list of available makes and models, leave out one or both of "make" and "model" options. The filter will send the full list to the log with level "INFO". The first column is the make and the second column is the model. To obtain a list of available lenses, set any values for make and model and leave out the "lens_model" option. The filter will send the full list of lenses in the log with level "INFO". The ffmpeg tool will exit after the list is printed.

The filter accepts the following options:

make

The make of the camera (for example, "Canon"). This option is required.

model

The model of the camera (for example, "Canon EOS 100D"). This option is required.

lens_model

The model of the lens (for example, "Canon EF-S 18-55mm f/3.5-5.6 IS STM"). This option is

required.

db_path

The full path to the lens database folder. If not set, the filter will attempt to load the database from the install path when the library was built. Default is unset.

mode

The type of correction to apply. The following values are valid options:

vignetting

Enables fixing lens vignetting.

geometry

Enables fixing lens geometry. This is the default.

subpixel

Enables fixing chromatic aberrations.

vig_geo

Enables fixing lens vignetting and lens geometry.

vig_subpixel

Enables fixing lens vignetting and chromatic aberrations.

distortion

Enables fixing both lens geometry and chromatic aberrations.

all Enables all possible corrections.

focal_length

The focal length of the image/video (zoom; expected constant for video). For example, a 18--55mm lens has focal length range of [18--55], so a value in that range should be chosen when using that lens. Default 18.

aperture

The aperture of the image/video (expected constant for video). Note that aperture is only used for vignetting correction. Default 3.5.

focus_distance

The focus distance of the image/video (expected constant for video). Note that focus distance is

only used for vignetting and only slightly affects the vignetting correction process. If unknown, leave it at the default value (which is 1000).

scale

The scale factor which is applied after transformation. After correction the video is no longer necessarily rectangular. This parameter controls how much of the resulting image is visible. The value 0 means that a value will be chosen automatically such that there is little or no unmapped area in the output image. 1.0 means that no additional scaling is done. Lower values may result in more of the corrected image being visible, while higher values may avoid unmapped areas in the output.

target_geometry

The target geometry of the output image/video. The following values are valid options:

rectilinear (default)

fisheye

panoramic

equirectangular

fisheye_orthographic

fisheye_stereographic

fisheye_equisolid

fisheye_thoby

reverse

Apply the reverse of image correction (instead of correcting distortion, apply it).

interpolation

The type of interpolation used when correcting distortion. The following values are valid options:

nearest

linear (default)

lanczos

Examples

- ⊕ Apply lens correction with make "Canon", camera model "Canon EOS 100D", and lens model "Canon EF-S 18-55mm f/3.5-5.6 IS STM" with focal length of "18" and aperture of "8.0".

```
ffmpeg -i input.mov -vf lensfun=make=Canon:model="Canon EOS 100D":lens_model="Canon EF-S 18-55mm f/3.5-5.6 IS STM":focal_length=18:aperture=8.0
```

- ⊕ Apply the same as before, but only for the first 5 seconds of video.

```
ffmpeg -i input.mov -vf lensfun=make=Canon:model="Canon EOS 100D":lens_model="Canon EF-S 18-55n
```

libplacebo

Flexible GPU-accelerated processing filter based on libplacebo (<<https://code.videolan.org/videolan/libplacebo>>). Note that this filter currently only accepts Vulkan input frames.

Options

The options for this filter are divided into the following sections:

Output mode

These options control the overall output mode. By default, libplacebo will try to preserve the source colorimetry and size as best as it can, but it will apply any embedded film grain, dolby vision metadata or anamorphic SAR present in source frames.

w

h Set the output video dimension expression. Default value is the input dimension.

Allows for the same expressions as the **scale** filter.

format

Set the output format override. If unset (the default), frames will be output in the same format as the respective input frames. Otherwise, format conversion will be performed.

force_original_aspect_ratio

force_divisible_by

Work the same as the identical **scale** filter options.

normalize_sar

If enabled, output frames will always have a pixel aspect ratio of 1:1. This will introduce padding/cropping as necessary. If disabled (the default), any aspect ratio mismatches, including those from e.g. anamorphic video sources, are forwarded to the output pixel aspect ratio.

pad_crop_ratio

Specifies a ratio (between 0.0 and 1.0) between padding and cropping when the input aspect ratio does not match the output aspect ratio and **normalize_sar** is in effect. The default of 0.0 always pads the content with black borders, while a value of 1.0 always crops off parts of the content. Intermediate values are possible, leading to a mix of the two approaches.

colorspace**color primaries****color_trc****range**

Configure the colorspace that output frames will be delivered in. The default value of "auto" outputs frames in the same format as the input frames, leading to no change. For any other value, conversion will be performed.

See the **setparams** filter for a list of possible values.

apply_filmgrain

Apply film grain (e.g. AV1 or H.274) if present in source frames, and strip it from the output. Enabled by default.

apply_dolbyvision

Apply Dolby Vision RPU metadata if present in source frames, and strip it from the output. Enabled by default. Note that Dolby Vision will always output BT.2020+PQ, overriding the usual input frame metadata. These will also be picked as the values of "auto" for the respective frame output options.

Scaling

The options in this section control how libplacebo performs upscaling and (if necessary) downscaling. Note that libplacebo will always internally operate on 4:4:4 content, so any sub-sampled chroma formats such as "yuv420p" will necessarily be upsampled and downsampled as part of the rendering process. That means scaling might be in effect even if the source and destination resolution are the same.

upscaler**downscaler**

Configure the filter kernel used for upscaling and downscaling. The respective defaults are "spline36" and "mitchell". For a full list of possible values, pass "help" to these options. The most important values are:

none

Forces the use of built-in GPU texture sampling (typically bilinear). Extremely fast but poor quality, especially when downscaling.

bilinear

Bilinear interpolation. Can generally be done for free on GPUs, except when doing so would

lead to aliasing. Fast and low quality.

nearest

Nearest-neighbour interpolation. Sharp but highly aliasing.

oversample

Algorithm that looks visually similar to nearest-neighbour interpolation but tries to preserve pixel aspect ratio. Good for pixel art, since it results in minimal distortion of the artistic appearance.

lanczos

Standard sinc-sinc interpolation kernel.

spline36

Cubic spline approximation of lanczos. No difference in performance, but has very slightly less ringing.

ewa_lanczos

Elliptically weighted average version of lanczos, based on a jinc-sinc kernel. This is also popularly referred to as just "Jinc scaling". Slow but very high quality.

gaussian

Gaussian kernel. Has certain ideal mathematical properties, but subjectively very blurry.

mitchell

Cubic BC spline with parameters recommended by Mitchell and Netravali. Very little ringing.

lut_entries

Configures the size of scaler LUTs, ranging from 1 to 256. The default of 0 will pick libplacebo's internal default, typically 64.

antirringing

Enables anti-ringing (for non-EWA filters). The value (between 0.0 and 1.0) configures the strength of the anti-ringing algorithm. May increase aliasing if set too high. Disabled by default.

sigmoid

Enable sigmoidal compression during upscaling. Reduces ringing slightly. Enabled by default.

Debanding

Libplacebo comes with a built-in debanding filter that is good at counteracting many common sources of banding and blocking. Turning this on is highly recommended whenever quality is desired.

deband

Enable (fast) debanding algorithm. Disabled by default.

deband_iterations

Number of deband iterations of the debanding algorithm. Each iteration is performed with progressively increased radius (and diminished threshold). Recommended values are in the range 1 to 4. Defaults to 1.

deband_threshold

Debanding filter strength. Higher numbers lead to more aggressive debanding. Defaults to 4.0.

deband_radius

Debanding filter radius. A higher radius is better for slow gradients, while a lower radius is better for steep gradients. Defaults to 16.0.

deband_grain

Amount of extra output grain to add. Helps hide imperfections. Defaults to 6.0.

Color adjustment

A collection of subjective color controls. Not very rigorous, so the exact effect will vary somewhat depending on the input primaries and colorspace.

brightness

Brightness boost, between "-1.0" and 1.0. Defaults to 0.0.

contrast

Contrast gain, between 0.0 and 16.0. Defaults to 1.0.

saturation

Saturation gain, between 0.0 and 16.0. Defaults to 1.0.

hue Hue shift in radians, between "-3.14" and 3.14. Defaults to 0.0. This will rotate the UV subvector, defaulting to BT.709 coefficients for RGB inputs.

gamma

Gamma adjustment, between 0.0 and 16.0. Defaults to 1.0.

cones

Cone model to use for color blindness simulation. Accepts any combination of "l", "m" and "s". Here are some examples:

m Deuteranomaly / deuteranopia (affecting 3%-4% of the population)

l Protanomaly / protanopia (affecting 1%-2% of the population)

l+m Monochromacy (very rare)

l+m+s

Achromatopsy (complete loss of daytime vision, extremely rare)

cone-strength

Gain factor for the cones specified by "cones", between 0.0 and 10.0. A value of 1.0 results in no change to color vision. A value of 0.0 (the default) simulates complete loss of those cones. Values above 1.0 result in exaggerating the differences between cones, which may help compensate for reduced color vision.

Peak detection

To help deal with sources that only have static HDR10 metadata (or no tagging whatsoever), libplacebo uses its own internal frame analysis compute shader to analyze source frames and adapt the tone mapping function in realtime. If this is too slow, or if exactly reproducible frame-perfect results are needed, it's recommended to turn this feature off.

peak_detect

Enable HDR peak detection. Ignores static MaxCLL/MaxFALL values in favor of dynamic detection from the input. Note that the detected values do not get written back to the output frames, they merely guide the internal tone mapping process. Enabled by default.

smoothing_period

Peak detection smoothing period, between 0.0 and 1000.0. Higher values result in peak detection becoming less responsive to changes in the input. Defaults to 100.0.

minimum_peak

Lower bound on the detected peak (relative to SDR white), between 0.0 and 100.0. Defaults to 1.0.

scene_threshold_low

scene_threshold_high

Lower and upper thresholds for scene change detection. Expressed in a logarithmic scale between 0.0 and 100.0. Default to 5.5 and 10.0, respectively. Setting either to a negative value disables this functionality.

overshoot

Peak smoothing overshoot margin, between 0.0 and 1.0. Provides a safety margin to prevent clipping as a result of peak smoothing. Defaults to 0.05, corresponding to a margin of 5%.

Tone mapping

The options in this section control how libplacebo performs tone-mapping and gamut-mapping when dealing with mismatches between wide-gamut or HDR content. In general, libplacebo relies on accurate source tagging and mastering display gamut information to produce the best results.

intent

Rendering intent to use when adapting between different primary color gamuts (after tone-mapping).

perceptual

Perceptual gamut mapping. Currently equivalent to relative colorimetric.

relative

Relative colorimetric. This is the default.

absolute

Absolute colorimetric.

saturation

Saturation mapping. Forcibly stretches the source gamut to the target gamut.

gamut_mode

How to handle out-of-gamut colors that can occur as a result of colorimetric gamut mapping.

clip Do nothing, simply clip out-of-range colors to the RGB volume. This is the default.

warn

Highlight out-of-gamut pixels (by coloring them pink).

darken

Linearly reduces content brightness to preserve saturated details, followed by clipping the remaining out-of-gamut colors. As the name implies, this makes everything darker, but provides a good balance between preserving details and colors.

desaturate

Hard-desaturates out-of-gamut colors towards white, while preserving the luminance. Has a tendency to shift colors.

tonemapping

Tone-mapping algorithm to use. Available values are:

auto Automatic selection based on internal heuristics. This is the default.

clip Performs no tone-mapping, just clips out-of-range colors. Retains perfect color accuracy for in-range colors but completely destroys out-of-range information. Does not perform any black point adaptation. Not configurable.

st2094-40

EETF from SMPTE ST 2094-40 Annex B, which applies the Bezier curves from HDR10+ dynamic metadata based on Bezier curves to perform tone-mapping. The OOTF used is adjusted based on the ratio between the targeted and actual display peak luminances.

st2094-10

EETF from SMPTE ST 2094-10 Annex B.2, which takes into account the input signal average luminance in addition to the maximum/minimum. The configurable contrast parameter influences the slope of the linear output segment, defaulting to 1.0 for no increase/decrease in contrast. Note that this does not currently include the subjective gain/offset/gamma controls defined in Annex B.3.

bt.2390

EETF from the ITU-R Report BT.2390, a hermite spline roll-off with linear segment. The knee point offset is configurable. Note that this parameter defaults to 1.0, rather than the value of 0.5 from the ITU-R spec.

bt.2446a

EETF from ITU-R Report BT.2446, method A. Designed for well-mastered HDR sources. Can be used for both forward and inverse tone mapping. Not configurable.

spline

Simple spline consisting of two polynomials, joined by a single pivot point. The parameter

gives the pivot point (in PQ space), defaulting to 0.30. Can be used for both forward and inverse tone mapping.

reinhard

Simple non-linear, global tone mapping algorithm. The parameter specifies the local contrast coefficient at the display peak. Essentially, a parameter of 0.5 implies that the reference white will be about half as bright as when clipping. Defaults to 0.5, which results in the simplest formulation of this function.

mobius

Generalization of the reinhard tone mapping algorithm to support an additional linear slope near black. The tone mapping parameter indicates the trade-off between the linear section and the non-linear section. Essentially, for a given parameter x , every color value below x will be mapped linearly, while higher values get non-linearly tone-mapped. Values near 1.0 make this curve behave like "clip", while values near 0.0 make this curve behave like "reinhard". The default value is 0.3, which provides a good balance between colorimetric accuracy and preserving out-of-gamut details.

hable

Piece-wise, filmic tone-mapping algorithm developed by John Hable for use in Uncharted 2, inspired by a similar tone-mapping algorithm used by Kodak. Popularized by its use in video games with HDR rendering. Preserves both dark and bright details very well, but comes with the drawback of changing the average brightness quite significantly. This is sort of similar to "reinhard" with parameter 0.24.

gamma

Fits a gamma (power) function to transfer between the source and target color spaces, effectively resulting in a perceptual hard-knee joining two roughly linear sections. This preserves details at all scales fairly accurately, but can result in an image with a muted or dull appearance. The parameter is used as the cutoff point, defaulting to 0.5.

linear

Linearly stretches the input range to the output range, in PQ space. This will preserve all details accurately, but results in a significantly different average brightness. Can be used for inverse tone-mapping in addition to regular tone-mapping. The parameter can be used as an additional linear gain coefficient (defaulting to 1.0).

tonemapping_param

For tunable tone mapping functions, this parameter can be used to fine-tune the curve behavior. Refer to the documentation of "tonemapping". The default value of 0.0 is replaced by the curve's

preferred default setting.

tonemapping_mode

This option determines how the tone mapping function specified by "tonemapping" is applied to the colors in a scene. Possible values are:

auto Automatic selection based on internal heuristics. This is the default.

rgb Apply the function per-channel in the RGB colorspace. Per-channel tone-mapping in RGB. Guarantees no clipping and heavily desaturates the output, but distorts the colors quite significantly. Very similar to the "Hollywood" look and feel.

max Tone-mapping is performed on the brightest component found in the signal. Good at preserving details in highlights, but has a tendency to crush blacks.

hybrid

Tone-map per-channel for highlights and linearly (luma-based) for midtones/shadows, based on a fixed gamma 2.4 coefficient curve.

luma

Tone-map linearly on the luma component (CIE Y), and adjust (desaturate) the chromaticities to compensate using a simple constant factor. This is essentially the mode used in ITU-R BT.2446 method A.

inverse_tonemapping

If enabled, this filter will also attempt stretching SDR signals to fill HDR output color volumes. Disabled by default.

tonemapping_crosstalk

Extra tone-mapping crosstalk factor, between 0.0 and 0.3. This can help reduce issues tone-mapping certain bright spectral colors. Defaults to 0.04.

tonemapping_lut_size

Size of the tone-mapping LUT, between 2 and 1024. Defaults to 256. Note that this figure is squared when combined with "peak_detect".

Dithering

By default, libplacebo will dither whenever necessary, which includes rendering to any integer format below 16-bit precision. It's recommended to always leave this on, since not doing so may result in

visible banding in the output, even if the "debanding" filter is enabled. If maximum performance is needed, use "ordered_fixed" instead of disabling dithering.

dithering

Dithering method to use. Accepts the following values:

none

Disables dithering completely. May result in visible banding.

blue Dither with pseudo-blue noise. This is the default.

ordered

Tunable ordered dither pattern.

ordered_fixed

Faster ordered dither with a fixed size of 6. Texture-less.

white

Dither with white noise. Texture-less.

dither_lut_size

Dither LUT size, as log base2 between 1 and 8. Defaults to 6, corresponding to a LUT size of "64x64".

dither_temporal

Enables temporal dithering. Disabled by default.

Custom shaders

libplacebo supports a number of custom shaders based on the mpv .hook GLSL syntax. A collection of such shaders can be found here: <<https://github.com/mpv-player/mpv/wiki/User-Scripts#user-shaders>>

A full description of the mpv shader format is beyond the scope of this section, but a summary can be found here: <<https://mpv.io/manual/master/#options-gsl-shader>>

custom_shader_path

Specifies a path to a custom shader file to load at runtime.

custom_shader_bin

Specifies a complete custom shader as a raw string.

Debugging / performance

All of the options in this section default off. They may be of assistance when attempting to squeeze the maximum performance at the cost of quality.

skip_aa

Disable anti-aliasing when downscaling.

polar_cutoff

Truncate polar (EWA) scaler kernels below this absolute magnitude, between 0.0 and 1.0.

disable_linear

Disable linear light scaling.

disable_builtin

Disable built-in GPU sampling (forces LUT).

disable_fbos

Forcibly disable FBOs, resulting in loss of almost all functionality, but offering the maximum possible speed.

Commands

This filter supports almost all of the above options as **commands**.

Examples

- ⊕ Complete example for how to initialize the Vulkan device, upload frames to the GPU, perform filter conversion to yuv420p, and download frames back to the CPU for output. Note that in specific cases you can get around the need to perform format conversion by specifying the correct "format" filter option corresponding to the input frames.

```
ffmpeg -i $INPUT -init_hw_device vulkan -vf hwupload,libplacebo=format=yuv420p,hwdownload,format=y
```

- ⊕ Tone-map input to standard gamut BT.709 output:

```
libplacebo=colorspace=bt709:color_primaries=bt709:color_trc=bt709:range=tv
```

- ⊕ Rescale input to fit into standard 1080p, with high quality scaling:

```
libplacebo=w=1920:h=1080:force_original_aspect_ratio=decrease:normalize_sar=true:upscaler=ewa_lanczo
```

- ⊕ Convert input to standard sRGB JPEG:

```
libplacebo=format=yuv420p:colorspace=bt470bg:color_primaries=bt709:color_trc=iec61966-2-1:range=pc
```

- ⊕ Use higher quality debanding settings:

```
libplacebo=deband=true:deband_iterations=3:deband_radius=8:deband_threshold=6
```

- ⊕ Run this filter on the CPU, on systems with Mesa installed (and with the most expensive options disabled):

```
ffmpeg ... -init_hw_device vulkan:llvmpipe ... -vf libplacebo=upscaler=none:downscaler=none:peak_detect=
```

- ⊕ Suppress CPU-based AV1/H.274 film grain application in the decoder, in favor of doing it with this filter. Note that this is only a gain if the frames are either already on the GPU, or if you're using libplacebo for other purposes, since otherwise the VRAM roundtrip will more than offset any expected speedup.

```
ffmpeg -export_side_data +film_grain ... -vf libplacebo=apply_filmgrain=true
```

libvmaf

Calculate the VMAF (Video Multi-Method Assessment Fusion) score for a reference/distorted pair of input videos.

The first input is the distorted video, and the second input is the reference video.

The obtained VMAF score is printed through the logging system.

It requires Netflix's vmaf library (libvmaf) as a pre-requisite. After installing the library it can be enabled using: `./configure --enable-libvmaf`.

The filter has following options:

model

A '|' delimited list of vmaf models. Each model can be configured with a number of parameters.

Default value: `"version=vmaf_v0.6.1"`

model_path

Deprecated, use `model='path=...'`.

enable_transform

Deprecated, use `model='enable_transform=true'`.

phone_model

Deprecated, use `model='enable_transform=true'`.

enable_conf_interval

Deprecated, use `model='enable_conf_interval=true'`.

feature

A '|' delimited list of features. Each feature can be configured with a number of parameters.

psnr Deprecated, use `feature='name=psnr'`.

ssim

Deprecated, use `feature='name=ssim'`.

ms_ssim

Deprecated, use `feature='name=ms_ssim'`.

log_path

Set the file path to be used to store log files.

log_fmt

Set the format of the log file (xml, json, csv, or sub).

n_threads

Set number of threads to be used when initializing libvmaf. Default value: 0, no threads.

n_subsample

Set frame subsampling interval to be used.

This filter also supports the **framesync** options.

Examples

- ⊕ In the examples below, a distorted video *distorted.mpg* is compared with a reference file *reference.mpg*.

- ⊕ Basic usage:

```
ffmpeg -i distorted.mpg -i reference.mpg -lavfi libvmaf=log_path=output.xml -f null -
```

- ⊕ Example with multiple models:

```
ffmpeg -i distorted.mpg -i reference.mpg -lavfi libvmaf='model=version=vmaf_v0.6.1\\:name=vmaf|version=
```

- ⊕ Example with multiple additional features:

```
ffmpeg -i distorted.mpg -i reference.mpg -lavfi libvmaf='feature=name=psnr|name=ciede' -f null -
```

- ⊕ Example with options and different containers:

```
ffmpeg -i distorted.mpg -i reference.mkv -lavfi "[0:v]settb=AVTB,setpts=PTS-STARTPTS[main];[1:v]settb=
```

limitdiff

Apply limited difference filter using second and optionally third video stream.

The filter accepts the following options:

threshold

Set the threshold to use when allowing certain differences between video streams. Any absolute difference value lower or exact than this threshold will pick pixel components from first video stream.

elasticity

Set the elasticity of soft thresholding when processing video streams. This value multiplied with first one sets second threshold. Any absolute difference value greater or exact than second threshold will pick pixel components from second video stream. For values between those two threshold linear interpolation between first and second video stream will be used.

reference

Enable the reference (third) video stream processing. By default is disabled. If set, this video stream will be used for calculating absolute difference with first video stream.

planes

Specify which planes will be processed. Defaults to all available.

Commands

This filter supports the all above options as **commands** except option **reference**.

limiter

Limits the pixel components values to the specified range [min, max].

The filter accepts the following options:

min Lower bound. Defaults to the lowest allowed value for the input.

max Upper bound. Defaults to the highest allowed value for the input.

planes

Specify which planes will be processed. Defaults to all available.

Commands

This filter supports the all above options as **commands**.

loop

Loop video frames.

The filter accepts the following options:

loop

Set the number of loops. Setting this value to -1 will result in infinite loops. Default is 0.

size Set maximal size in number of frames. Default is 0.

start Set first frame of loop. Default is 0.

Examples

⊕ Loop single first frame infinitely:

```
loop=loop=-1:size=1:start=0
```

⊕ Loop single first frame 10 times:

```
loop=loop=10:size=1:start=0
```


- ⊕ Loop 10 first frames 5 times:

```
loop=loop=5:size=10:start=0
```

lut1d

Apply a 1D LUT to an input video.

The filter accepts the following options:

file Set the 1D LUT file name.

Currently supported formats:

cube

Iridas

csp cineSpace

interp

Select interpolation mode.

Available values are:

nearest

Use values from the nearest defined point.

linear

Interpolate values using the linear interpolation.

cosine

Interpolate values using the cosine interpolation.

cubic

Interpolate values using the cubic interpolation.

spline

Interpolate values using the spline interpolation.

Commands

This filter supports the all above options as **commands**.

lut3d

Apply a 3D LUT to an input video.

The filter accepts the following options:

file Set the 3D LUT file name.

Currently supported formats:

3dl AfterEffects

cube

Iridas

dat DaVinci

m3d

Pandora

csp cineSpace

interp

Select interpolation mode.

Available values are:

nearest

Use values from the nearest defined point.

trilinear

Interpolate values using the 8 points defining a cube.

tetrahedral

Interpolate values using a tetrahedron.

pyramid

Interpolate values using a pyramid.

prism

Interpolate values using a prism.

Commands

This filter supports the "interp" option as **commands**.

lumakey

Turn certain luma values into transparency.

The filter accepts the following options:

threshold

Set the luma which will be used as base for transparency. Default value is 0.

tolerance

Set the range of luma values to be keyed out. Default value is 0.01.

softness

Set the range of softness. Default value is 0. Use this to control gradual transition from zero to full transparency.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

lut, lutrgb, lutyuv

Compute a look-up table for binding each pixel component input value to an output value, and apply it to the input video.

lutyuv applies a lookup table to a YUV input video, *lutrgb* to an RGB input video.

These filters accept the following parameters:

c0 set first pixel component expression

c1 set second pixel component expression

- c2** set third pixel component expression
- c3** set fourth pixel component expression, corresponds to the alpha component
- r** set red component expression
- g** set green component expression
- b** set blue component expression
- a** alpha component expression
- y** set Y/luminance component expression
- u** set U/Cb component expression
- v** set V/Cr component expression

Each of them specifies the expression to use for computing the lookup table for the corresponding pixel component values.

The exact component associated to each of the *c** options depends on the format in input.

The *lut* filter requires either YUV or RGB pixel formats in input, *lutrgb* requires RGB pixel formats in input, and *lutyuv* requires YUV.

The expressions can contain the following constants and functions:

w

h The input width and height.

val The input value for the pixel component.

clipval

The input value, clipped to the *minval-maxval* range.

maxval

The maximum value for the pixel component.

minval

The minimum value for the pixel component.

negval

The negated value for the pixel component value, clipped to the *minval-maxval* range; it corresponds to the expression "maxval-clipval+minval".

clip(val)

The computed value in *val*, clipped to the *minval-maxval* range.

gammaval(gamma)

The computed gamma correction value of the pixel component value, clipped to the *minval-maxval* range. It corresponds to the expression "pow((clipval-minval)/(maxval-minval),*gamma*)*(maxval-minval)+minval"

All expressions default to "clipval".

Commands

This filter supports same **commands** as options.

Examples

- ⊕ Negate input video:

```
lutrgb="r=maxval+minval-val:g=maxval+minval-val:b=maxval+minval-val"
lutyuv="y=maxval+minval-val:u=maxval+minval-val:v=maxval+minval-val"
```

The above is the same as:

```
lutrgb="r=negval:g=negval:b=negval"
lutyuv="y=negval:u=negval:v=negval"
```

- ⊕ Negate luminance:

```
lutyuv=y=negval
```

- ⊕ Remove chroma components, turning the video into a graytone image:

```
lutyuv="u=128:v=128"
```

- ⊕ Apply a luma burning effect:

```
lutyuv="y=2*val"
```

- ⊕ Remove green and blue components:

```
lutrgb="g=0:b=0"
```

- ⊕ Set a constant alpha channel value on input:

```
format=rgba,lutrgb=a="maxval-minval/2"
```

- ⊕ Correct luminance gamma by a factor of 0.5:

```
lutyuv=y=gammaval(0.5)
```

- ⊕ Discard least significant bits of luma:

```
lutyuv=y='bitand(val, 128+64+32)'
```

- ⊕ Technicolor like effect:

```
lutyuv=u='(val-maxval/2)*2+maxval/2':v='(val-maxval/2)*2+maxval/2'
```

lut2, tlut2

The "lut2" filter takes two input streams and outputs one stream.

The "tlut2" (time lut2) filter takes two consecutive frames from one single stream.

This filter accepts the following parameters:

- c0** set first pixel component expression
- c1** set second pixel component expression
- c2** set third pixel component expression
- c3** set fourth pixel component expression, corresponds to the alpha component
- d** set output bit depth, only available for "lut2" filter. By default is 0, which means bit depth is

automatically picked from first input format.

The "lut2" filter also supports the **framesync** options.

Each of them specifies the expression to use for computing the lookup table for the corresponding pixel component values.

The exact component associated to each of the *c** options depends on the format in inputs.

The expressions can contain the following constants:

w

h The input width and height.

x The first input value for the pixel component.

y The second input value for the pixel component.

bdx The first input video bit depth.

bdy The second input video bit depth.

All expressions default to "x".

Commands

This filter supports the all above options as **commands** except option "d".

Examples

- ⊕ Highlight differences between two RGB video streams:

```
lut2='ifnot(x-y,0,pow(2,bdx)-1):ifnot(x-y,0,pow(2,bdx)-1):ifnot(x-y,0,pow(2,bdx)-1)'
```

- ⊕ Highlight differences between two YUV video streams:

```
lut2='ifnot(x-y,0,pow(2,bdx)-1):ifnot(x-y,pow(2,bdx)-1):ifnot(x-y,pow(2,bdx)-1,pow(2,bdx)-1)'
```

- ⊕ Show max difference between two video streams:

```
lut2='if(lt(x,y),0,if(gt(x,y),pow(2,bdx)-1,pow(2,bdx-1))):if(lt(x,y),0,if(gt(x,y),pow(2,bdx)-1,pow(2,bdx-1))):i
```

maskedclamp

Clamp the first input stream with the second input and third input stream.

Returns the value of first stream to be between second input stream - "undershoot" and third input stream + "overshoot".

This filter accepts the following options:

undershoot

Default value is 0.

overshoot

Default value is 0.

planes

Set which planes will be processed as bitmap, unprocessed planes will be copied from first stream. By default value 0xf, all planes will be processed.

Commands

This filter supports the all above options as **commands**.

maskedmax

Merge the second and third input stream into output stream using absolute differences between second input stream and first input stream and absolute difference between third input stream and first input stream. The picked value will be from second input stream if second absolute difference is greater than first one or from third input stream otherwise.

This filter accepts the following options:

planes

Set which planes will be processed as bitmap, unprocessed planes will be copied from first stream. By default value 0xf, all planes will be processed.

Commands

This filter supports the all above options as **commands**.

maskedmerge

Merge the first input stream with the second input stream using per pixel weights in the third input stream.

A value of 0 in the third stream pixel component means that pixel component from first stream is returned unchanged, while maximum value (eg. 255 for 8-bit videos) means that pixel component from second stream is returned unchanged. Intermediate values define the amount of merging between both input stream's pixel components.

This filter accepts the following options:

planes

Set which planes will be processed as bitmap, unprocessed planes will be copied from first stream. By default value 0xf, all planes will be processed.

Commands

This filter supports the all above options as **commands**.

maskedmin

Merge the second and third input stream into output stream using absolute differences between second input stream and first input stream and absolute difference between third input stream and first input stream. The picked value will be from second input stream if second absolute difference is less than first one or from third input stream otherwise.

This filter accepts the following options:

planes

Set which planes will be processed as bitmap, unprocessed planes will be copied from first stream. By default value 0xf, all planes will be processed.

Commands

This filter supports the all above options as **commands**.

maskedthreshold

Pick pixels comparing absolute difference of two video streams with fixed threshold.

If absolute difference between pixel component of first and second video stream is equal or lower than user supplied threshold than pixel component from first video stream is picked, otherwise pixel

component from second video stream is picked.

This filter accepts the following options:

threshold

Set threshold used when picking pixels from absolute difference from two input video streams.

planes

Set which planes will be processed as bitmap, unprocessed planes will be copied from second stream. By default value 0xf, all planes will be processed.

mode

Set mode of filter operation. Can be "abs" or "diff". Default is "abs".

Commands

This filter supports the all above options as **commands**.

maskfun

Create mask from input video.

For example it is useful to create motion masks after "tblend" filter.

This filter accepts the following options:

low Set low threshold. Any pixel component lower or exact than this value will be set to 0.

high

Set high threshold. Any pixel component higher than this value will be set to max value allowed for current pixel format.

planes

Set planes to filter, by default all available planes are filtered.

fill Fill all frame pixels with this value.

sum Set max average pixel value for frame. If sum of all pixel components is higher than this average, output frame will be completely filled with value set by *fill* option. Typically useful for scene changes when used in combination with "tblend" filter.

Commands

This filter supports the all above options as **commands**.

mcdeint

Apply motion-compensation deinterlacing.

It needs one field per frame as input and must thus be used together with yadif=1/3 or equivalent.

This filter is only available in ffmpeg version 4.4 or earlier.

This filter accepts the following options:

mode

Set the deinterlacing mode.

It accepts one of the following values:

fast

medium

slow

use iterative motion estimation

extra_slow

like **slow**, but use multiple reference frames.

Default value is **fast**.

parity

Set the picture field parity assumed for the input video. It must be one of the following values:

0, tff

assume top field first

1, bff

assume bottom field first

Default value is **bff**.

qp Set per-block quantization parameter (QP) used by the internal encoder.

Higher values should result in a smoother motion vector field but less optimal individual vectors.
Default value is 1.

median

Pick median pixel from certain rectangle defined by radius.

This filter accepts the following options:

radius

Set horizontal radius size. Default value is 1. Allowed range is integer from 1 to 127.

planes

Set which planes to process. Default is 15, which is all available planes.

radiusV

Set vertical radius size. Default value is 0. Allowed range is integer from 0 to 127. If it is 0, value will be picked from horizontal "radius" option.

percentile

Set median percentile. Default value is 0.5. Default value of 0.5 will pick always median values, while 0 will pick minimum values, and 1 maximum values.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

mergeplanes

Merge color channel components from several video streams.

The filter accepts up to 4 input streams, and merge selected input planes to the output video.

This filter accepts the following options:

mapping

Set input to output plane mapping. Default is 0.

The mappings is specified as a bitmap. It should be specified as a hexadecimal number in the form

0xAa[Bb[Cc[Dd]]]. 'Aa' describes the mapping for the first plane of the output stream. 'A' sets the number of the input stream to use (from 0 to 3), and 'a' the plane number of the corresponding input to use (from 0 to 3). The rest of the mappings is similar, 'Bb' describes the mapping for the output stream second plane, 'Cc' describes the mapping for the output stream third plane and 'Dd' describes the mapping for the output stream fourth plane.

format

Set output pixel format. Default is "yuva444p".

map0s

map1s

map2s

map3s

Set input to output stream mapping for output Nth plane. Default is 0.

map0p

map1p

map2p

map3p

Set input to output plane mapping for output Nth plane. Default is 0.

Examples

- ⊕ Merge three gray video streams of same width and height into single video stream:

```
[a0][a1][a2]mergeplanes=0x001020: yuv444p
```

- ⊕ Merge 1st yuv444p stream and 2nd gray video stream into yuva444p video stream:

```
[a0][a1]mergeplanes=0x00010210: yuva444p
```

- ⊕ Swap Y and A plane in yuva444p stream:

```
format=yuva444p,mergeplanes=0x03010200: yuva444p
```

- ⊕ Swap U and V plane in yuv420p stream:

```
format=yuv420p,mergeplanes=0x000201: yuv420p
```

- ⊕ Cast a rgb24 clip to yuv444p:

```
format=rgb24,mergeplanes=0x000102:yuv444p
```

mestimate

Estimate and export motion vectors using block matching algorithms. Motion vectors are stored in frame side data to be used by other filters.

This filter accepts the following options:

method

Specify the motion estimation method. Accepts one of the following values:

esa Exhaustive search algorithm.

tss Three step search algorithm.

tdls Two dimensional logarithmic search algorithm.

ntss New three step search algorithm.

fss Four step search algorithm.

ds Diamond search algorithm.

hexbs

Hexagon-based search algorithm.

epzs

Enhanced predictive zonal search algorithm.

umh

Uneven multi-hexagon search algorithm.

Default value is **esa**.

mb_size

Macroblock size. Default 16.

search_param

Search parameter. Default 7.

midequalizer

Apply Midway Image Equalization effect using two video streams.

Midway Image Equalization adjusts a pair of images to have the same histogram, while maintaining their dynamics as much as possible. It's useful for e.g. matching exposures from a pair of stereo cameras.

This filter has two inputs and one output, which must be of same pixel format, but may be of different sizes. The output of filter is first input adjusted with midway histogram of both inputs.

This filter accepts the following option:

planes

Set which planes to process. Default is 15, which is all available planes.

minterpolate

Convert the video to specified frame rate using motion interpolation.

This filter accepts the following options:

fps Specify the output frame rate. This can be rational e.g. "60000/1001". Frames are dropped if *fps* is lower than source fps. Default 60.

mi_mode

Motion interpolation mode. Following values are accepted:

dup Duplicate previous or next frame for interpolating new ones.

blend

Blend source frames. Interpolated frame is mean of previous and next frames.

mci Motion compensated interpolation. Following options are effective when this mode is selected:

mc_mode

Motion compensation mode. Following values are accepted:

obmc

Overlapped block motion compensation.

aobmc

Adaptive overlapped block motion compensation. Window weighting coefficients are controlled adaptively according to the reliabilities of the neighboring motion vectors to reduce oversmoothing.

Default mode is **obmc**.

me_mode

Motion estimation mode. Following values are accepted:

bidir

Bidirectional motion estimation. Motion vectors are estimated for each source frame in both forward and backward directions.

bilat

Bilateral motion estimation. Motion vectors are estimated directly for interpolated frame.

Default mode is **bilat**.

me The algorithm to be used for motion estimation. Following values are accepted:

esa Exhaustive search algorithm.

tss Three step search algorithm.

tdls Two dimensional logarithmic search algorithm.

ntss New three step search algorithm.

fss Four step search algorithm.

ds Diamond search algorithm.

hexbs

Hexagon-based search algorithm.

epzs

Enhanced predictive zonal search algorithm.

umh

Uneven multi-hexagon search algorithm.

Default algorithm is **epzs**.

mb_size

Macroblock size. Default 16.

search_param

Motion estimation search parameter. Default 32.

vsbmc

Enable variable-size block motion compensation. Motion estimation is applied with smaller block sizes at object boundaries in order to make the them less blur. Default is 0 (disabled).

scd Scene change detection method. Scene change leads motion vectors to be in random direction. Scene change detection replace interpolated frames by duplicate ones. May not be needed for other modes. Following values are accepted:

none

Disable scene change detection.

fdiff

Frame difference. Corresponding pixel values are compared and if it satisfies *scd_threshold* scene change is detected.

Default method is **fdiff**.

scd_threshold

Scene change detection threshold. Default is 10..

mix

Mix several video input streams into one video stream.

A description of the accepted options follows.

inputs

The number of inputs. If unspecified, it defaults to 2.

weights

Specify weight of each input video stream as sequence. Each weight is separated by space. If number of weights is smaller than number of *frames* last specified weight will be used for all remaining unset weights.

scale

Specify scale, if it is set it will be multiplied with sum of each weight multiplied with pixel values to give final destination pixel value. By default *scale* is auto scaled to sum of weights.

planes

Set which planes to filter. Default is all. Allowed range is from 0 to 15.

duration

Specify how end of stream is determined.

longest

The duration of the longest input. (default)

shortest

The duration of the shortest input.

first The duration of the first input.

Commands

This filter supports the following commands:

weights**scale****planes**

Syntax is same as option with same name.

monochrome

Convert video to gray using custom color filter.

A description of the accepted options follows.

cb Set the chroma blue spot. Allowed range is from -1 to 1. Default value is 0.

cr Set the chroma red spot. Allowed range is from -1 to 1. Default value is 0.

size Set the color filter size. Allowed range is from .1 to 10. Default value is 1.

high

Set the highlights strength. Allowed range is from 0 to 1. Default value is 0.

Commands

This filter supports the all above options as **commands**.

morpho

This filter allows to apply main morphological grayscale transforms, erode and dilate with arbitrary structures set in second input stream.

Unlike naive implementation and much slower performance in **erosion** and **dilation** filters, when speed is critical "morpho" filter should be used instead.

A description of accepted options follows,

mode

Set morphological transform to apply, can be:

erode

dilate

open

close

gradient

tophat

blackhat

Default is "erode".

planes

Set planes to filter, by default all planes except alpha are filtered.

structure

Set which structure video frames will be processed from second input stream, can be *first* or *all*.

Default is *all*.

The "morpho" filter also supports the **framesync** options.

Commands

This filter supports same **commands** as options.

mpdecimate

Drop frames that do not differ greatly from the previous frame in order to reduce frame rate.

The main use of this filter is for very-low-bitrate encoding (e.g. streaming over dialup modem), but it could in theory be used for fixing movies that were inverse-telecined incorrectly.

A description of the accepted options follows.

max Set the maximum number of consecutive frames which can be dropped (if positive), or the minimum interval between dropped frames (if negative). If the value is 0, the frame is dropped disregarding the number of previous sequentially dropped frames.

Default value is 0.

hi

lo

frac Set the dropping threshold values.

Values for **hi** and **lo** are for 8x8 pixel blocks and represent actual pixel value differences, so a threshold of 64 corresponds to 1 unit of difference for each pixel, or the same spread out differently over the block.

A frame is a candidate for dropping if no 8x8 blocks differ by more than a threshold of **hi**, and if no more than **frac** blocks (1 meaning the whole image) differ by more than a threshold of **lo**.

Default value for **hi** is 64*12, default value for **lo** is 64*5, and default value for **frac** is 0.33.

msad

Obtain the MSAD (Mean Sum of Absolute Differences) between two input videos.

This filter takes two input videos.

Both input videos must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained per component, average, min and max MSAD is printed through the logging system.

The filter stores the calculated MSAD of each frame in frame metadata.

This filter also supports the **framesync** options.

In the below example the input file *main.mpg* being processed is compared with the reference file *ref.mpg*.

```
ffmpeg -i main.mpg -i ref.mpg -lavfi msad -f null -
```

multiply

Multiply first video stream pixels values with second video stream pixels values.

The filter accepts the following options:

scale

Set the scale applied to second video stream. By default is 1. Allowed range is from 0 to 9.

offset

Set the offset applied to second video stream. By default is 0.5. Allowed range is from "-1" to 1.

planes

Specify planes from input video stream that will be processed. By default all planes are processed.

Commands

This filter supports same **commands** as options.

negate

Negate (invert) the input video.

It accepts the following option:

components

Set components to negate.

Available values for components are:

y
u

v
a
r
g
b

negate_alpha

With value 1, it negates the alpha component, if present. Default value is 0.

Commands

This filter supports same **commands** as options.

nlmeans

Denoise frames using Non-Local Means algorithm.

Each pixel is adjusted by looking for other pixels with similar contexts. This context similarity is defined by comparing their surrounding patches of size **pxp**. Patches are searched in an area of **rxr** around the pixel.

Note that the research area defines centers for patches, which means some patches will be made of pixels outside that research area.

The filter accepts the following options.

s Set denoising strength. Default is 1.0. Must be in range [1.0, 30.0].

p Set patch size. Default is 7. Must be odd number in range [0, 99].

pc Same as **p** but for chroma planes.

The default value is 0 and means automatic.

r Set research size. Default is 15. Must be odd number in range [0, 99].

rc Same as **r** but for chroma planes.

The default value is 0 and means automatic.

nmedi

Deinterlace video using neural network edge directed interpolation.

This filter accepts the following options:

weights

Mandatory option, without binary file filter can not work. Currently file can be found here:
https://github.com/dubhater/vapoursynth-nnedi3/blob/master/src/nnedi3_weights.bin

deint

Set which frames to deinterlace, by default it is "all". Can be "all" or "interlaced".

field

Set mode of operation.

Can be one of the following:

- af** Use frame flags, both fields.
- a** Use frame flags, single field.
- t** Use top field only.
- b** Use bottom field only.
- tf** Use both fields, top first.
- bf** Use both fields, bottom first.

planes

Set which planes to process, by default filter process all frames.

nsize

Set size of local neighborhood around each pixel, used by the predictor neural network.

Can be one of the following:

- s8x6**
- s16x6**
- s32x6**
- s48x6**
- s8x4**
- s16x4**

s32x4

nns Set the number of neurons in predictor neural network. Can be one of the following:

n16

n32

n64

n128

n256

qual Controls the number of different neural network predictions that are blended together to compute the final output value. Can be "fast", default or "slow".

etype

Set which set of weights to use in the predictor. Can be one of the following:

a, abs

weights trained to minimize absolute error

s, mse

weights trained to minimize squared error

pscrn

Controls whether or not the prescreener neural network is used to decide which pixels should be processed by the predictor neural network and which can be handled by simple cubic interpolation. The prescreener is trained to know whether cubic interpolation will be sufficient for a pixel or whether it should be predicted by the predictor nn. The computational complexity of the prescreener nn is much less than that of the predictor nn. Since most pixels can be handled by cubic interpolation, using the prescreener generally results in much faster processing. The prescreener is pretty accurate, so the difference between using it and not using it is almost always unnoticeable.

Can be one of the following:

none

original

new

new2

new3

Default is "new".

Commands

This filter supports same **commands** as options, excluding *weights* option.

noformat

Force libavfilter not to use any of the specified pixel formats for the input to the next filter.

It accepts the following parameters:

pix_fmts

A '|' -separated list of pixel format names, such as `pix_fmts=yuv420p|monow|rgb24`".

Examples

- ⊕ Force libavfilter to use a format different from *yuv420p* for the input to the *vflip* filter:

```
noformat=pix_fmts=yuv420p,vflip
```

- ⊕ Convert the input video to any of the formats not contained in the list:

```
noformat=yuv420p|yuv444p|yuv410p
```

noise

Add noise on video input frame.

The filter accepts the following options:

all_seed**c0_seed****c1_seed****c2_seed****c3_seed**

Set noise seed for specific pixel component or all pixel components in case of *all_seed*. Default value is 123457.

all_strength, alls**c0_strength, c0s****c1_strength, c1s****c2_strength, c2s****c3_strength, c3s**

Set noise strength for specific pixel component or all pixel components in case *all_strength*. Default value is 0. Allowed range is [0, 100].

all_flags, allf

c0_flags, c0f

c1_flags, c1f

c2_flags, c2f

c3_flags, c3f

Set pixel component flags or set flags for all components if *all_flags*. Available values for component flags are:

- a** averaged temporal noise (smoother)
- p** mix random noise with a (semi)regular pattern
- t** temporal noise (noise pattern changes between frames)
- u** uniform noise (gaussian otherwise)

Examples

Add temporal and uniform noise to input video:

```
noise=all=20:allf=t+u
```

normalize

Normalize RGB video (aka histogram stretching, contrast stretching). See: [https://en.wikipedia.org/wiki/Normalization_\(image_processing\)](https://en.wikipedia.org/wiki/Normalization_(image_processing))

For each channel of each frame, the filter computes the input range and maps it linearly to the user-specified output range. The output range defaults to the full dynamic range from pure black to pure white.

Temporal smoothing can be used on the input range to reduce flickering (rapid changes in brightness) caused when small dark or bright objects enter or leave the scene. This is similar to the auto-exposure (automatic gain control) on a video camera, and, like a video camera, it may cause a period of over- or under-exposure of the video.

The R,G,B channels can be normalized independently, which may cause some color shifting, or linked together as a single channel, which prevents color shifting. Linked normalization preserves hue.

Independent normalization does not, so it can be used to remove some color casts. Independent and linked normalization can be combined in any ratio.

The normalize filter accepts the following options:

blackpt**whitept**

Colors which define the output range. The minimum input value is mapped to the *blackpt*. The maximum input value is mapped to the *whitept*. The defaults are black and white respectively. Specifying white for *blackpt* and black for *whitept* will give color-inverted, normalized video. Shades of grey can be used to reduce the dynamic range (contrast). Specifying saturated colors here can create some interesting effects.

smoothing

The number of previous frames to use for temporal smoothing. The input range of each channel is smoothed using a rolling average over the current frame and the *smoothing* previous frames. The default is 0 (no temporal smoothing).

independence

Controls the ratio of independent (color shifting) channel normalization to linked (color preserving) normalization. 0.0 is fully linked, 1.0 is fully independent. Defaults to 1.0 (fully independent).

strength

Overall strength of the filter. 1.0 is full strength. 0.0 is a rather expensive no-op. Defaults to 1.0 (full strength).

Commands

This filter supports same **commands** as options, excluding *smoothing* option. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

Examples

Stretch video contrast to use the full dynamic range, with no temporal smoothing; may flicker depending on the source content:

```
normalize=blackpt=black:whitept=white:smoothing=0
```

As above, but with 50 frames of temporal smoothing; flicker should be reduced, depending on the source content:

```
normalize=blackpt=black:whitept=white:smoothing=50
```

As above, but with hue-preserving linked channel normalization:

```
normalize=blackpt=black:whitept=white:smoothing=50:independence=0
```

As above, but with half strength:

```
normalize=blackpt=black:whitept=white:smoothing=50:independence=0:strength=0.5
```

Map the darkest input color to red, the brightest input color to cyan:

```
normalize=blackpt=red:whitept=cyan
```

null

Pass the video source unchanged to the output.

ocr

Optical Character Recognition

This filter uses Tesseract for optical character recognition. To enable compilation of this filter, you need to configure FFmpeg with "--enable-libtesseract".

It accepts the following options:

datapath

Set datapath to tesseract data. Default is to use whatever was set at installation.

language

Set language, default is "eng".

whitelist

Set character whitelist.

blacklist

Set character blacklist.

The filter exports recognized text as the frame metadata "lavfi.ocr.text". The filter exports confidence of recognized words as the frame metadata "lavfi.ocr.confidence".

ocv

Apply a video transform using libopencv.

To enable this filter, install the libopencv library and headers and configure FFmpeg with "--enable-libopencv".

It accepts the following parameters:

filter_name

The name of the libopencv filter to apply.

filter_params

The parameters to pass to the libopencv filter. If not specified, the default values are assumed.

Refer to the official libopencv documentation for more precise information:

<<http://docs.opencv.org/master/modules/imgproc/doc/filtering.html>>

Several libopencv filters are supported; see the following subsections.

dilate

Dilate an image by using a specific structuring element. It corresponds to the libopencv function "cvDilate".

It accepts the parameters: *struct_el|nb_iterations*.

struct_el represents a structuring element, and has the syntax: *colsxrows+anchor_xxanchor_y/shape*

cols and *rows* represent the number of columns and rows of the structuring element, *anchor_x* and *anchor_y* the anchor point, and *shape* the shape for the structuring element. *shape* must be "rect", "cross", "ellipse", or "custom".

If the value for *shape* is "custom", it must be followed by a string of the form "*=filename*". The file with name *filename* is assumed to represent a binary image, with each printable character corresponding to a bright pixel. When a custom *shape* is used, *cols* and *rows* are ignored, the number of columns and rows of the read file are assumed instead.

The default value for *struct_el* is "3x3+0x0/rect".

nb_iterations specifies the number of times the transform is applied to the image, and defaults to 1.

Some examples:

```
# Use the default values
ocv=dilate

# Dilate using a structuring element with a 5x5 cross, iterating two times
ocv=filter_name=dilate:filter_params=5x5+2x2/cross|2

# Read the shape from the file diamond.shape, iterating two times.
# The file diamond.shape may contain a pattern of characters like this
# *
# ***
# *****
# ***
# *
# The specified columns and rows are ignored
# but the anchor point coordinates are not
ocv=dilate:0x0+2x2/custom=diamond.shape|2
```

erode

Erode an image by using a specific structuring element. It corresponds to the libopencv function "cvErode".

It accepts the parameters: *struct_el:nb_iterations*, with the same syntax and semantics as the **dilate** filter.

smooth

Smooth the input video.

The filter takes the following parameters: *type|param1|param2|param3|param4*.

type is the type of smooth filter to apply, and must be one of the following values: "blur", "blur_no_scale", "median", "gaussian", or "bilateral". The default value is "gaussian".

The meaning of *param1*, *param2*, *param3*, and *param4* depends on the smooth type. *param1* and *param2* accept integer positive values or 0. *param3* and *param4* accept floating point values.

The default value for *param1* is 3. The default value for the other parameters is 0.

These parameters correspond to the parameters assigned to the libopencv function "cvSmooth".

oscilloscope

2D Video Oscilloscope.

Useful to measure spatial impulse, step responses, chroma delays, etc.

It accepts the following parameters:

- x** Set scope center x position.
- y** Set scope center y position.
- s** Set scope size, relative to frame diagonal.
- t** Set scope tilt/rotation.
- o** Set trace opacity.
- tx** Set trace center x position.
- ty** Set trace center y position.
- tw** Set trace width, relative to width of frame.
- th** Set trace height, relative to height of frame.
- c** Set which components to trace. By default it traces first three components.
- g** Draw trace grid. By default is enabled.
- st** Draw some statistics. By default is enabled.
- sc** Draw scope. By default is enabled.

Commands

This filter supports same **commands** as options. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

Examples

- ⊕ Inspect full first row of video frame.

```
oscilloscope=x=0.5:y=0:s=1
```

- ⊕ Inspect full last row of video frame.

```
oscilloscope=x=0.5:y=1:s=1
```

- ⊕ Inspect full 5th line of video frame of height 1080.

```
oscilloscope=x=0.5:y=5/1080:s=1
```

- ⊕ Inspect full last column of video frame.

```
oscilloscope=x=1:y=0.5:s=1:t=1
```

overlay

Overlay one video on top of another.

It takes two inputs and has one output. The first input is the "main" video on which the second input is overlaid.

It accepts the following parameters:

A description of the accepted options follows.

x

- y** Set the expression for the x and y coordinates of the overlaid video on the main video. Default value is "0" for both expressions. In case the expression is invalid, it is set to a huge value (meaning that the overlay will not be displayed within the output visible area).

eof_action

See **framesync**.

eval Set when the expressions for **x**, and **y** are evaluated.

It accepts the following values:

init only evaluate expressions once during the filter initialization or when a command is processed

frame

evaluate expressions for each incoming frame

Default value is **frame**.

shortest

See **framesync**.

format

Set the format for the output video.

It accepts the following values:

yuv420

force YUV420 output

yuv420p10

force YUV420p10 output

yuv422

force YUV422 output

yuv422p10

force YUV422p10 output

yuv444

force YUV444 output

rgb force packed RGB output

gbrp

force planar RGB output

auto automatically pick format

Default value is **yuv420**.

repeatlast

See **framesync**.

alpha

Set format of alpha of the overlaid video, it can be *straight* or *premultiplied*. Default is *straight*.

The **x**, and **y** expressions can contain the following parameters.

main_w, **W**

main_h, **H**

The main input width and height.

overlay_w, **w**

overlay_h, **h**

The overlay input width and height.

x

y The computed values for *x* and *y*. They are evaluated for each new frame.

hsub

vsub

horizontal and vertical chroma subsample values of the output format. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

n the number of input frame, starting from 0

pos the position in the file of the input frame, NAN if unknown

t The timestamp, expressed in seconds. It's NAN if the input timestamp is unknown.

This filter also supports the **framesync** options.

Note that the *n*, *pos*, *t* variables are available only when evaluation is done *per frame*, and will evaluate

to NAN when **eval** is set to **init**.

Be aware that frames are taken from each input video in timestamp order, hence, if their initial timestamps differ, it is a good idea to pass the two inputs through a `setpts=PTS-STARTPTS` filter to have them begin in the same zero timestamp, as the example for the `movie` filter does.

You can chain together more overlays but you should test the efficiency of such approach.

Commands

This filter supports the following commands:

x

y Modify the x and y of the overlay input. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

Examples

- ⊕ Draw the overlay at 10 pixels from the bottom right corner of the main video:

```
overlay=main_w-overlay_w-10:main_h-overlay_h-10
```

Using named options the example above becomes:

```
overlay=x=main_w-overlay_w-10:y=main_h-overlay_h-10
```

- ⊕ Insert a transparent PNG logo in the bottom left corner of the input, using the **ffmpeg** tool with the `"-filter_complex"` option:

```
ffmpeg -i input -i logo -filter_complex 'overlay=10:main_h-overlay_h-10' output
```

- ⊕ Insert 2 different transparent PNG logos (second logo on bottom right corner) using the **ffmpeg** tool:

```
ffmpeg -i input -i logo1 -i logo2 -filter_complex 'overlay=x=10:y=H-h-10,overlay=x=W-w-10:y=H-h-10' ou
```

- ⊕ Add a transparent color layer on top of the main video; "WxH" must specify the size of the main input to the overlay filter:

```
color=color=red@.3:size=WxH [over]; [in][over] overlay [out]
```

- ⊕ Play an original video and a filtered version (here with the deshake filter) side by side using the **ffplay** tool:

```
ffplay input.avi -vf 'split[a][b]; [a]pad=iw*2:ih[src]; [b]deshake[filter]; [src][filter]overlay=w'
```

The above command is the same as:

```
ffplay input.avi -vf 'split[b], pad=iw*2[src], [b]deshake, [src]overlay=w'
```

- ⊕ Make a sliding overlay appearing from the left to the right top part of the screen starting since time 2:

```
overlay=x='if(gte(t,2), -w+(t-2)*20, NAN)':y=0
```

- ⊕ Compose output by putting two input videos side to side:

```
ffmpeg -i left.avi -i right.avi -filter_complex "
nullsrc=size=200x100 [background];
[0:v] setpts=PTS-STARTPTS, scale=100x100 [left];
[1:v] setpts=PTS-STARTPTS, scale=100x100 [right];
[background][left] overlay=shortest=1 [background+left];
[background+left][right] overlay=shortest=1:x=100 [left+right]
"
```

- ⊕ Mask 10-20 seconds of a video by applying the delogo filter to a section

```
ffmpeg -i test.avi -codec:v:0 wmv2 -ar 11025 -b:v 9000k
-vf '[in]split[split_main][split_delogo];[split_delogo]trim=start=360:end=371,delogo=0:0:640:480[delogod]
masked.avi
```

- ⊕ Chain several overlays in cascade:

```
nullsrc=s=200x200 [bg];
testsrc=s=100x100, split=4 [in0][in1][in2][in3];
[in0] lutrgb=r=0, [bg] overlay=0:0 [mid0];
[in1] lutrgb=g=0, [mid0] overlay=100:0 [mid1];
[in2] lutrgb=b=0, [mid1] overlay=0:100 [mid2];
[in3] null, [mid2] overlay=100:100 [out0]
```

overlay_cuda

Overlay one video on top of another.

This is the CUDA variant of the **overlay** filter. It only accepts CUDA frames. The underlying input pixel formats have to match.

It takes two inputs and has one output. The first input is the "main" video on which the second input is overlaid.

It accepts the following parameters:

x

y Set expressions for the x and y coordinates of the overlaid video on the main video.

They can contain the following parameters:

main_w, W

main_h, H

The main input width and height.

overlay_w, w

overlay_h, h

The overlay input width and height.

x

y The computed values for x and y. They are evaluated for each new frame.

n The ordinal index of the main input frame, starting from 0.

pos The byte offset position in the file of the main input frame, NAN if unknown.

t The timestamp of the main input frame, expressed in seconds, NAN if unknown.

Default value is "0" for both expressions.

eval Set when the expressions for **x** and **y** are evaluated.

It accepts the following values:

init Evaluate expressions once during filter initialization or when a command is processed.

frame

Evaluate expressions for each incoming frame

Default value is **frame**.

eof_action

See **framesync**.

shortest

See **framesync**.

repeatlast

See **framesync**.

This filter also supports the **framesync** options.

owdenoise

Apply Overcomplete Wavelet denoiser.

The filter accepts the following options:

depth

Set depth.

Larger depth values will denoise lower frequency components more, but slow down filtering.

Must be an int in the range 8-16, default is 8.

luma_strength, ls

Set luma strength.

Must be a double value in the range 0-1000, default is 1.0.

chroma_strength, cs

Set chroma strength.

Must be a double value in the range 0-1000, default is 1.0.

pad

Add paddings to the input image, and place the original input at the provided *x*, *y* coordinates.

It accepts the following parameters:

width, w**height, h**

Specify an expression for the size of the output image with the paddings added. If the value for *width* or *height* is 0, the corresponding input size is used for the output.

The *width* expression can reference the value set by the *height* expression, and vice versa.

The default value of *width* and *height* is 0.

x

y Specify the offsets to place the input image at within the padded area, with respect to the top/left border of the output image.

The *x* expression can reference the value set by the *y* expression, and vice versa.

The default value of *x* and *y* is 0.

If *x* or *y* evaluate to a negative number, they'll be changed so the input image is centered on the padded area.

color

Specify the color of the padded area. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**.

The default value of *color* is "black".

eval Specify when to evaluate *width*, *height*, *x* and *y* expression.

It accepts the following values:

init Only evaluate expressions once during the filter initialization or when a command is processed.

frame

Evaluate expressions for each incoming frame.

Default value is **init**.

aspect

Pad to aspect instead to a resolution.

The value for the *width*, *height*, *x*, and *y* options are expressions containing the following constants:

in_w**in_h**

The input video width and height.

iw

ih These are the same as *in_w* and *in_h*.

out_w**out_h**

The output width and height (the size of the padded area), as specified by the *width* and *height* expressions.

ow

oh These are the same as *out_w* and *out_h*.

x

y The x and y offsets as specified by the *x* and *y* expressions, or NAN if not yet specified.

a same as *iw / ih*

sar input sample aspect ratio

dar input display aspect ratio, it is the same as $(iw / ih) * sar$

hsub**vsub**

The horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

Examples

- ⊕ Add paddings with the color "violet" to the input video. The output video size is 640x480, and the top-left corner of the input video is placed at column 0, row 40

pad=640:480:0:40:violet

The example above is equivalent to the following command:

```
pad=width=640:height=480:x=0:y=40:color=violet
```

- ⊕ Pad the input to get an output with dimensions increased by 3/2, and put the input video at the center of the padded area:

```
pad="3/2*iw:3/2*ih:(ow-iw)/2:(oh-ih)/2"
```

- ⊕ Pad the input to get a squared output with size equal to the maximum value between the input width and height, and put the input video at the center of the padded area:

```
pad="max(iw\,ih):ow:(ow-iw)/2:(oh-ih)/2"
```

- ⊕ Pad the input to get a final w/h ratio of 16:9:

```
pad="ih*16/9:ih:(ow-iw)/2:(oh-ih)/2"
```

- ⊕ In case of anamorphic video, in order to set the output display aspect correctly, it is necessary to use *sar* in the expression, according to the relation:

$$(ih * X / ih) * sar = output_dar$$

$$X = output_dar / sar$$

Thus the previous example needs to be modified to:

```
pad="ih*16/9/sar:ih:(ow-iw)/2:(oh-ih)/2"
```

- ⊕ Double the output size and put the input video in the bottom-right corner of the output padded area:

```
pad="2*iw:2*ih:ow-iw:oh-ih"
```

palettegen

Generate one palette for a whole video stream.

It accepts the following options:

max_colors

Set the maximum number of colors to quantize in the palette. Note: the palette will still contain

256 colors; the unused palette entries will be black.

reserve_transparent

Create a palette of 255 colors maximum and reserve the last one for transparency. Reserving the transparency color is useful for GIF optimization. If not set, the maximum of colors in the palette will be 256. You probably want to disable this option for a standalone image. Set by default.

transparency_color

Set the color that will be used as background for transparency.

stats_mode

Set statistics mode.

It accepts the following values:

full Compute full frame histograms.

diff Compute histograms only for the part that differs from previous frame. This might be relevant to give more importance to the moving part of your input if the background is static.

single

Compute new histogram for each frame.

Default value is *full*.

The filter also exports the frame metadata "lavfi.color_quant_ratio" ("nb_color_in / nb_color_out") which you can use to evaluate the degree of color quantization of the palette. This information is also visible at *info* logging level.

Examples

- ⊕ Generate a representative palette of a given video using **ffmpeg**:

```
ffmpeg -i input.mkv -vf palettegen palette.png
```

paletteuse

Use a palette to downsample an input video stream.

The filter takes two inputs: one video stream and a palette. The palette must be a 256 pixels image.

It accepts the following options:

dither

Select dithering mode. Available algorithms are:

bayer

Ordered 8x8 bayer dithering (deterministic)

heckbert

Dithering as defined by Paul Heckbert in 1982 (simple error diffusion). Note: this dithering is sometimes considered "wrong" and is included as a reference.

floyd_steinberg

Floyd and Steingberg dithering (error diffusion)

sierra2

Frankie Sierra dithering v2 (error diffusion)

sierra2_4a

Frankie Sierra dithering v2 "Lite" (error diffusion)

sierra3

Frankie Sierra dithering v3 (error diffusion)

burkes

Burkes dithering (error diffusion)

atkinson

Atkinson dithering by Bill Atkinson at Apple Computer (error diffusion)

Default is *sierra2_4a*.

bayer_scale

When *bayer* dithering is selected, this option defines the scale of the pattern (how much the crosshatch pattern is visible). A low value means more visible pattern for less banding, and higher value means less visible pattern at the cost of more banding.

The option must be an integer value in the range [0,5]. Default is 2.

diff_mode

If set, define the zone to process

rectangle

Only the changing rectangle will be reprocessed. This is similar to GIF cropping/offsetting compression mechanism. This option can be useful for speed if only a part of the image is changing, and has use cases such as limiting the scope of the error diffusal **dither** to the rectangle that bounds the moving scene (it leads to more deterministic output if the scene doesn't change much, and as a result less moving noise and better GIF compression).

Default is *none*.

new Take new palette for each output frame.

alpha_threshold

Sets the alpha threshold for transparency. Alpha values above this threshold will be treated as completely opaque, and values below this threshold will be treated as completely transparent.

The option must be an integer value in the range [0,255]. Default is *128*.

Examples

- ⊕ Use a palette (generated for example with **palettegen**) to encode a GIF using **ffmpeg**:

```
ffmpeg -i input.mkv -i palette.png -lavfi paletteuse output.gif
```

perspective

Correct perspective of video not recorded perpendicular to the screen.

A description of the accepted parameters follows.

x0

y0

x1

y1

x2

y2

x3

y3 Set coordinates expression for top left, top right, bottom left and bottom right corners. Default values are "0:0:W:0:0:H:W:H" with which perspective will remain unchanged. If the "sense" option is set to "source", then the specified points will be sent to the corners of the destination. If

the "sense" option is set to "destination", then the corners of the source will be sent to the specified coordinates.

The expressions can use the following variables:

W

H the width and height of video frame.

in Input frame count.

on Output frame count.

interpolation

Set interpolation for perspective correction.

It accepts the following values:

linear

cubic

Default value is **linear**.

sense

Set interpretation of coordinate options.

It accepts the following values:

0, source

Send point in the source specified by the given coordinates to the corners of the destination.

1, destination

Send the corners of the source to the point in the destination specified by the given coordinates.

Default value is **source**.

eval Set when the expressions for coordinates **x0,y0,...x3,y3** are evaluated.

It accepts the following values:

init only evaluate expressions once during the filter initialization or when a command is processed

frame

evaluate expressions for each incoming frame

Default value is **init**.

phase

Delay interlaced video by one field time so that the field order changes.

The intended use is to fix PAL movies that have been captured with the opposite field order to the film-to-video transfer.

A description of the accepted parameters follows.

mode

Set phase mode.

It accepts the following values:

- t** Capture field order top-first, transfer bottom-first. Filter will delay the bottom field.
- b** Capture field order bottom-first, transfer top-first. Filter will delay the top field.
- p** Capture and transfer with the same field order. This mode only exists for the documentation of the other options to refer to, but if you actually select it, the filter will faithfully do nothing.
- a** Capture field order determined automatically by field flags, transfer opposite. Filter selects among **t** and **b** modes on a frame by frame basis using field flags. If no field information is available, then this works just like **u**.
- u** Capture unknown or varying, transfer opposite. Filter selects among **t** and **b** on a frame by frame basis by analyzing the images and selecting the alternative that produces best match between the fields.
- T** Capture top-first, transfer unknown or varying. Filter selects among **t** and **p** using image analysis.

- B** Capture bottom-first, transfer unknown or varying. Filter selects among **b** and **p** using image analysis.
- A** Capture determined by field flags, transfer unknown or varying. Filter selects among **t**, **b** and **p** using field flags and image analysis. If no field information is available, then this works just like **U**. This is the default mode.
- U** Both capture and transfer unknown or varying. Filter selects among **t**, **b** and **p** using image analysis only.

Commands

This filter supports the all above options as **commands**.

photosensitivity

Reduce various flashes in video, so to help users with epilepsy.

It accepts the following options:

frames, f

Set how many frames to use when filtering. Default is 30.

threshold, t

Set detection threshold factor. Default is 1. Lower is stricter.

skip Set how many pixels to skip when sampling frames. Default is 1. Allowed range is from 1 to 1024.

bypass

Leave frames unchanged. Default is disabled.

pixdesctest

Pixel format descriptor test filter, mainly useful for internal testing. The output video should be equal to the input video.

For example:

```
format=monow, pixdesctest
```

can be used to test the monowhite pixel format descriptor definition.

pixelize

Apply pixelization to video stream.

The filter accepts the following options:

width, w**height, h**

Set block dimensions that will be used for pixelization. Default value is 16.

mode, m

Set the mode of pixelization used.

Possible values are:

avg

min

max

Default value is "avg".

planes, p

Set what planes to filter. Default is to filter all planes.

Commands

This filter supports all options as **commands**.

pixscope

Display sample values of color channels. Mainly useful for checking color and levels. Minimum supported resolution is 640x480.

The filters accept the following options:

x Set scope X position, relative offset on X axis.

y Set scope Y position, relative offset on Y axis.

w Set scope width.

h Set scope height.

- o Set window opacity. This window also holds statistics about pixel area.
- wx** Set window X position, relative offset on X axis.
- wy** Set window Y position, relative offset on Y axis.

Commands

This filter supports same **commands** as options.

pp

Enable the specified chain of postprocessing subfilters using libpostproc. This library should be automatically selected with a GPL build ("--enable-gpl"). Subfilters must be separated by '/' and can be disabled by prepending a '-'. Each subfilter and some options have a short and a long name that can be used interchangeably, i.e. dr/dering are the same.

The filters accept the following options:

subfilters

Set postprocessing subfilters string.

All subfilters share common options to determine their scope:

a/autoq

Honor the quality commands for this subfilter.

c/chrom

Do chrominance filtering, too (default).

y/nochrom

Do luminance filtering only (no chrominance).

n/noluma

Do chrominance filtering only (no luminance).

These options can be appended after the subfilter name, separated by a '|'.

Available subfilters are:

hb/hdeblock[|**difference**[|**flatness**]]

Horizontal deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

vb/vdeblock[*difference*[*flatness*]]

Vertical deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

ha/hadeblock[*difference*[*flatness*]]

Accurate horizontal deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

va/vadeblock[*difference*[*flatness*]]

Accurate vertical deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

The horizontal and vertical deblocking filters share the difference and flatness values so you cannot set different horizontal and vertical thresholds.

h1/x1hdeblock

Experimental horizontal deblocking filter

v1/x1vdeblock

Experimental vertical deblocking filter

dr/dering

Deringing filter

tn/tmpnoise[[threshold1[[threshold2[[threshold3]]], temporal noise reducer**threshold1**

larger -> stronger filtering

threshold2

larger -> stronger filtering

threshold3

larger -> stronger filtering

al/autolevels[:f/fullyrange], automatic brightness / contrast correction**f/fullyrange**

Stretch luminance to "0-255".

lb/linblenddeint

Linear blend deinterlacing filter that deinterlaces the given block by filtering all lines with a "(1 2 1)" filter.

li/linipoldeint

Linear interpolating deinterlacing filter that deinterlaces the given block by linearly interpolating every second line.

ci/cubicpoldeint

Cubic interpolating deinterlacing filter deinterlaces the given block by cubically interpolating every second line.

md/mediandeint

Median deinterlacing filter that deinterlaces the given block by applying a median filter to every second line.

fd/ffmpegdeint

FFmpeg deinterlacing filter that deinterlaces the given block by filtering every second line with a "(-1 4 2 4 -1)" filter.

l5/lowpass5

Vertically applied FIR lowpass deinterlacing filter that deinterlaces the given block by filtering all lines with a "(-1 2 6 2 -1)" filter.

fq/forceQuant[*quantizer*]

Overrides the quantizer table from the input with the constant quantizer you specify.

quantizer

Quantizer to use

de/default

Default pp filter combination ("hb|a,vb|a,dr|a")

fa/fast

Fast pp filter combination ("h1|a,v1|a,dr|a")

ac High quality pp filter combination ("ha|a|128|7,va|a,dr|a")

Examples

- ⊕ Apply horizontal and vertical deblocking, deringing and automatic brightness/contrast:

```
pp=hb/vb/dr/al
```

- ⊕ Apply default filters without brightness/contrast correction:

```
pp=de/-al
```

- ⊕ Apply default filters and temporal denoiser:

```
pp=default/tmpnoise|1|2|3
```

- ⊕ Apply deblocking on luminance only, and switch vertical deblocking on or off automatically depending on available CPU time:

```
pp=hb|y/vb|a
```

pp7

Apply Postprocessing filter 7. It is variant of the **spp** filter, similar to spp = 6 with 7 point DCT, where only the center sample is used after IDCT.

The filter accepts the following options:

qp Force a constant quantization parameter. It accepts an integer in range 0 to 63. If not set, the filter will use the QP from the video stream (if available).

mode

Set thresholding mode. Available modes are:

hard

Set hard thresholding.

soft Set soft thresholding (better de-ringing effect, but likely blurrier).

medium

Set medium thresholding (good results, default).

premultiply

Apply alpha premultiply effect to input video stream using first plane of second stream as alpha.

Both streams must have same dimensions and same pixel format.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

inplace

Do not require 2nd input for processing, instead use alpha plane from input stream.

prewitt

Apply prewitt operator to input video stream.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

scale

Set value which will be multiplied with filtered result.

delta

Set value which will be added to filtered result.

Commands

This filter supports the all above options as **commands**.

pseudocolor

Alter frame colors in video with pseudocolors.

This filter accepts the following options:

c0 set pixel first component expression

c1 set pixel second component expression

c2 set pixel third component expression

c3 set pixel fourth component expression, corresponds to the alpha component

index, i

set component to use as base for altering colors

preset, p

Pick one of built-in LUTs. By default is set to none.

Available LUTs:

magma

inferno

plasma

viridis

turbo

cividis

range1

range2

shadows

highlights

solar
nominal
preferred
total
spectral

opacity

Set opacity of output colors. Allowed range is from 0 to 1. Default value is set to 1.

Each of the expression options specifies the expression to use for computing the lookup table for the corresponding pixel component values.

The expressions can contain the following constants and functions:

w

h The input width and height.

val The input value for the pixel component.

ymin, umin, vmin, amin

The minimum allowed component value.

ymax, umax, vmax, amax

The maximum allowed component value.

All expressions default to "val".

Commands

This filter supports the all above options as **commands**.

Examples

⊕ Change too high luma values to gradient:

```
pseudocolor=""if(between(val,ymax,amax),lerp(ymin,ymax,(val-ymax)/(amax-ymax)),-1):if(between(val,ym
```

psnr

Obtain the average, maximum and minimum PSNR (Peak Signal to Noise Ratio) between two input videos.

This filter takes in input two input videos, the first input is considered the "main" source and is passed unchanged to the output. The second input is used as a "reference" video for computing the PSNR.

Both video inputs must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained average PSNR is printed through the logging system.

The filter stores the accumulated MSE (mean squared error) of each frame, and at the end of the processing it is averaged across all frames equally, and the following formula is applied to obtain the PSNR:

$$\text{PSNR} = 10 * \log_{10}(\text{MAX}^2 / \text{MSE})$$

Where MAX is the average of the maximum values of each component of the image.

The description of the accepted parameters follows.

stats_file, f

If specified the filter will use the named file to save the PSNR of each individual frame. When filename equals "-" the data is sent to standard output.

stats_version

Specifies which version of the stats file format to use. Details of each format are written below. Default value is 1.

stats_add_max

Determines whether the max value is output to the stats log. Default value is 0. Requires stats_version >= 2. If this is set and stats_version < 2, the filter will return an error.

This filter also supports the **framesync** options.

The file printed if *stats_file* is selected, contains a sequence of key/value pairs of the form *key:value* for each compared couple of frames.

If a *stats_version* greater than 1 is specified, a header line precedes the list of per-frame-pair stats, with key value pairs following the frame format with the following parameters:

psnr_log_version

The version of the log file format. Will match *stats_version*.

fields

A comma separated list of the per-frame-pair parameters included in the log.

A description of each shown per-frame-pair parameter follows:

n sequential number of the input frame, starting from 1

mse_avg

Mean Square Error pixel-by-pixel average difference of the compared frames, averaged over all the image components.

mse_y, mse_u, mse_v, mse_r, mse_g, mse_b, mse_a

Mean Square Error pixel-by-pixel average difference of the compared frames for the component specified by the suffix.

psnr_y, psnr_u, psnr_v, psnr_r, psnr_g, psnr_b, psnr_a

Peak Signal to Noise ratio of the compared frames for the component specified by the suffix.

max_avg, max_y, max_u, max_v

Maximum allowed value for each channel, and average over all channels.

Examples

⊕ For example:

```
movie=ref_movie.mpg, setpts=PTS-STARTPTS [main];
[main][ref] psnr="stats_file=stats.log" [out]
```

On this example the input file being processed is compared with the reference file *ref_movie.mpg*. The PSNR of each individual frame is stored in *stats.log*.

⊕ Another example with different containers:

```
ffmpeg -i main.mpg -i ref.mkv -lavfi "[0:v]settb=AVTB,setpts=PTS-STARTPTS[main];[1:v]settb=AVTB,se
```

pullup

Pulldown reversal (inverse telecine) filter, capable of handling mixed hard-telecine, 24000/1001 fps progressive, and 30000/1001 fps progressive content.

The pullup filter is designed to take advantage of future context in making its decisions. This filter is

stateless in the sense that it does not lock onto a pattern to follow, but it instead looks forward to the following fields in order to identify matches and rebuild progressive frames.

To produce content with an even framerate, insert the fps filter after pullup, use "fps=24000/1001" if the input frame rate is 29.97fps, "fps=24" for 30fps and the (rare) telecined 25fps input.

The filter accepts the following options:

jl

jr

jt

jb These options set the amount of "junk" to ignore at the left, right, top, and bottom of the image, respectively. Left and right are in units of 8 pixels, while top and bottom are in units of 2 lines. The default is 8 pixels on each side.

sb Set the strict breaks. Setting this option to 1 will reduce the chances of filter generating an occasional mismatched frame, but it may also cause an excessive number of frames to be dropped during high motion sequences. Conversely, setting it to -1 will make filter match fields more easily. This may help processing of video where there is slight blurring between the fields, but may also cause there to be interlaced frames in the output. Default value is 0.

mp Set the metric plane to use. It accepts the following values:

l Use luma plane.

u Use chroma blue plane.

v Use chroma red plane.

This option may be set to use chroma plane instead of the default luma plane for doing filter's computations. This may improve accuracy on very clean source material, but more likely will decrease accuracy, especially if there is chroma noise (rainbow effect) or any grayscale video. The main purpose of setting **mp** to a chroma plane is to reduce CPU load and make pullup usable in realtime on slow machines.

For best results (without duplicated frames in the output file) it is necessary to change the output frame rate. For example, to inverse telecine NTSC input:

```
ffmpeg -i input -vf pullup -r 24000/1001 ...
```

qp

Change video quantization parameters (QP).

The filter accepts the following option:

qp Set expression for quantization parameter.

The expression is evaluated through the eval API and can contain, among others, the following constants:

known

1 if index is not 129, 0 otherwise.

qp Sequential index starting from -129 to 128.

Examples

⊕ Some equation like:

$$qp=2+2*\sin(\text{PI}*qp)$$

random

Flush video frames from internal cache of frames into a random order. No frame is discarded. Inspired by **frei0r** nervous filter.

frames

Set size in number of frames of internal cache, in range from 2 to 512. Default is 30.

seed

Set seed for random number generator, must be an integer included between 0 and "UINT32_MAX". If not specified, or if explicitly set to less than 0, the filter will try to use a good random seed on a best effort basis.

readeia608

Read closed captioning (EIA-608) information from the top lines of a video frame.

This filter adds frame metadata for "lavfi.readeia608.X.cc" and "lavfi.readeia608.X.line", where "X" is the number of the identified line with EIA-608 data (starting from 0). A description of each metadata value follows:

lavfi.readeia608.X.cc

The two bytes stored as EIA-608 data (printed in hexadecimal).

lavfi.readeia608.X.line

The number of the line on which the EIA-608 data was identified and read.

This filter accepts the following options:

scan_min

Set the line to start scanning for EIA-608 data. Default is 0.

scan_max

Set the line to end scanning for EIA-608 data. Default is 29.

spw Set the ratio of width reserved for sync code detection. Default is 0.27. Allowed range is "[0.1 - 0.7]".

chp Enable checking the parity bit. In the event of a parity error, the filter will output 0x00 for that character. Default is false.

lp Lowpass lines prior to further processing. Default is enabled.

Commands

This filter supports the all above options as **commands**.

Examples

- ⊕ Output a csv with presentation time and the first two lines of identified EIA-608 captioning data.

```
ffprobe -f lavfi -i movie=captioned_video.mov,readeia608 -show_entries frame=pts_time:frame_tags=lavfi.r
```

readvtc

Read vertical interval timecode (VITC) information from the top lines of a video frame.

The filter adds frame metadata key "lavfi.readvtc.tc_str" with the timecode value, if a valid timecode has been detected. Further metadata key "lavfi.readvtc.found" is set to 0/1 depending on whether timecode data has been found or not.

This filter accepts the following options:

scan_max

Set the maximum number of lines to scan for VITC data. If the value is set to "-1" the full video frame is scanned. Default is 45.

thr_b

Set the luma threshold for black. Accepts float numbers in the range [0.0,1.0], default value is 0.2. The value must be equal or less than "thr_w".

thr_w

Set the luma threshold for white. Accepts float numbers in the range [0.0,1.0], default value is 0.6. The value must be equal or greater than "thr_b".

Examples

- ⊕ Detect and draw VITC data onto the video frame; if no valid VITC is detected, draw "--:--:--:--" as a placeholder:

```
ffmpeg -i input.avi -filter:v 'readvitc,drawtext=fontfile=FreeMono.ttf:text=% { metadata\\:lavfi.readvitc.tc_str
```

remap

Remap pixels using 2nd: Xmap and 3rd: Ymap input video stream.

Destination pixel at position (X, Y) will be picked from source (x, y) position where $x = Xmap(X, Y)$ and $y = Ymap(X, Y)$. If mapping values are out of range, zero value for pixel will be used for destination pixel.

Xmap and Ymap input video streams must be of same dimensions. Output video stream will have Xmap/Ymap video stream dimensions. Xmap and Ymap input video streams are 16bit depth, single channel.

format

Specify pixel format of output from this filter. Can be "color" or "gray". Default is "color".

fill Specify the color of the unmapped pixels. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. Default color is "black".

removegrain

The removegrain filter is a spatial denoiser for progressive video.

m0 Set mode for the first plane.

m1 Set mode for the second plane.

m2 Set mode for the third plane.

m3 Set mode for the fourth plane.

Range of mode is from 0 to 24. Description of each mode follows:

0 Leave input plane unchanged. Default.

1 Clips the pixel with the minimum and maximum of the 8 neighbour pixels.

2 Clips the pixel with the second minimum and maximum of the 8 neighbour pixels.

3 Clips the pixel with the third minimum and maximum of the 8 neighbour pixels.

4 Clips the pixel with the fourth minimum and maximum of the 8 neighbour pixels. This is equivalent to a median filter.

5 Line-sensitive clipping giving the minimal change.

6 Line-sensitive clipping, intermediate.

7 Line-sensitive clipping, intermediate.

8 Line-sensitive clipping, intermediate.

9 Line-sensitive clipping on a line where the neighbours pixels are the closest.

10 Replaces the target pixel with the closest neighbour.

11 [1 2 1] horizontal and vertical kernel blur.

12 Same as mode 11.

13 Bob mode, interpolates top field from the line where the neighbours pixels are the closest.

14 Bob mode, interpolates bottom field from the line where the neighbours pixels are the closest.

15 Bob mode, interpolates top field. Same as 13 but with a more complicated interpolation formula.

- 16 Bob mode, interpolates bottom field. Same as 14 but with a more complicated interpolation formula.
- 17 Clips the pixel with the minimum and maximum of respectively the maximum and minimum of each pair of opposite neighbour pixels.
- 18 Line-sensitive clipping using opposite neighbours whose greatest distance from the current pixel is minimal.
- 19 Replaces the pixel with the average of its 8 neighbours.
- 20 Averages the 9 pixels ([1 1 1] horizontal and vertical blur).
- 21 Clips pixels using the averages of opposite neighbour.
- 22 Same as mode 21 but simpler and faster.
- 23 Small edge and halo removal, but reputed useless.
- 24 Similar as 23.

removelogo

Suppress a TV station logo, using an image file to determine which pixels comprise the logo. It works by filling in the pixels that comprise the logo with neighboring pixels.

The filter accepts the following options:

filename, f

Set the filter bitmap file, which can be any image format supported by libavformat. The width and height of the image file must match those of the video stream being processed.

Pixels in the provided bitmap image with a value of zero are not considered part of the logo, non-zero pixels are considered part of the logo. If you use white (255) for the logo and black (0) for the rest, you will be safe. For making the filter bitmap, it is recommended to take a screen capture of a black frame with the logo visible, and then using a threshold filter followed by the erode filter once or twice.

If needed, little splotches can be fixed manually. Remember that if logo pixels are not covered, the filter quality will be much reduced. Marking too many pixels as part of the logo does not hurt as much, but it will increase the amount of blurring needed to cover over the image and will destroy more information than necessary, and extra pixels will slow things down on a large logo.

repeatfields

This filter uses the `repeat_field` flag from the Video ES headers and hard repeats fields based on its value.

reverse

Reverse a video clip.

Warning: This filter requires memory to buffer the entire clip, so trimming is suggested.

Examples

- ⊕ Take the first 5 seconds of a clip, and reverse it.

```
trim=end=5,reverse
```

rgbashift

Shift R/G/B/A pixels horizontally and/or vertically.

The filter accepts the following options:

rh Set amount to shift red horizontally.

rv Set amount to shift red vertically.

gh Set amount to shift green horizontally.

gv Set amount to shift green vertically.

bh Set amount to shift blue horizontally.

bv Set amount to shift blue vertically.

ah Set amount to shift alpha horizontally.

av Set amount to shift alpha vertically.

edge

Set edge mode, can be *smear*, *default*, or *warp*.

Commands

This filter supports the all above options as **commands**.

roberts

Apply roberts cross operator to input video stream.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

scale

Set value which will be multiplied with filtered result.

delta

Set value which will be added to filtered result.

Commands

This filter supports the all above options as **commands**.

rotate

Rotate video by an arbitrary angle expressed in radians.

The filter accepts the following options:

A description of the optional parameters follows.

angle, a

Set an expression for the angle by which to rotate the input video clockwise, expressed as a number of radians. A negative value will result in a counter-clockwise rotation. By default it is set to "0".

This expression is evaluated for each frame.

out_w, ow

Set the output width expression, default value is "iw". This expression is evaluated just once during configuration.

out_h, oh

Set the output height expression, default value is "ih". This expression is evaluated just once during configuration.

bilinear

Enable bilinear interpolation if set to 1, a value of 0 disables it. Default value is 1.

fillcolor, c

Set the color used to fill the output area not covered by the rotated image. For the general syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. If the special value "none" is selected then no background is printed (useful for example if the background is never shown).

Default value is "black".

The expressions for the angle and the output size can contain the following constants and functions:

- n** sequential number of the input frame, starting from 0. It is always NAN before the first frame is filtered.
- t** time in seconds of the input frame, it is set to 0 when the filter is configured. It is always NAN before the first frame is filtered.

hsub

vsub

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

in_w, iw

in_h, ih

the input video width and height

out_w, ow

out_h, oh

the output width and height, that is the size of the padded area as specified by the *width* and *height* expressions

rotw(a)

roth(a)

the minimal width/height required for completely containing the input video rotated by *a* radians.

These are only available when computing the **out_w** and **out_h** expressions.

Examples

- ⊕ Rotate the input by $\pi/6$ radians clockwise:

rotate= $\pi/6$

- ⊕ Rotate the input by $\pi/6$ radians counter-clockwise:

rotate= $-\pi/6$

- ⊕ Rotate the input by 45 degrees clockwise:

rotate= $45 \cdot \pi/180$

- ⊕ Apply a constant rotation with period T, starting from an angle of $\pi/3$:

rotate= $\pi/3 + 2 \cdot \pi \cdot t/T$

- ⊕ Make the input video rotation oscillating with a period of T seconds and an amplitude of A radians:

rotate= $A \cdot \sin(2 \cdot \pi/T \cdot t)$

- ⊕ Rotate the video, output size is chosen so that the whole rotating input video is always completely contained in the output:

rotate= $'2 \cdot \pi \cdot t:ow=hypot(iw,ih):oh=ow'$

- ⊕ Rotate the video, reduce the output size so that no background is ever shown:

rotate= $2 \cdot \pi \cdot t:ow='min(iw,ih)/sqrt(2)':oh=ow:c=none$

Commands

The filter supports the following commands:

a, angle

Set the angle expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

sab

Apply Shape Adaptive Blur.

The filter accepts the following options:

luma_radius, lr

Set luma blur filter strength, must be a value in range 0.1-4.0, default value is 1.0. A greater value will result in a more blurred image, and in slower processing.

luma_pre_filter_radius, lpfr

Set luma pre-filter radius, must be a value in the 0.1-2.0 range, default value is 1.0.

luma_strength, ls

Set luma maximum difference between pixels to still be considered, must be a value in the 0.1-100.0 range, default value is 1.0.

chroma_radius, cr

Set chroma blur filter strength, must be a value in range -0.9-4.0. A greater value will result in a more blurred image, and in slower processing.

chroma_pre_filter_radius, cpfr

Set chroma pre-filter radius, must be a value in the -0.9-2.0 range.

chroma_strength, cs

Set chroma maximum difference between pixels to still be considered, must be a value in the -0.9-100.0 range.

Each chroma option value, if not explicitly specified, is set to the corresponding luma option value.

scale

Scale (resize) the input video, using the libswscale library.

The scale filter forces the output display aspect ratio to be the same of the input, by changing the output sample aspect ratio.

If the input image format is different from the format requested by the next filter, the scale filter will convert the input to the requested format.

Options

The filter accepts the following options, or any of the options supported by the libswscale scaler.

See **the ffmpeg-scaler manual** for the complete list of scaler options.

width, w

height, h

Set the output video dimension expression. Default value is the input dimension.

If the *width* or *w* value is 0, the input width is used for the output. If the *height* or *h* value is 0, the input height is used for the output.

If one and only one of the values is $-n$ with $n \geq 1$, the scale filter will use a value that maintains the aspect ratio of the input image, calculated from the other specified dimension. After that it will, however, make sure that the calculated dimension is divisible by n and adjust the value if necessary.

If both values are $-n$ with $n \geq 1$, the behavior will be identical to both values being set to 0 as previously detailed.

See below for the list of accepted constants for use in the dimension expression.

eval Specify when to evaluate *width* and *height* expression. It accepts the following values:

init Only evaluate expressions once during the filter initialization or when a command is processed.

frame

Evaluate expressions for each incoming frame.

Default value is **init**.

interl

Set the interlacing mode. It accepts the following values:

1 Force interlaced aware scaling.

0 Do not apply interlaced scaling.

-1 Select interlaced aware scaling depending on whether the source frames are flagged as interlaced or not.

Default value is **0**.

flags

Set libswscale scaling flags. See **the ffmpeg-scaler manual** for the complete list of values. If not explicitly specified the filter applies the default flags.

param0, param1

Set libswscale input parameters for scaling algorithms that need them. See **the ffmpeg-scaler manual** for the complete documentation. If not explicitly specified the filter applies empty parameters.

size, s

Set the video size. For the syntax of this option, check the **"Video size" section in the ffmpeg-utils manual**.

in_color_matrix**out_color_matrix**

Set in/output YCbCr color space type.

This allows the autodetected value to be overridden as well as allows forcing a specific value used for the output and encoder.

If not specified, the color space type depends on the pixel format.

Possible values:

auto Choose automatically.

bt709

Format conforming to International Telecommunication Union (ITU) Recommendation BT.709.

fcc Set color space conforming to the United States Federal Communications Commission (FCC) Code of Federal Regulations (CFR) Title 47 (2003) 73.682 (a).

bt601**bt470****smpte170m**

Set color space conforming to:

- ⊕ ITU Radiocommunication Sector (ITU-R) Recommendation BT.601
- ⊕ ITU-R Rec. BT.470-6 (1998) Systems B, B1, and G
- ⊕ Society of Motion Picture and Television Engineers (SMPTE) ST 170:2004

smpte240m

Set color space conforming to SMPTE ST 240:1999.

bt2020

Set color space conforming to ITU-R BT.2020 non-constant luminance system.

in_range**out_range**

Set in/output YCbCr sample range.

This allows the autodetected value to be overridden as well as allows forcing a specific value used for the output and encoder. If not specified, the range depends on the pixel format. Possible values:

auto/unknown

Choose automatically.

jpeg/full/pc

Set full range (0-255 in case of 8-bit luma).

mpeg/limited/tv

Set "MPEG" range (16-235 in case of 8-bit luma).

force_original_aspect_ratio

Enable decreasing or increasing output video width or height if necessary to keep the original aspect ratio. Possible values:

disable

Scale the video as specified and disable this feature.

decrease

The output video dimensions will automatically be decreased if needed.

increase

The output video dimensions will automatically be increased if needed.

One useful instance of this option is that when you know a specific device's maximum allowed resolution, you can use this to limit the output video to that, while retaining the aspect ratio. For example, device A allows 1280x720 playback, and your video is 1920x800. Using this option (set it to decrease) and specifying 1280x720 to the command line makes the output 1280x533.

Please note that this is a different thing than specifying -1 for **w** or **h**, you still need to specify the output resolution for this option to work.

force_divisible_by

Ensures that both the output dimensions, width and height, are divisible by the given integer when used together with **force_original_aspect_ratio**. This works similar to using "-n" in the **w** and **h** options.

This option respects the value set for **force_original_aspect_ratio**, increasing or decreasing the resolution accordingly. The video's aspect ratio may be slightly modified.

This option can be handy if you need to have a video fit within or exceed a defined resolution using **force_original_aspect_ratio** but also have encoder restrictions on width or height divisibility.

The values of the **w** and **h** options are expressions containing the following constants:

in_w

in_h

The input width and height

iw

ih These are the same as *in_w* and *in_h*.

out_w

out_h

The output (scaled) width and height

ow

oh These are the same as *out_w* and *out_h*

a The same as *iw / ih*

sar input sample aspect ratio

dar The input display aspect ratio. Calculated from "(iw / ih) * sar".

hsub

vsub

horizontal and vertical input chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

ohsub

ovsub

horizontal and vertical output chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

n The (sequential) number of the input frame, starting from 0. Only available with "eval=frame".

t The presentation timestamp of the input frame, expressed as a number of seconds. Only available with "eval=frame".

pos The position (byte offset) of the frame in the input stream, or NaN if this information is unavailable and/or meaningless (for example in case of synthetic video). Only available with "eval=frame".

Examples

- ⊕ Scale the input video to a size of 200x100

```
scale=w=200:h=100
```

This is equivalent to:

```
scale=200:100
```

or:

```
scale=200x100
```

- ⊕ Specify a size abbreviation for the output size:

```
scale=qcif
```

which can also be written as:

scale=size=qcif

- ⊕ Scale the input to 2x:

scale=w=2*iw:h=2*ih

- ⊕ The above is the same as:

scale=2*in_w:2*in_h

- ⊕ Scale the input to 2x with forced interlaced scaling:

scale=2*iw:2*ih:interl=1

- ⊕ Scale the input to half size:

scale=w=iw/2:h=ih/2

- ⊕ Increase the width, and set the height to the same size:

scale=3/2*iw:ow

- ⊕ Seek Greek harmony:

scale=iw:1/PHI*iw

scale=ih*PHI:ih

- ⊕ Increase the height, and set the width to 3/2 of the height:

scale=w=3/2*oh:h=3/5*ih

- ⊕ Increase the size, making the size a multiple of the chroma subsample values:

scale="trunc(3/2*iw/hsub)*hsub:trunc(3/2*ih/vsub)*vsub"

- ⊕ Increase the width to a maximum of 500 pixels, keeping the same aspect ratio as the input:

scale=w='min(500, iw*3/2):h=-1'

- ⊕ Make pixels square by combining scale and setsar:

```
scale='trunc(ih*dar):ih',setsar=1/1
```

- ⊕ Make pixels square by combining scale and setsar, making sure the resulting resolution is even (required by some codecs):

```
scale='trunc(ih*dar/2)*2:trunc(ih/2)*2',setsar=1/1
```

Commands

This filter supports the following commands:

width, w

height, h

Set the output video dimension expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

scale_cuda

Scale (resize) and convert (pixel format) the input video, using accelerated CUDA kernels. Setting the output width and height works in the same way as for the **scale** filter.

The filter accepts the following options:

w

h Set the output video dimension expression. Default value is the input dimension.

Allows for the same expressions as the **scale** filter.

interp_algo

Sets the algorithm used for scaling:

nearest

Nearest neighbour

Used by default if input parameters match the desired output.

bilinear

Bilinear

bicubic

Bicubic

This is the default.

lanczos

Lanczos

format

Controls the output pixel format. By default, or if none is specified, the input pixel format is used.

The filter does not support converting between YUV and RGB pixel formats.

passthrough

If set to 0, every frame is processed, even if no conversion is necessary. This mode can be useful to use the filter as a buffer for a downstream frame-consumer that exhausts the limited decoder frame pool.

If set to 1, frames are passed through as-is if they match the desired output parameters. This is the default behaviour.

param

Algorithm-Specific parameter.

Affects the curves of the bicubic algorithm.

force_original_aspect_ratio

force_divisible_by

Work the same as the identical **scale** filter options.

Examples

- ⊕ Scale input to 720p, keeping aspect ratio and ensuring the output is yuv420p.

```
scale_cuda=-2:720:format=yuv420p
```

- ⊕ Upscale to 4K using nearest neighbour algorithm.

```
scale_cuda=4096:2160:interp_algo=nearest
```

- ⊕ Don't do any conversion or scaling, but copy all input frames into newly allocated ones. This can be useful to deal with a filter and encode chain that otherwise exhausts the decoders frame pool.

scale_cuda=passthrough=0

scale_npp

Use the NVIDIA Performance Primitives (libnpp) to perform scaling and/or pixel format conversion on CUDA video frames. Setting the output width and height works in the same way as for the *scale* filter.

The following additional options are accepted:

format

The pixel format of the output CUDA frames. If set to the string "same" (the default), the input format will be kept. Note that automatic format negotiation and conversion is not yet supported for hardware frames

interp_algo

The interpolation algorithm used for resizing. One of the following:

nn Nearest neighbour.

linear

cubic

cubic2p_bspline

2-parameter cubic (B=1, C=0)

cubic2p_catmullrom

2-parameter cubic (B=0, C=1/2)

cubic2p_b05c03

2-parameter cubic (B=1/2, C=3/10)

super

Supersampling

lanczos

force_original_aspect_ratio

Enable decreasing or increasing output video width or height if necessary to keep the original aspect ratio. Possible values:

disable

Scale the video as specified and disable this feature.

decrease

The output video dimensions will automatically be decreased if needed.

increase

The output video dimensions will automatically be increased if needed.

One useful instance of this option is that when you know a specific device's maximum allowed resolution, you can use this to limit the output video to that, while retaining the aspect ratio. For example, device A allows 1280x720 playback, and your video is 1920x800. Using this option (set it to decrease) and specifying 1280x720 to the command line makes the output 1280x533.

Please note that this is a different thing than specifying -1 for **w** or **h**, you still need to specify the output resolution for this option to work.

force_divisible_by

Ensures that both the output dimensions, width and height, are divisible by the given integer when used together with **force_original_aspect_ratio**. This works similar to using "-n" in the **w** and **h** options.

This option respects the value set for **force_original_aspect_ratio**, increasing or decreasing the resolution accordingly. The video's aspect ratio may be slightly modified.

This option can be handy if you need to have a video fit within or exceed a defined resolution using **force_original_aspect_ratio** but also have encoder restrictions on width or height divisibility.

eval Specify when to evaluate *width* and *height* expression. It accepts the following values:

init Only evaluate expressions once during the filter initialization or when a command is processed.

frame

Evaluate expressions for each incoming frame.

The values of the **w** and **h** options are expressions containing the following constants:

in_w

in_h

The input width and height

iw

ih These are the same as *in_w* and *in_h*.

out_w

out_h

The output (scaled) width and height

ow

oh These are the same as *out_w* and *out_h*

a The same as *iw / ih*

sar input sample aspect ratio

dar The input display aspect ratio. Calculated from " $(iw / ih) * sar$ ".

n The (sequential) number of the input frame, starting from 0. Only available with "eval=frame".

t The presentation timestamp of the input frame, expressed as a number of seconds. Only available with "eval=frame".

pos The position (byte offset) of the frame in the input stream, or NaN if this information is unavailable and/or meaningless (for example in case of synthetic video). Only available with "eval=frame".

scale2ref

Scale (resize) the input video, based on a reference video.

See the scale filter for available options, scale2ref supports the same but uses the reference video instead of the main input as basis. scale2ref also supports the following additional constants for the **w** and **h** options:

main_w

main_h

The main input video's width and height

main_a

The same as $main_w / main_h$

main_sar

The main input video's sample aspect ratio

main_dar, mdar

The main input video's display aspect ratio. Calculated from "(main_w / main_h) * main_sar".

*main_hsub**main_vsub*

The main input video's horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

main_n

The (sequential) number of the main input frame, starting from 0. Only available with "eval=frame".

main_t

The presentation timestamp of the main input frame, expressed as a number of seconds. Only available with "eval=frame".

main_pos

The position (byte offset) of the frame in the main input stream, or NaN if this information is unavailable and/or meaningless (for example in case of synthetic video). Only available with "eval=frame".

Examples

- ⊕ Scale a subtitle stream (b) to match the main video (a) in size before overlaying

```
'scale2ref[b][a];[a][b]overlay'
```

- ⊕ Scale a logo to 1/10th the height of a video, while preserving its display aspect ratio.

```
[logo-in][video-in]scale2ref=w=oh*mdar:h=ih/10[logo-out][video-out]
```

Commands

This filter supports the following commands:

width, w

height, h

Set the output video dimension expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

scale2ref_npp

Use the NVIDIA Performance Primitives (libnpp) to scale (resize) the input video, based on a reference video.

See the **scale_npp** filter for available options, `scale2ref_npp` supports the same but uses the reference video instead of the main input as basis. `scale2ref_npp` also supports the following additional constants for the **w** and **h** options:

main_w

main_h

The main input video's width and height

main_a

The same as *main_w* / *main_h*

main_sar

The main input video's sample aspect ratio

main_dar, *mdar*

The main input video's display aspect ratio. Calculated from " $(main_w / main_h) * main_sar$ ".

main_n

The (sequential) number of the main input frame, starting from 0. Only available with "eval=frame".

main_t

The presentation timestamp of the main input frame, expressed as a number of seconds. Only available with "eval=frame".

main_pos

The position (byte offset) of the frame in the main input stream, or NaN if this information is unavailable and/or meaningless (for example in case of synthetic video). Only available with "eval=frame".

Examples

- ⊕ Scale a subtitle stream (b) to match the main video (a) in size before overlaying

```
'scale2ref_npp[b][a];[a][b]overlay_cuda'
```

- ⊕ Scale a logo to 1/10th the height of a video, while preserving its display aspect ratio.

```
[logo-in][video-in]scale2ref_npp=w=oh*mdar:h=ih/10[logo-out][video-out]
```

scharr

Apply scharr operator to input video stream.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

scale

Set value which will be multiplied with filtered result.

delta

Set value which will be added to filtered result.

Commands

This filter supports the all above options as **commands**.

scroll

Scroll input video horizontally and/or vertically by constant speed.

The filter accepts the following options:

horizontal, h

Set the horizontal scrolling speed. Default is 0. Allowed range is from -1 to 1. Negative values changes scrolling direction.

vertical, v

Set the vertical scrolling speed. Default is 0. Allowed range is from -1 to 1. Negative values changes scrolling direction.

hpos

Set the initial horizontal scrolling position. Default is 0. Allowed range is from 0 to 1.

vpos

Set the initial vertical scrolling position. Default is 0. Allowed range is from 0 to 1.

Commands

This filter supports the following **commands**:

horizontal, h

Set the horizontal scrolling speed.

vertical, v

Set the vertical scrolling speed.

scdet

Detect video scene change.

This filter sets frame metadata with `mafd` between frame, the scene score, and forward the frame to the next filter, so they can use these metadata to detect scene change or others.

In addition, this filter logs a message and sets frame metadata when it detects a scene change by **threshold**.

"lavfi.scd.mafd" metadata keys are set with `mafd` for every frame.

"lavfi.scd.score" metadata keys are set with scene change score for every frame to detect scene change.

"lavfi.scd.time" metadata keys are set with current filtered frame time which detect scene change with **threshold**.

The filter accepts the following options:

threshold, t

Set the scene change detection threshold as a percentage of maximum change. Good values are in the "[8.0, 14.0]" range. The range for **threshold** is "[0., 100.]".

Default value is 10..

sc_pass, s

Set the flag to pass scene change frames to the next filter. Default value is 0 You can enable it if you want to get snapshot of scene change frames only.

selectivecolor

Adjust cyan, magenta, yellow and black (CMYK) to certain ranges of colors (such as "reds", "yellows", "greens", "cyans", ...). The adjustment range is defined by the "purity" of the color (that is, how saturated it already is).

This filter is similar to the Adobe Photoshop Selective Color tool.

The filter accepts the following options:

correction_method

Select color correction method.

Available values are:

absolute

Specified adjustments are applied "as-is" (added/subtracted to original pixel component value).

relative

Specified adjustments are relative to the original component value.

Default is "absolute".

reds Adjustments for red pixels (pixels where the red component is the maximum)

yellows

Adjustments for yellow pixels (pixels where the blue component is the minimum)

greens

Adjustments for green pixels (pixels where the green component is the maximum)

cyans

Adjustments for cyan pixels (pixels where the red component is the minimum)

blues

Adjustments for blue pixels (pixels where the blue component is the maximum)

magentas

Adjustments for magenta pixels (pixels where the green component is the minimum)

whites

Adjustments for white pixels (pixels where all components are greater than 128)

neutrals

Adjustments for all pixels except pure black and pure white

blacks

Adjustments for black pixels (pixels where all components are lesser than 128)

psfile

Specify a Photoshop selective color file (".asv") to import the settings from.

All the adjustment settings (**reds**, **yellows**, ...) accept up to 4 space separated floating point adjustment values in the [-1,1] range, respectively to adjust the amount of cyan, magenta, yellow and black for the pixels of its range.

Examples

- ⊕ Increase cyan by 50% and reduce yellow by 33% in every green areas, and increase magenta by 27% in blue areas:

```
selectivecolor=greens=.5 0 -.33 0:blues=0 .27
```

- ⊕ Use a Photoshop selective color preset:

```
selectivecolor=psfile=MySelectiveColorPresets/Misty.asv
```

separatefields

The "separatefields" takes a frame-based video input and splits each frame into its components fields, producing a new half height clip with twice the frame rate and twice the frame count.

This filter use field-dominance information in frame to decide which of each pair of fields to place first in the output. If it gets it wrong use **setfield** filter before "separatefields" filter.

setdar, setsar

The "setdar" filter sets the Display Aspect Ratio for the filter output video.

This is done by changing the specified Sample (aka Pixel) Aspect Ratio, according to the following equation:

$$\langle \text{DAR} \rangle = \langle \text{HORIZONTAL_RESOLUTION} \rangle / \langle \text{VERTICAL_RESOLUTION} \rangle * \langle \text{SAR} \rangle$$

Keep in mind that the "setdar" filter does not modify the pixel dimensions of the video frame. Also, the display aspect ratio set by this filter may be changed by later filters in the filterchain, e.g. in case of scaling or if another "setdar" or a "setsar" filter is applied.

The "setsar" filter sets the Sample (aka Pixel) Aspect Ratio for the filter output video.

Note that as a consequence of the application of this filter, the output display aspect ratio will change according to the equation above.

Keep in mind that the sample aspect ratio set by the "setsar" filter may be changed by later filters in the filterchain, e.g. if another "setsar" or a "setdar" filter is applied.

It accepts the following parameters:

r, ratio, dar ("setdar" only), sar ("setsar" only)

Set the aspect ratio used by the filter.

The parameter can be a floating point number string, an expression, or a string of the form *num:den*, where *num* and *den* are the numerator and denominator of the aspect ratio. If the parameter is not specified, it is assumed the value "0". In case the form "*num:den*" is used, the ":" character should be escaped.

max Set the maximum integer value to use for expressing numerator and denominator when reducing the expressed aspect ratio to a rational. Default value is 100.

The parameter *sar* is an expression containing the following constants:

E, PI, PHI

These are approximated values for the mathematical constants e (Euler's number), pi (Greek pi), and phi (the golden ratio).

w, h The input width and height.

a These are the same as *w / h*.

sar The input sample aspect ratio.

dar The input display aspect ratio. It is the same as $(w / h) * sar$.

hsub, vsub

Horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

Examples

- ⊕ To change the display aspect ratio to 16:9, specify one of the following:

```
setdar=dar=1.77777
setdar=dar=16/9
```

- ⊕ To change the sample aspect ratio to 10:11, specify:

```
setsar=sar=10/11
```

- ⊕ To set a display aspect ratio of 16:9, and specify a maximum integer value of 1000 in the aspect ratio reduction, use the command:

```
setdar=ratio=16/9:max=1000
```

setfield

Force field for the output video frame.

The "setfield" filter marks the interlace type field for the output frames. It does not change the input frame, but only sets the corresponding property, which affects how the frame is treated by following filters (e.g. "fieldorder" or "yadif").

The filter accepts the following options:

mode

Available values are:

auto Keep the same field property.

bff Mark the frame as bottom-field-first.

tff Mark the frame as top-field-first.

prog

Mark the frame as progressive.

setparams

Force frame parameter for the output video frame.

The "setparams" filter marks interlace and color range for the output frames. It does not change the input frame, but only sets the corresponding property, which affects how the frame is treated by filters/encoders.

field_mode

Available values are:

auto Keep the same field property (default).

bff Mark the frame as bottom-field-first.

tff Mark the frame as top-field-first.

prog

Mark the frame as progressive.

range

Available values are:

auto Keep the same color range property (default).

unspecified, unknown

Mark the frame as unspecified color range.

limited, tv, mpeg

Mark the frame as limited range.

full, pc, jpeg

Mark the frame as full range.

color primaries

Set the color primaries. Available values are:

auto Keep the same color primaries property (default).

bt709

unknown

bt470m

bt470bg

smpte170m

smpte240m

film

bt2020

smpte428

smpte431

smpte432

jedec-p22

color_trc

Set the color transfer. Available values are:

auto Keep the same color trc property (default).

bt709

unknown

bt470m

bt470bg

smpte170m

smpte240m

linear

log100

log316

iec61966-2-4

bt1361e

iec61966-2-1

bt2020-10

bt2020-12

smpte2084

smpte428

arib-std-b67

colorspace

Set the colorspace. Available values are:

auto Keep the same colorspace property (default).

gbr
bt709
unknown
fcc
bt470bg
smpte170m
smpte240m
ycgco
bt2020nc
bt2020c
smpte2085
chroma-derived-nc
chroma-derived-c
ictcp

sharpen_npp

Use the NVIDIA Performance Primitives (libnpp) to perform image sharpening with border control.

The following additional options are accepted:

border_type

Type of sampling to be used ad frame borders. One of the following:

replicate

Replicate pixel values.

shear

Apply shear transform to input video.

This filter supports the following options:

shx Shear factor in X-direction. Default value is 0. Allowed range is from -2 to 2.

shy Shear factor in Y-direction. Default value is 0. Allowed range is from -2 to 2.

fillcolor, c

Set the color used to fill the output area not covered by the transformed video. For the general syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. If the special value "none" is selected then no background is printed (useful for example if the background is never shown).

Default value is "black".

interp

Set interpolation type. Can be "bilinear" or "nearest". Default is "bilinear".

Commands

This filter supports the all above options as **commands**.

showinfo

Show a line containing various information for each input video frame. The input video is not modified.

This filter supports the following options:

checksum

Calculate checksums of each plane. By default enabled.

The shown line contains a sequence of key/value pairs of the form *key:value*.

The following values are shown in the output:

- n** The (sequential) number of the input frame, starting from 0.
- pts** The Presentation TimeStamp of the input frame, expressed as a number of time base units. The time base unit depends on the filter input pad.
- pts_time**
The Presentation TimeStamp of the input frame, expressed as a number of seconds.
- pos** The position of the frame in the input stream, or -1 if this information is unavailable and/or meaningless (for example in case of synthetic video).
- fmt** The pixel format name.
- sar** The sample aspect ratio of the input frame, expressed in the form *num/den*.
- s** The size of the input frame. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

i The type of interlaced mode ("P" for "progressive", "T" for top field first, "B" for bottom field first).

iskey

This is 1 if the frame is a key frame, 0 otherwise.

type The picture type of the input frame ("I" for an I-frame, "P" for a P-frame, "B" for a B-frame, or "?" for an unknown type). Also refer to the documentation of the "AVPictureType" enum and of the "av_get_picture_type_char" function defined in *libavutil/avutil.h*.

checksum

The Adler-32 checksum (printed in hexadecimal) of all the planes of the input frame.

plane_checksum

The Adler-32 checksum (printed in hexadecimal) of each plane of the input frame, expressed in the form "[*c0 c1 c2 c3*]".

mean

The mean value of pixels in each plane of the input frame, expressed in the form "[*mean0 mean1 mean2 mean3*]".

stdev

The standard deviation of pixel values in each plane of the input frame, expressed in the form "[*stdev0 stdev1 stdev2 stdev3*]".

showpalette

Displays the 256 colors palette of each frame. This filter is only relevant for *pal8* pixel format frames.

It accepts the following option:

s Set the size of the box used to represent one palette color entry. Default is 30 (for a "30x30" pixel box).

shuffleframes

Reorder and/or duplicate and/or drop video frames.

It accepts the following parameters:

mapping

Set the destination indexes of input frames. This is space or '|' separated list of indexes that maps

input frames to output frames. Number of indexes also sets maximal value that each index may have. '-1' index have special meaning and that is to drop frame.

The first frame has the index 0. The default is to keep the input unchanged.

Examples

- ⊕ Swap second and third frame of every three frames of the input:

```
ffmpeg -i INPUT -vf "shuffleframes=0 2 1" OUTPUT
```

- ⊕ Swap 10th and 1st frame of every ten frames of the input:

```
ffmpeg -i INPUT -vf "shuffleframes=9 1 2 3 4 5 6 7 8 0" OUTPUT
```

shufflepixels

Reorder pixels in video frames.

This filter accepts the following options:

direction, d

Set shuffle direction. Can be forward or inverse direction. Default direction is forward.

mode, m

Set shuffle mode. Can be horizontal, vertical or block mode.

width, w

height, h

Set shuffle block_size. In case of horizontal shuffle mode only width part of size is used, and in case of vertical shuffle mode only height part of size is used.

seed, s

Set random seed used with shuffling pixels. Mainly useful to set to be able to reverse filtering process to get original input. For example, to reverse forward shuffle you need to use same parameters and exact same seed and to set direction to inverse.

shuffleplanes

Reorder and/or duplicate video planes.

It accepts the following parameters:

map0

The index of the input plane to be used as the first output plane.

map1

The index of the input plane to be used as the second output plane.

map2

The index of the input plane to be used as the third output plane.

map3

The index of the input plane to be used as the fourth output plane.

The first plane has the index 0. The default is to keep the input unchanged.

Examples

- ⊕ Swap the second and third planes of the input:

```
ffmpeg -i INPUT -vf shuffleplanes=0:2:1:3 OUTPUT
```

signalstats

Evaluate various visual metrics that assist in determining issues associated with the digitization of analog video media.

By default the filter will log these metadata values:

YMIN

Display the minimal Y value contained within the input frame. Expressed in range of [0-255].

YLOW

Display the Y value at the 10% percentile within the input frame. Expressed in range of [0-255].

YAVG

Display the average Y value within the input frame. Expressed in range of [0-255].

YHIGH

Display the Y value at the 90% percentile within the input frame. Expressed in range of [0-255].

YMAX

Display the maximum Y value contained within the input frame. Expressed in range of [0-255].

UMIN

Display the minimal U value contained within the input frame. Expressed in range of [0-255].

ULOW

Display the U value at the 10% percentile within the input frame. Expressed in range of [0-255].

UAVG

Display the average U value within the input frame. Expressed in range of [0-255].

UHIGH

Display the U value at the 90% percentile within the input frame. Expressed in range of [0-255].

UMAX

Display the maximum U value contained within the input frame. Expressed in range of [0-255].

VMIN

Display the minimal V value contained within the input frame. Expressed in range of [0-255].

VLOW

Display the V value at the 10% percentile within the input frame. Expressed in range of [0-255].

VAVG

Display the average V value within the input frame. Expressed in range of [0-255].

VHIGH

Display the V value at the 90% percentile within the input frame. Expressed in range of [0-255].

VMAX

Display the maximum V value contained within the input frame. Expressed in range of [0-255].

SATMIN

Display the minimal saturation value contained within the input frame. Expressed in range of [0~~181.02].

SATLOW

Display the saturation value at the 10% percentile within the input frame. Expressed in range of [0~~181.02].

SATAVG

Display the average saturation value within the input frame. Expressed in range of [0~~181.02].

SATHIGH

Display the saturation value at the 90% percentile within the input frame. Expressed in range of [0~181.02].

SATMAX

Display the maximum saturation value contained within the input frame. Expressed in range of [0~181.02].

HUEMED

Display the median value for hue within the input frame. Expressed in range of [0-360].

HUEAVG

Display the average value for hue within the input frame. Expressed in range of [0-360].

YDIF

Display the average of sample value difference between all values of the Y plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

UDIF

Display the average of sample value difference between all values of the U plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

VDIF

Display the average of sample value difference between all values of the V plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

YBITDEPTH

Display bit depth of Y plane in current frame. Expressed in range of [0-16].

UBITDEPTH

Display bit depth of U plane in current frame. Expressed in range of [0-16].

VBITDEPTH

Display bit depth of V plane in current frame. Expressed in range of [0-16].

The filter accepts the following options:

stat

out stat specify an additional form of image analysis. **out** output video with the specified type of pixel highlighted.

Both options accept the following values:

tout Identify *temporal outliers* pixels. A *temporal outlier* is a pixel unlike the neighboring pixels of the same field. Examples of temporal outliers include the results of video dropouts, head clogs, or tape tracking issues.

vrep

Identify *vertical line repetition*. Vertical line repetition includes similar rows of pixels within a frame. In born-digital video vertical line repetition is common, but this pattern is uncommon in video digitized from an analog source. When it occurs in video that results from the digitization of an analog source it can indicate concealment from a dropout compensator.

brng

Identify pixels that fall outside of legal broadcast range.

color, c

Set the highlight color for the **out** option. The default color is yellow.

Examples

- ⊕ Output data of various video metrics:

```
ffprobe -f lavfi movie=example.mov,signalstats="stat=tout+vrep+brng" -show_frames
```

- ⊕ Output specific data about the minimum and maximum values of the Y plane per frame:

```
ffprobe -f lavfi movie=example.mov,signalstats -show_entries frame_tags=lavfi.signalstats.YMAX,lavfi.sig
```

- ⊕ Playback video while highlighting pixels that are outside of broadcast range in red.

```
ffplay example.mov -vf signalstats="out=brng:color=red"
```

- ⊕ Playback video with signalstats metadata drawn over the frame.

```
ffplay example.mov -vf signalstats=stat=brng+vrep+tout,drawtext=fontfile=FreeSerif.ttf:textfile=signalstat_d
```

The contents of signalstat_drawtext.txt used in the command are:

```
time % {pts:hms }
```

```
Y (% {metadata:lavfi.signalstats.YMIN}-% {metadata:lavfi.signalstats.YMAX})  
U (% {metadata:lavfi.signalstats.UMIN}-% {metadata:lavfi.signalstats.UMAX})  
V (% {metadata:lavfi.signalstats.VMIN}-% {metadata:lavfi.signalstats.VMAX})  
saturation maximum: % {metadata:lavfi.signalstats.SATMAX}
```

signature

Calculates the MPEG-7 Video Signature. The filter can handle more than one input. In this case the matching between the inputs can be calculated additionally. The filter always passes through the first input. The signature of each stream can be written into a file.

It accepts the following options:

detectmode

Enable or disable the matching process.

Available values are:

off Disable the calculation of a matching (default).

full Calculate the matching for the whole video and output whether the whole video matches or only parts.

fast Calculate only until a matching is found or the video ends. Should be faster in some cases.

nb_inputs

Set the number of inputs. The option value must be a non negative integer. Default value is 1.

filename

Set the path to which the output is written. If there is more than one input, the path must be a prototype, i.e. must contain %d or %0nd (where n is a positive integer), that will be replaced with the input number. If no filename is specified, no output will be written. This is the default.

format

Choose the output format.

Available values are:

binary

Use the specified binary representation (default).

xml Use the specified xml representation.

th_d

Set threshold to detect one word as similar. The option value must be an integer greater than zero. The default value is 9000.

th_dc

Set threshold to detect all words as similar. The option value must be an integer greater than zero. The default value is 60000.

th_xh

Set threshold to detect frames as similar. The option value must be an integer greater than zero. The default value is 116.

th_di

Set the minimum length of a sequence in frames to recognize it as matching sequence. The option value must be a non negative integer value. The default value is 0.

th_it

Set the minimum relation, that matching frames to all frames must have. The option value must be a double value between 0 and 1. The default value is 0.5.

Examples

- ⊕ To calculate the signature of an input video and store it in signature.bin:

```
ffmpeg -i input.mkv -vf signature=filename=signature.bin -map 0:v -f null -
```

- ⊕ To detect whether two videos match and store the signatures in XML format in signature0.xml and signature1.xml:

```
ffmpeg -i input1.mkv -i input2.mkv -filter_complex "[0:v][1:v] signature=nb_inputs=2:detectmode=full:form
```

siti

Calculate Spatial Info (SI) and Temporal Info (TI) scores for a video, as defined in ITU-T P.910: Subjective video quality assessment methods for multimedia applications. Available PDF at <https://www.itu.int/rec/T-REC-P.910-199909-S/en> >.

It accepts the following option:

print_summary

If set to 1, Summary statistics will be printed to the console. Default 0.

Examples

- ⊕ To calculate SI/TI metrics and print summary:

```
ffmpeg -i input.mp4 -vf siti=print_summary=1 -f null -
```

smartblur

Blur the input video without impacting the outlines.

It accepts the following options:

luma_radius, lr

Set the luma radius. The option value must be a float number in the range [0.1,5.0] that specifies the variance of the gaussian filter used to blur the image (slower if larger). Default value is 1.0.

luma_strength, ls

Set the luma strength. The option value must be a float number in the range [-1.0,1.0] that configures the blurring. A value included in [0.0,1.0] will blur the image whereas a value included in [-1.0,0.0] will sharpen the image. Default value is 1.0.

luma_threshold, lt

Set the luma threshold used as a coefficient to determine whether a pixel should be blurred or not. The option value must be an integer in the range [-30,30]. A value of 0 will filter all the image, a value included in [0,30] will filter flat areas and a value included in [-30,0] will filter edges. Default value is 0.

chroma_radius, cr

Set the chroma radius. The option value must be a float number in the range [0.1,5.0] that specifies the variance of the gaussian filter used to blur the image (slower if larger). Default value is **luma_radius**.

chroma_strength, cs

Set the chroma strength. The option value must be a float number in the range [-1.0,1.0] that configures the blurring. A value included in [0.0,1.0] will blur the image whereas a value included in [-1.0,0.0] will sharpen the image. Default value is **luma_strength**.

chroma_threshold, ct

Set the chroma threshold used as a coefficient to determine whether a pixel should be blurred or not. The option value must be an integer in the range [-30,30]. A value of 0 will filter all the image, a value included in [0,30] will filter flat areas and a value included in [-30,0] will filter edges. Default value is **luma_threshold**.

If a chroma option is not explicitly set, the corresponding luma value is set.

sobel

Apply sobel operator to input video stream.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

scale

Set value which will be multiplied with filtered result.

delta

Set value which will be added to filtered result.

Commands

This filter supports the all above options as **commands**.

spp

Apply a simple postprocessing filter that compresses and decompresses the image at several (or - in the case of **quality** level 6 - all) shifts and average the results.

The filter accepts the following options:

quality

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 0-6. If set to 0, the filter will have no effect. A value of 6 means the higher quality. For each increment of that value the speed drops by a factor of approximately 2. Default value is 3.

qp Force a constant quantization parameter. If not set, the filter will use the QP from the video stream (if available).

mode

Set thresholding mode. Available modes are:

hard

Set hard thresholding (default).

soft Set soft thresholding (better de-ringing effect, but likely blurrier).

use_bframe_qp

Enable the use of the QP from the B-Frames if set to 1. Using this option may cause flicker since the B-Frames have often larger QP. Default is 0 (not enabled).

Commands

This filter supports the following commands:

quality, level

Set quality level. The value "max" can be used to set the maximum level, currently 6.

sr

Scale the input by applying one of the super-resolution methods based on convolutional neural networks. Supported models:

- ⊕ Super-Resolution Convolutional Neural Network model (SRCNN). See <https://arxiv.org/abs/1501.00092>.
- ⊕ Efficient Sub-Pixel Convolutional Neural Network model (ESPCN). See <https://arxiv.org/abs/1609.05158>.

Training scripts as well as scripts for model file (.pb) saving can be found at https://github.com/XuweiMeng/sr/tree/sr_dnn_native. Original repository is at <https://github.com/HighVoltageRocknRoll/sr.git>.

Native model files (.model) can be generated from TensorFlow model files (.pb) by using `tools/python/convert.py`

The filter accepts the following options:

dnn_backend

Specify which DNN backend to use for model loading and execution. This option accepts the

following values:

native

Native implementation of DNN loading and execution.

tensorflow

TensorFlow backend. To enable this backend you need to install the TensorFlow for C library (see <https://www.tensorflow.org/install/lang_c>) and configure FFmpeg with "--enable-libtensorflow"

Default value is **native**.

model

Set path to model file specifying network architecture and its parameters. Note that different backends use different file formats. TensorFlow backend can load files for both formats, while native backend can load files for only its format.

scale_factor

Set scale factor for SRCNN model. Allowed values are 2, 3 and 4. Default value is 2. Scale factor is necessary for SRCNN model, because it accepts input upscaled using bicubic upscaling with proper scale factor.

To get full functionality (such as async execution), please use the **dnn_processing** filter.

ssim

Obtain the SSIM (Structural Similarity Metric) between two input videos.

This filter takes in input two input videos, the first input is considered the "main" source and is passed unchanged to the output. The second input is used as a "reference" video for computing the SSIM.

Both video inputs must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The filter stores the calculated SSIM of each frame.

The description of the accepted parameters follows.

stats_file, f

If specified the filter will use the named file to save the SSIM of each individual frame. When filename equals "-" the data is sent to standard output.

The file printed if *stats_file* is selected, contains a sequence of key/value pairs of the form *key:value* for each compared couple of frames.

A description of each shown parameter follows:

n sequential number of the input frame, starting from 1

Y, U, V, R, G, B

SSIM of the compared frames for the component specified by the suffix.

All SSIM of the compared frames for the whole frame.

dB Same as above but in dB representation.

This filter also supports the **framesync** options.

Examples

⊕ For example:

```
movie=ref_movie.mpg, setpts=PTS-STARTPTS [main];
[main][ref] ssim="stats_file=stats.log" [out]
```

On this example the input file being processed is compared with the reference file *ref_movie.mpg*. The SSIM of each individual frame is stored in *stats.log*.

⊕ Another example with both psnr and ssim at same time:

```
ffmpeg -i main.mpg -i ref.mpg -lavfi "ssim;[0:v][1:v]psnr" -f null -
```

⊕ Another example with different containers:

```
ffmpeg -i main.mpg -i ref.mkv -lavfi "[0:v]settb=AVTB,setpts=PTS-STARTPTS[main];[1:v]settb=AVTB,se
```

stereo3d

Convert between different stereoscopic image formats.

The filters accept the following options:

in Set stereoscopic image format of input.

Available values for input image formats are:

sbsl side by side parallel (left eye left, right eye right)

sbsr side by side crosseye (right eye left, left eye right)

sbs2l

side by side parallel with half width resolution (left eye left, right eye right)

sbs2r

side by side crosseye with half width resolution (right eye left, left eye right)

abl

tbl above-below (left eye above, right eye below)

abr

tbr above-below (right eye above, left eye below)

ab2l

tb2l above-below with half height resolution (left eye above, right eye below)

ab2r

tb2r above-below with half height resolution (right eye above, left eye below)

al alternating frames (left eye first, right eye second)

ar alternating frames (right eye first, left eye second)

irl interleaved rows (left eye has top row, right eye starts on next row)

irr interleaved rows (right eye has top row, left eye starts on next row)

icl interleaved columns, left eye first

icr interleaved columns, right eye first

Default value is **sbsl**.

out Set stereoscopic image format of output.

sbsl side by side parallel (left eye left, right eye right)

sbsr side by side crosseye (right eye left, left eye right)

sbs2l

side by side parallel with half width resolution (left eye left, right eye right)

sbs2r

side by side crosseye with half width resolution (right eye left, left eye right)

abl

tbl above-below (left eye above, right eye below)

abr

tbr above-below (right eye above, left eye below)

ab2l

tb2l above-below with half height resolution (left eye above, right eye below)

ab2r

tb2r above-below with half height resolution (right eye above, left eye below)

al alternating frames (left eye first, right eye second)

ar alternating frames (right eye first, left eye second)

irl interleaved rows (left eye has top row, right eye starts on next row)

irr interleaved rows (right eye has top row, left eye starts on next row)

arbg

anaglyph red/blue gray (red filter on left eye, blue filter on right eye)

argg

anaglyph red/green gray (red filter on left eye, green filter on right eye)

arcg anaglyph red/cyan gray (red filter on left eye, cyan filter on right eye)

arch anaglyph red/cyan half colored (red filter on left eye, cyan filter on right eye)

arcc anaglyph red/cyan color (red filter on left eye, cyan filter on right eye)

arcd anaglyph red/cyan color optimized with the least squares projection of dubois (red filter on left eye, cyan filter on right eye)

agmg

anaglyph green/magenta gray (green filter on left eye, magenta filter on right eye)

agmh

anaglyph green/magenta half colored (green filter on left eye, magenta filter on right eye)

agmc

anaglyph green/magenta colored (green filter on left eye, magenta filter on right eye)

agmd

anaglyph green/magenta color optimized with the least squares projection of dubois (green filter on left eye, magenta filter on right eye)

aybg

anaglyph yellow/blue gray (yellow filter on left eye, blue filter on right eye)

aybh

anaglyph yellow/blue half colored (yellow filter on left eye, blue filter on right eye)

aybc

anaglyph yellow/blue colored (yellow filter on left eye, blue filter on right eye)

aybd

anaglyph yellow/blue color optimized with the least squares projection of dubois (yellow filter on left eye, blue filter on right eye)

ml mono output (left eye only)

mr mono output (right eye only)

chl checkerboard, left eye first

chr checkerboard, right eye first

icl interleaved columns, left eye first

icr interleaved columns, right eye first

hdmi

HDMI frame pack

Default value is **arcd**.

Examples

- ⊕ Convert input video from side by side parallel to anaglyph yellow/blue dubs:

```
stereo3d=sbsl:aybd
```

- ⊕ Convert input video from above below (left eye above, right eye below) to side by side crosseye.

```
stereo3d=abl:sbsr
```

streamselect, astreamselect

Select video or audio streams.

The filter accepts the following options:

inputs

Set number of inputs. Default is 2.

map Set input indexes to remap to outputs.

Commands

The "streamselect" and "astreamselect" filter supports the following commands:

map Set input indexes to remap to outputs.

Examples

- ⊕ Select first 5 seconds 1st stream and rest of time 2nd stream:

```
sendcmd='5.0 streamselect map 1',streamselect=inputs=2:map=0
```

- ⊕ Same as above, but for audio:

```
asendcmd='5.0 astreamselect map 1',astreamselect=inputs=2:map=0
```

subtitles

Draw subtitles on top of input video using the libass library.

To enable compilation of this filter you need to configure FFmpeg with "--enable-libass". This filter also requires a build with libavcodec and libavformat to convert the passed subtitles file to ASS (Advanced Substation Alpha) subtitles format.

The filter accepts the following options:

filename, f

Set the filename of the subtitle file to read. It must be specified.

original_size

Specify the size of the original video, the video for which the ASS file was composed. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Due to a misdesign in ASS aspect ratio arithmetic, this is necessary to correctly scale the fonts if the aspect ratio has been changed.

fontsdir

Set a directory path containing fonts that can be used by the filter. These fonts will be used in addition to whatever the font provider uses.

alpha

Process alpha channel, by default alpha channel is untouched.

charenc

Set subtitles input character encoding. "subtitles" filter only. Only useful if not UTF-8.

stream_index, si

Set subtitles stream index. "subtitles" filter only.

force_style

Override default style or script info parameters of the subtitles. It accepts a string containing ASS style format "KEY=VALUE" couples separated by ",".

If the first key is not specified, it is assumed that the first value specifies the **filename**.

For example, to render the file *sub.srt* on top of the input video, use the command:

```
subtitles=sub.srt
```

which is equivalent to:

```
subtitles=filename=sub.srt
```

To render the default subtitles stream from file *video.mkv*, use:

```
subtitles=video.mkv
```

To render the second subtitles stream from that file, use:

```
subtitles=video.mkv:si=1
```

To make the subtitles stream from *sub.srt* appear in 80% transparent blue "DejaVu Serif", use:

```
subtitles=sub.srt:force_style='Fontname=DejaVu Serif,PrimaryColour=&HCCFF0000'
```

super2xsai

Scale the input by 2x and smooth using the Super2xSaI (Scale and Interpolate) pixel art scaling algorithm.

Useful for enlarging pixel art images without reducing sharpness.

swaprect

Swap two rectangular objects in video.

This filter accepts the following options:

w Set object width.

h Set object height.

x1 Set 1st rect x coordinate.

y1 Set 1st rect y coordinate.

x2 Set 2nd rect x coordinate.

y2 Set 2nd rect y coordinate.

All expressions are evaluated once for each frame.

The all options are expressions containing the following constants:

w

h The input width and height.

a same as w / h

sar input sample aspect ratio

dar input display aspect ratio, it is the same as $(w / h) * sar$

n The number of the input frame, starting from 0.

t The timestamp expressed in seconds. It's NAN if the input timestamp is unknown.

pos the position in the file of the input frame, NAN if unknown

Commands

This filter supports the all above options as **commands**.

swapuv

Swap U & V plane.

tblend

Blend successive video frames.

See **blend**

telecine

Apply telecine process to the video.

This filter accepts the following options:

first_field

top, t

top field first

bottom, b

bottom field first The default value is "top".

pattern

A string of numbers representing the pulldown pattern you wish to apply. The default value is 23.

Some typical patterns:

NTSC output (30i):

27.5p: 32222

24p: 23 (classic)

24p: 2332 (preferred)

20p: 33

18p: 334

16p: 3444

PAL output (25i):

27.5p: 12222

24p: 222222222223 ("Euro pulldown")

16.67p: 33

16p: 33333334

histogram

Compute and draw a color distribution histogram for the input video across time.

Unlike **histogram** video filter which only shows histogram of single input frame at certain time, this filter shows also past histograms of number of frames defined by "width" option.

The computed histogram is a representation of the color component distribution in an image.

The filter accepts the following options:

width, w

Set width of single color component output. Default value is 0. Value of 0 means width will be picked from input video. This also set number of passed histograms to keep. Allowed range is [0, 8192].

display_mode, d

Set display mode. It accepts the following values:

stack

Per color component graphs are placed below each other.

parade

Per color component graphs are placed side by side.

overlay

Presents information identical to that in the "parade", except that the graphs representing color components are superimposed directly over one another.

Default is "stack".

levels_mode, m

Set mode. Can be either "linear", or "logarithmic". Default is "linear".

components, c

Set what color components to display. Default is 7.

bgopacity, b

Set background opacity. Default is 0.9.

envelope, e

Show envelope. Default is disabled.

ecolor, ec

Set envelope color. Default is "gold".

slide

Set slide mode.

Available values for slide is:

frame

Draw new frame when right border is reached.

replace

Replace old columns with new ones.

scroll

Scroll from right to left.

rscroll

Scroll from left to right.

picture

Draw single picture.

Default is "replace".

threshold

Apply threshold effect to video stream.

This filter needs four video streams to perform thresholding. First stream is stream we are filtering. Second stream is holding threshold values, third stream is holding min values, and last, fourth stream is holding max values.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

For example if first stream pixel's component value is less then threshold value of pixel component from 2nd threshold stream, third stream value will be picked, otherwise fourth stream pixel component value will be picked.

Using color source filter one can perform various types of thresholding:

Commands

This filter supports the all options as **commands**.

Examples

- ⊕ Binary threshold, using gray color as threshold:

```
ffmpeg -i 320x240.avi -f lavfi -i color=gray -f lavfi -i color=black -f lavfi -i color=white -lavfi threshold outp
```

- ⊕ Inverted binary threshold, using gray color as threshold:

```
ffmpeg -i 320x240.avi -f lavfi -i color=gray -f lavfi -i color=white -f lavfi -i color=black -lavfi threshold outp
```

- ⊕ Truncate binary threshold, using gray color as threshold:

```
ffmpeg -i 320x240.avi -f lavfi -i color=gray -i 320x240.avi -f lavfi -i color=gray -lavfi threshold output.avi
```

- ⊕ Threshold to zero, using gray color as threshold:

```
ffmpeg -i 320x240.avi -f lavfi -i color=gray -f lavfi -i color=white -i 320x240.avi -lavfi threshold output.avi
```

- ⊕ Inverted threshold to zero, using gray color as threshold:

```
ffmpeg -i 320x240.avi -f lavfi -i color=gray -i 320x240.avi -f lavfi -i color=white -lavfi threshold output.avi
```

thumbnail

Select the most representative frame in a given sequence of consecutive frames.

The filter accepts the following options:

- n** Set the frames batch size to analyze; in a set of n frames, the filter will pick one of them, and then handle the next batch of n frames until the end. Default is 100.

- log** Set the log level to display picked frame stats. Default is "info".

Since the filter keeps track of the whole frames sequence, a bigger n value will result in a higher memory usage, so a high value is not recommended.

Examples

- ⊕ Extract one picture each 50 frames:

```
thumbnail=50
```

- ⊕ Complete example of a thumbnail creation with **ffmpeg**:

```
ffmpeg -i in.avi -vf thumbnail,scale=300:200 -frames:v 1 out.png
```

tile

Tile several successive frames together.

The **untile** filter can do the reverse.

The filter accepts the following options:

layout

Set the grid size in the form "COLUMNSxROWS". Range is upto `UINT_MAX` cells. Default is "6x5".

nb_frames

Set the maximum number of frames to render in the given area. It must be less than or equal to *wxh*. The default value is 0, meaning all the area will be used.

margin

Set the outer border margin in pixels. Range is 0 to 1024. Default is 0.

padding

Set the inner border thickness (i.e. the number of pixels between frames). For more advanced padding options (such as having different values for the edges), refer to the `pad` video filter. Range is 0 to 1024. Default is 0.

color

Specify the color of the unused area. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. The default value of *color* is "black".

overlap

Set the number of frames to overlap when tiling several successive frames together. The value must be between 0 and *nb_frames - 1*. Default is 0.

init_padding

Set the number of frames to initially be empty before displaying first output frame. This controls how soon will one get first output frame. The value must be between 0 and *nb_frames - 1*. Default is 0.

Examples

- ⊕ Produce 8x8 PNG tiles of all keyframes (**-skip_frame nokey**) in a movie:

```
ffmpeg -skip_frame nokey -i file.avi -vf 'scale=128:72,tile=8x8' -an -vsync 0 keyframes%03d.png
```

The **-vsync 0** is necessary to prevent **ffmpeg** from duplicating each output frame to accommodate the originally detected frame rate.

- ⊕ Display 5 pictures in an area of "3x2" frames, with 7 pixels between them, and 2 pixels of initial margin, using mixed flat and named options:

```
tile=3x2:nb_frames=5:padding=7:margin=2
```

tinterlace

Perform various types of temporal field interlacing.

Frames are counted starting from 1, so the first input frame is considered odd.

The filter accepts the following options:

mode

Specify the mode of the interlacing. This option can also be specified as a value alone. See below for a list of values for this option.

Available values are:

merge, 0

Move odd frames into the upper field, even into the lower field, generating a double height frame at half frame rate.

```

-----> time
Input:
Frame 1   Frame 2   Frame 3   Frame 4

11111     22222     33333     44444
11111     22222     33333     44444
11111     22222     33333     44444
11111     22222     33333     44444

Output:
11111           33333
22222           44444
11111           33333
22222           44444
11111           33333
22222           44444
11111           33333
22222           44444

```

drop_even, 1

Only output odd frames, even frames are dropped, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1    Frame 2    Frame 3    Frame 4

11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444

Output:
11111              33333
11111              33333
11111              33333
11111              33333

```

drop_odd, 2

Only output even frames, odd frames are dropped, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1    Frame 2    Frame 3    Frame 4

11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444

Output:
      22222              44444
      22222              44444
      22222              44444
      22222              44444

```

pad, 3

Expand each frame to full height, but pad alternate lines with black, generating a frame with

double height at the same input frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444

Output:
11111      .....      33333      .....
.....      22222      .....      44444
11111      .....      33333      .....
.....      22222      .....      44444
11111      .....      33333      .....
.....      22222      .....      44444
11111      .....      33333      .....
.....      22222      .....      44444

```

interleave_top, 4

Interleave the upper field from odd frames with the lower field from even frames, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111<-      22222      33333<-      44444
11111      22222<-      33333      44444<-
11111<-      22222      33333<-      44444
11111      22222<-      33333      44444<-

Output:
11111      33333
22222      44444
11111      33333
22222      44444

```

interleave_bottom, 5

Interleave the lower field from odd frames with the upper field from even frames, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111      22222<-   33333      44444<-
11111<-    22222      33333<-    44444
11111      22222<-   33333      44444<-
11111<-    22222      33333<-    44444

Output:
22222              44444
11111              33333
22222              44444
11111              33333

```

interlaced2, 6

Double frame rate with unchanged height. Frames are inserted each containing the second temporal field from the previous input frame and the first temporal field from the next input frame. This mode relies on the `top_field_first` flag. Useful for interlaced video displays with no field synchronisation.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444

Output:
11111 22222 22222 33333 33333 44444 44444
11111 11111 22222 22222 33333 33333 44444
11111 22222 22222 33333 33333 44444 44444
11111 11111 22222 22222 33333 33333 44444

```


mergex2, 7

Move odd frames into the upper field, even into the lower field, generating a double height frame at same frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444
11111      22222      33333      44444

Output:
11111      33333      33333      55555
22222      22222      44444      44444
11111      33333      33333      55555
22222      22222      44444      44444
11111      33333      33333      55555
22222      22222      44444      44444
11111      33333      33333      55555
22222      22222      44444      44444

```

Numeric values are deprecated but are accepted for backward compatibility reasons.

Default mode is "merge".

flags

Specify flags influencing the filter process.

Available value for *flags* is:

low_pass_filter, vlpf

Enable linear vertical low-pass filtering in the filter. Vertical low-pass filtering is required when creating an interlaced destination from a progressive source which contains high-frequency vertical detail. Filtering will reduce interlace 'twitter' and Moire patterning.

complex_filter, cvlpf

Enable complex vertical low-pass filtering. This will slightly less reduce interlace 'twitter' and Moire patterning but better retain detail and subjective sharpness impression.

bypass_il

Bypass already interlaced frames, only adjust the frame rate.

Vertical low-pass filtering and bypassing already interlaced frames can only be enabled for **mode** *interleave_top* and *interleave_bottom*.

tmedian

Pick median pixels from several successive input video frames.

The filter accepts the following options:

radius

Set radius of median filter. Default is 1. Allowed range is from 1 to 127.

planes

Set which planes to filter. Default value is 15, by which all planes are processed.

percentile

Set median percentile. Default value is 0.5. Default value of 0.5 will pick always median values, while 0 will pick minimum values, and 1 maximum values.

Commands

This filter supports all above options as **commands**, excluding option "radius".

tmidequalizer

Apply Temporal Midway Video Equalization effect.

Midway Video Equalization adjusts a sequence of video frames to have the same histograms, while maintaining their dynamics as much as possible. It's useful for e.g. matching exposures from a video frames sequence.

This filter accepts the following option:

radius

Set filtering radius. Default is 5. Allowed range is from 1 to 127.

sigma

Set filtering sigma. Default is 0.5. This controls strength of filtering. Setting this option to 0 effectively does nothing.

planes

Set which planes to process. Default is 15, which is all available planes.

tmix

Mix successive video frames.

A description of the accepted options follows.

frames

The number of successive frames to mix. If unspecified, it defaults to 3.

weights

Specify weight of each input video frame. Each weight is separated by space. If number of weights is smaller than number of *frames* last specified weight will be used for all remaining unset weights.

scale

Specify scale, if it is set it will be multiplied with sum of each weight multiplied with pixel values to give final destination pixel value. By default *scale* is auto scaled to sum of weights.

planes

Set which planes to filter. Default is all. Allowed range is from 0 to 15.

Examples

- ⊕ Average 7 successive frames:

```
tmix=frames=7:weights="1 1 1 1 1 1 1"
```

- ⊕ Apply simple temporal convolution:

```
tmix=frames=3:weights="-1 3 -1"
```

- ⊕ Similar as above but only showing temporal differences:

```
tmix=frames=3:weights="-1 2 -1":scale=1
```

Commands

This filter supports the following commands:

weights**scale****planes**

Syntax is same as option with same name.

tonemap

Tone map colors from different dynamic ranges.

This filter expects data in single precision floating point, as it needs to operate on (and can output) out-of-range values. Another filter, such as **zscale**, is needed to convert the resulting frame to a usable format.

The tonemapping algorithms implemented only work on linear light, so input data should be linearized beforehand (and possibly correctly tagged).

```
ffmpeg -i INPUT -vf zscale=transfer=linear,tonemap=clip,zscale=transfer=bt709,format=yuv420p OUTPUT
```

Options

The filter accepts the following options.

tonemap

Set the tone map algorithm to use.

Possible values are:

none

Do not apply any tone map, only desaturate overbright pixels.

clip Hard-clip any out-of-range values. Use it for perfect color accuracy for in-range values, while distorting out-of-range values.

linear

Stretch the entire reference gamut to a linear multiple of the display.

gamma

Fit a logarithmic transfer between the tone curves.

reinhard

Preserve overall image brightness with a simple curve, using nonlinear contrast, which

results in flattening details and degrading color accuracy.

hable

Preserve both dark and bright details better than *reinhard*, at the cost of slightly darkening everything. Use it when detail preservation is more important than color and brightness accuracy.

mobius

Smoothly map out-of-range values, while retaining contrast and colors for in-range material as much as possible. Use it when color accuracy is more important than detail preservation.

Default is none.

param

Tune the tone mapping algorithm.

This affects the following algorithms:

none

Ignored.

linear

Specifies the scale factor to use while stretching. Default to 1.0.

gamma

Specifies the exponent of the function. Default to 1.8.

clip Specify an extra linear coefficient to multiply into the signal before clipping. Default to 1.0.

reinhard

Specify the local contrast coefficient at the display peak. Default to 0.5, which means that in-gamut values will be about half as bright as when clipping.

hable

Ignored.

mobius

Specify the transition point from linear to mobius transform. Every value below this point is guaranteed to be mapped 1:1. The higher the value, the more accurate the result will be, at the cost of losing bright details. Default to 0.3, which due to the steep initial slope still

preserves in-range colors fairly accurately.

desat

Apply desaturation for highlights that exceed this level of brightness. The higher the parameter, the more color information will be preserved. This setting helps prevent unnaturally blown-out colors for super-highlights, by (smoothly) turning into white instead. This makes images feel more natural, at the cost of reducing information about out-of-range colors.

The default of 2.0 is somewhat conservative and will mostly just apply to skies or directly sunlit surfaces. A setting of 0.0 disables this option.

This option works only if the input frame has a supported color tag.

peak

Override signal/nominal/reference peak with this value. Useful when the embedded peak information in display metadata is not reliable or when tone mapping from a lower range to a higher range.

tpad

Temporarily pad video frames.

The filter accepts the following options:

start Specify number of delay frames before input video stream. Default is 0.

stop Specify number of padding frames after input video stream. Set to -1 to pad indefinitely. Default is 0.

start_mode

Set kind of frames added to beginning of stream. Can be either *add* or *clone*. With *add* frames of solid-color are added. With *clone* frames are clones of first frame. Default is *add*.

stop_mode

Set kind of frames added to end of stream. Can be either *add* or *clone*. With *add* frames of solid-color are added. With *clone* frames are clones of last frame. Default is *add*.

start_duration, stop_duration

Specify the duration of the start/stop delay. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax. These options override *start* and *stop*. Default is 0.

color

Specify the color of the padded area. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**.

The default value of *color* is "black".

transpose

Transpose rows with columns in the input video and optionally flip it.

It accepts the following parameters:

dir Specify the transposition direction.

Can assume the following values:

0, 4, cclock_flip

Rotate by 90 degrees counterclockwise and vertically flip (default), that is:

```
L.R  L.l
..-> ..
l.r  R.r
```

1, 5, clock

Rotate by 90 degrees clockwise, that is:

```
L.R  l.L
..-> ..
l.r  r.R
```

2, 6, cclock

Rotate by 90 degrees counterclockwise, that is:

```
L.R  R.r
..-> ..
l.r  L.l
```

3, 7, clock_flip

Rotate by 90 degrees clockwise and vertically flip, that is:

```
L.R  r.R
```

```
..-> ..
l.r  l.L
```

For values between 4-7, the transposition is only done if the input video geometry is portrait and not landscape. These values are deprecated, the "passthrough" option should be used instead.

Numerical values are deprecated, and should be dropped in favor of symbolic constants.

passthrough

Do not apply the transposition if the input geometry matches the one specified by the specified value. It accepts the following values:

none

Always apply transposition.

portrait

Preserve portrait geometry (when *height* \geq *width*).

landscape

Preserve landscape geometry (when *width* \geq *height*).

Default value is "none".

For example to rotate by 90 degrees clockwise and preserve portrait layout:

```
transpose=dir=1:passthrough=portrait
```

The command above can also be specified as:

```
transpose=1:portrait
```

transpose_npp

Transpose rows with columns in the input video and optionally flip it. For more in depth examples see the **transpose** video filter, which shares mostly the same options.

It accepts the following parameters:

dir Specify the transposition direction.

Can assume the following values:

cclock_flip

Rotate by 90 degrees counterclockwise and vertically flip. (default)

clock

Rotate by 90 degrees clockwise.

cclock

Rotate by 90 degrees counterclockwise.

clock_flip

Rotate by 90 degrees clockwise and vertically flip.

passthrough

Do not apply the transposition if the input geometry matches the one specified by the specified value. It accepts the following values:

none

Always apply transposition. (default)

portrait

Preserve portrait geometry (when *height* \geq *width*).

landscape

Preserve landscape geometry (when *width* \geq *height*).

trim

Trim the input so that the output contains one continuous subpart of the input.

It accepts the following parameters:

start Specify the time of the start of the kept section, i.e. the frame with the timestamp *start* will be the first frame in the output.

end Specify the time of the first frame that will be dropped, i.e. the frame immediately preceding the one with the timestamp *end* will be the last frame in the output.

start_pts

This is the same as *start*, except this option sets the start timestamp in timebase units instead of seconds.

end_pts

This is the same as *end*, except this option sets the end timestamp in timebase units instead of seconds.

duration

The maximum duration of the output in seconds.

start_frame

The number of the first frame that should be passed to the output.

end_frame

The number of the first frame that should be dropped.

start, **end**, and **duration** are expressed as time duration specifications; see **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

Note that the first two sets of the start/end options and the **duration** option look at the frame timestamp, while the *_frame* variants simply count the frames that pass through the filter. Also note that this filter does not modify the timestamps. If you wish for the output timestamps to start at zero, insert a *setpts* filter after the trim filter.

If multiple start or end options are set, this filter tries to be greedy and keep all the frames that match at least one of the specified constraints. To keep only the part that matches all the constraints at once, chain multiple trim filters.

The defaults are such that all the input is kept. So it is possible to set e.g. just the end values to keep everything before the specified time.

Examples:

- ⊕ Drop everything except the second minute of input:

```
ffmpeg -i INPUT -vf trim=60:120
```

- ⊕ Keep only the first second:

```
ffmpeg -i INPUT -vf trim=duration=1
```

unpremultiply

Apply alpha unpremultiply effect to input video stream using first plane of second stream as alpha.

Both streams must have same dimensions and same pixel format.

The filter accepts the following option:

planes

Set which planes will be processed, unprocessed planes will be copied. By default value 0xf, all planes will be processed.

If the format has 1 or 2 components, then luma is bit 0. If the format has 3 or 4 components: for RGB formats bit 0 is green, bit 1 is blue and bit 2 is red; for YUV formats bit 0 is luma, bit 1 is chroma-U and bit 2 is chroma-V. If present, the alpha channel is always the last bit.

inplace

Do not require 2nd input for processing, instead use alpha plane from input stream.

unsharp

Sharpen or blur the input video.

It accepts the following parameters:

luma_msize_x, lx

Set the luma matrix horizontal size. It must be an odd integer between 3 and 23. The default value is 5.

luma_msize_y, ly

Set the luma matrix vertical size. It must be an odd integer between 3 and 23. The default value is 5.

luma_amount, la

Set the luma effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 1.0.

chroma_msize_x, cx

Set the chroma matrix horizontal size. It must be an odd integer between 3 and 23. The default value is 5.

chroma_msize_y, cy

Set the chroma matrix vertical size. It must be an odd integer between 3 and 23. The default value is 5.

chroma_amount, ca

Set the chroma effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 0.0.

alpha_msize_x, ax

Set the alpha matrix horizontal size. It must be an odd integer between 3 and 23. The default value is 5.

alpha_msize_y, ay

Set the alpha matrix vertical size. It must be an odd integer between 3 and 23. The default value is 5.

alpha_amount, aa

Set the alpha effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 0.0.

All parameters are optional and default to the equivalent of the string '5:5:1.0:5:5:0.0'.

Examples

- ⊕ Apply strong luma sharpen effect:

```
unsharp=luma_msize_x=7:luma_msize_y=7:luma_amount=2.5
```

- ⊕ Apply a strong blur of both luma and chroma parameters:

unsharp=7:7:-2:7:7:-2

untile

Decompose a video made of tiled images into the individual images.

The frame rate of the output video is the frame rate of the input video multiplied by the number of tiles.

This filter does the reverse of **tile**.

The filter accepts the following options:

layout

Set the grid size (i.e. the number of lines and columns). For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

Examples

- ⊕ Produce a 1-second video from a still image file made of 25 frames stacked vertically, like an analogic film reel:

```
ffmpeg -r 1 -i image.jpg -vf untile=1x25 movie.mkv
```

uspp

Apply ultra slow/simple postprocessing filter that compresses and decompresses the image at several (or - in the case of **quality** level 8 - all) shifts and average the results.

The way this differs from the behavior of **spp** is that **uspp** actually encodes & decodes each case with libavcodec Snow, whereas **spp** uses a simplified intra only 8x8 DCT similar to MJPEG.

This filter is only available in ffmpeg version 4.4 or earlier.

The filter accepts the following options:

quality

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 0-8. If set to 0, the filter will have no effect. A value of 8 means the higher quality. For each increment of that value the speed drops by a factor of approximately 2. Default value is 3.

- qp** Force a constant quantization parameter. If not set, the filter will use the QP from the video stream (if available).

v360

Convert 360 videos between various formats.

The filter accepts the following options:

input**output**

Set format of the input/output video.

Available formats:

e**equirect**

Equirectangular projection.

c3x2**c6x1****c1x6**

Cubemap with 3x2/6x1/1x6 layout.

Format specific options:

in_pad**out_pad**

Set padding proportion for the input/output cubemap. Values in decimals.

Example values:

0 No padding.

0.01 1% of face is padding. For example, with 1920x1280 resolution face size would be 640x640 and padding would be 3 pixels from each side. ($640 * 0.01 = 6$ pixels)

Default value is **@samp{0}**. Maximum value is **@samp{0.1}**.

fin_pad**fout_pad**

Set fixed padding for the input/output cubemap. Values in pixels.

Default value is **@samp{0}**. If greater than zero it overrides other padding options.

in_forder**out_forder**

Set order of faces for the input/output cubemap. Choose one direction for each position.

Designation of directions:

r right

l left

u up

d down

f forward

b back

Default value is **@samp{rludfb}**.

in_frot**out_frot**

Set rotation of faces for the input/output cubemap. Choose one angle for each position.

Designation of angles:

0 0 degrees clockwise

1 90 degrees clockwise

2 180 degrees clockwise

3 270 degrees clockwise

Default value is **@samp{000000}**.

eac Equi-Angular Cubemap.

flat

gnomonic

rectilinear

Regular video.

Format specific options:

h_fov

v_fov

d_fov

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov

iv_fov

id_fov

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

dfisheye

Dual fisheye.

Format specific options:

h_fov

v_fov

d_fov

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov

iv_fov

id_fov

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

barrel

fb

barrelsplit

Facebook's 360 formats.

sg Stereographic format.

Format specific options:

h_fov

v_fov

d_fov

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov

iv_fov

id_fov

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

mercator

Mercator format.

ball Ball format, gives significant distortion toward the back.

hammer

Hammer-Aitoff map projection format.

sinusoidal

Sinusoidal map projection format.

fisheye

Fisheye projection.

Format specific options:

h_fov

v_fov

d_fov

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov

iv_fov

id_fov

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

pannini

Pannini projection.

Format specific options:

h_fov

Set output pannini parameter.

ih_fov

Set input pannini parameter.

cylindrical

Cylindrical projection.

Format specific options:

h_fov

v_fov

d_fov

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov

iv_fov

id_fov

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

perspective

Perspective projection. (*output only*)

Format specific options:

v_fov

Set perspective parameter.

tetrahedron

Tetrahedron projection.

tsp Truncated square pyramid projection.

he**hequirect**

Half equirectangular projection.

equisolid

Equisolid format.

Format specific options:

h_fov**v_fov****d_fov**

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov**iv_fov****id_fov**

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

og Orthographic format.

Format specific options:

h_fov**v_fov****d_fov**

Set output horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

ih_fov**iv_fov****id_fov**

Set input horizontal/vertical/diagonal field of view. Values in degrees.

If diagonal field of view is set it overrides horizontal and vertical field of view.

octahedron

Octahedron projection.

cylindricalea

Cylindrical Equal Area projection.

interp

Set interpolation method.*Note: more complex interpolation methods require much more memory to run.*

Available methods:

near**nearest**

Nearest neighbour.

line**linear**

Bilinear interpolation.

lagrange9

Lagrange9 interpolation.

cube**cubic**

Bicubic interpolation.

lanc**lanczos**

Lanczos interpolation.

sp16**spline16**

Spline16 interpolation.

gauss**gaussian**

Gaussian interpolation.

mitchell

Mitchell interpolation.

Default value is **@samp{line}**.

w

h Set the output video resolution.

Default resolution depends on formats.

in_stereo**out_stereo**

Set the input/output stereo format.

2d 2D mono

sbs Side by side

tb Top bottom

Default value is **@samp{2d}** for input and output format.

yaw**pitch**

roll Set rotation for the output video. Values in degrees.

rorder

Set rotation order for the output video. Choose one item for each position.

y, Y yaw

p, P pitch

r, R roll

Default value is `@samp{ypr}`.

h_flip

v_flip

d_flip

Flip the output video horizontally(swaps left-right)/vertically(swaps up-down)/in-depth(swaps back-forward). Boolean values.

ih_flip

iv_flip

Set if input video is flipped horizontally/vertically. Boolean values.

in_trans

Set if input video is transposed. Boolean value, by default disabled.

out_trans

Set if output video needs to be transposed. Boolean value, by default disabled.

h_offset

v_offset

Set output horizontal/vertical off-axis offset. Default is set to 0. Allowed range is from -1 to 1.

alpha_mask

Build mask in alpha plane for all unmapped pixels by marking them fully transparent. Boolean value, by default disabled.

reset_rot

Reset rotation of output video. Boolean value, by default disabled.

Examples

- ⊕ Convert equirectangular video to cubemap with 3x2 layout and 1% padding using bicubic interpolation:

```
ffmpeg -i input.mkv -vf v360=e:c3x2:cubic:out_pad=0.01 output.mkv
```

- ⊕ Extract back view of Equi-Angular Cubemap:

```
ffmpeg -i input.mkv -vf v360=eac:flat:yaw=180 output.mkv
```

- ⊕ Convert transposed and horizontally flipped Equi-Angular Cubemap in side-by-side stereo format to equirectangular top-bottom stereo format:

```
v360=eac:equirect:in_stereo=sbs:in_trans=1:ih_flip=1:out_stereo=tb
```

Commands

This filter supports subset of above options as **commands**.

vaguedenoiser

Apply a wavelet based denoiser.

It transforms each frame from the video input into the wavelet domain, using Cohen-Daubechies-Feauveau 9/7. Then it applies some filtering to the obtained coefficients. It does an inverse wavelet transform after. Due to wavelet properties, it should give a nice smoothed result, and reduced noise, without blurring picture features.

This filter accepts the following options:

threshold

The filtering strength. The higher, the more filtered the video will be. Hard thresholding can use a higher threshold than soft thresholding before the video looks overfiltered. Default value is 2.

method

The filtering method the filter will use.

It accepts the following values:

hard

All values under the threshold will be zeroed.

soft All values under the threshold will be zeroed. All values above will be reduced by the threshold.

garrote

Scales or nullifies coefficients - intermediary between (more) soft and (less) hard thresholding.

Default is garrote.

nsteps

Number of times, the wavelet will decompose the picture. Picture can't be decomposed beyond a particular point (typically, 8 for a 640x480 frame - as $2^9 = 512 > 480$). Valid values are integers between 1 and 32. Default value is 6.

percent

Partial of full denoising (limited coefficients shrinking), from 0 to 100. Default value is 85.

planes

A list of the planes to process. By default all planes are processed.

type The threshold type the filter will use.

It accepts the following values:

universal

Threshold used is same for all decompositions.

bayes

Threshold used depends also on each decomposition coefficients.

Default is universal.

varblur

Apply variable blur filter by using 2nd video stream to set blur radius. The 2nd stream must have the same dimensions.

This filter accepts the following options:

min_r

Set min allowed radius. Allowed range is from 0 to 254. Default is 0.

max_r

Set max allowed radius. Allowed range is from 1 to 255. Default is 8.

planes

Set which planes to process. By default, all are used.

The "varblur" filter also supports the **framesync** options.

Commands

This filter supports all the above options as **commands**.

vectorscope

Display 2 color component values in the two dimensional graph (which is called a vectorscope).

This filter accepts the following options:

mode, m

Set vectorscope mode.

It accepts the following values:

gray

tint Gray values are displayed on graph, higher brightness means more pixels have same component color value on location in graph. This is the default mode.

color

Gray values are displayed on graph. Surrounding pixels values which are not present in video frame are drawn in gradient of 2 color components which are set by option "x" and "y". The 3rd color component is static.

color2

Actual color components values present in video frame are displayed on graph.

color3

Similar as color2 but higher frequency of same values "x" and "y" on graph increases value of another color component, which is luminance by default values of "x" and "y".

color4

Actual colors present in video frame are displayed on graph. If two different colors map to same position on graph then color with higher value of component not present in graph is picked.

color5

Gray values are displayed on graph. Similar to "color" but with 3rd color component picked from radial gradient.

x Set which color component will be represented on X-axis. Default is 1.

y Set which color component will be represented on Y-axis. Default is 2.

intensity, i

Set intensity, used by modes: gray, color, color3 and color5 for increasing brightness of color component which represents frequency of (X, Y) location in graph.

envelope, e**none**

No envelope, this is default.

instant

Instant envelope, even darkest single pixel will be clearly highlighted.

peak

Hold maximum and minimum values presented in graph over time. This way you can still spot out of range values without constantly looking at vectorscope.

peak+instant

Peak and instant envelope combined together.

graticule, g

Set what kind of graticule to draw.

none**green****color****invert****opacity, o**

Set graticule opacity.

flags, f

Set graticule flags.

white

Draw graticule for white point.

black

Draw graticule for black point.

name

Draw color points short names.

bgopacity, b

Set background opacity.

lthreshold, l

Set low threshold for color component not represented on X or Y axis. Values lower than this value will be ignored. Default is 0. Note this value is multiplied with actual max possible value one pixel component can have. So for 8-bit input and low threshold value of 0.1 actual threshold is $0.1 * 255 = 25$.

hthreshold, h

Set high threshold for color component not represented on X or Y axis. Values higher than this value will be ignored. Default is 1. Note this value is multiplied with actual max possible value one pixel component can have. So for 8-bit input and high threshold value of 0.9 actual threshold is $0.9 * 255 = 230$.

colorspace, c

Set what kind of colorspace to use when drawing graticule.

auto

601

709

Default is auto.

tint0, t0**tint1, t1**

Set color tint for gray/tint vectorscope mode. By default both options are zero. This means no tint, and output will remain gray.

vidstabdetect

Analyze video stabilization/deshaking. Perform pass 1 of 2, see **vidstabtransform** for pass 2.

This filter generates a file with relative translation and rotation transform information about subsequent frames, which is then used by the **vidstabtransform** filter.

To enable compilation of this filter you need to configure FFmpeg with "--enable-libvidstab".

This filter accepts the following options:

result

Set the path to the file used to write the transforms information. Default value is *transforms.trf*.

shakiness

Set how shaky the video is and how quick the camera is. It accepts an integer in the range 1-10, a value of 1 means little shakiness, a value of 10 means strong shakiness. Default value is 5.

accuracy

Set the accuracy of the detection process. It must be a value in the range 1-15. A value of 1 means low accuracy, a value of 15 means high accuracy. Default value is 15.

stepsize

Set stepsize of the search process. The region around minimum is scanned with 1 pixel resolution. Default value is 6.

mincontrast

Set minimum contrast. Below this value a local measurement field is discarded. Must be a floating point value in the range 0-1. Default value is 0.3.

tripod

Set reference frame number for tripod mode.

If enabled, the motion of the frames is compared to a reference frame in the filtered stream, identified by the specified number. The idea is to compensate all movements in a more-or-less static scene and keep the camera view absolutely still.

If set to 0, it is disabled. The frames are counted starting from 1.

show

Show fields and transforms in the resulting frames. It accepts an integer in the range 0-2. Default value is 0, which disables any visualization.

Examples

- ⊕ Use default values:

```
vidstabdetect
```

- ⊕ Analyze strongly shaky movie and put the results in file *mytransforms.trf*:

```
vidstabdetect=shakiness=10:accuracy=15:result="mytransforms.trf"
```

- ⊕ Visualize the result of internal transformations in the resulting video:

```
vidstabdetect=show=1
```

- ⊕ Analyze a video with medium shakiness using **ffmpeg**:

```
ffmpeg -i input -vf vidstabdetect=shakiness=5:show=1 dummy.avi
```

vidstabtransform

Video stabilization/deshaking: pass 2 of 2, see **vidstabdetect** for pass 1.

Read a file with transform information for each frame and apply/compensate them. Together with the **vidstabdetect** filter this can be used to deshake videos. See also <<http://public.hronopik.de/vid.stab>>. It is important to also use the **unsharp** filter, see below.

To enable compilation of this filter you need to configure FFmpeg with "--enable-libvidstab".

Options

input

Set path to the file used to read the transforms. Default value is *transforms.trf*.

smoothing

Set the number of frames ($\text{value} * 2 + 1$) used for lowpass filtering the camera movements. Default value is 10.

For example a number of 10 means that 21 frames are used (10 in the past and 10 in the future) to smoothen the motion in the video. A larger value leads to a smoother video, but limits the acceleration of the camera (pan/tilt movements). 0 is a special case where a static camera is simulated.

optalgo

Set the camera path optimization algorithm.

Accepted values are:

gauss

gaussian kernel low-pass filter on camera motion (default)

avg averaging on transformations

maxshift

Set maximal number of pixels to translate frames. Default value is -1, meaning no limit.

maxangle

Set maximal angle in radians (degree*PI/180) to rotate frames. Default value is -1, meaning no limit.

crop

Specify how to deal with borders that may be visible due to movement compensation.

Available values are:

keep

keep image information from previous frame (default)

black

fill the border black

invert

Invert transforms if set to 1. Default value is 0.

relative

Consider transforms as relative to previous frame if set to 1, absolute if set to 0. Default value is 0.

zoom

Set percentage to zoom. A positive value will result in a zoom-in effect, a negative value in a zoom-out effect. Default value is 0 (no zoom).

optzoom

Set optimal zooming to avoid borders.

Accepted values are:

- 0** disabled
- 1** optimal static zoom value is determined (only very strong movements will lead to visible borders) (default)
- 2** optimal adaptive zoom value is determined (no borders will be visible), see **zoomspeed**

Note that the value given at zoom is added to the one calculated here.

zoomspeed

Set percent to zoom maximally each frame (enabled when **optzoom** is set to 2). Range is from 0 to 5, default value is 0.25.

interpol

Specify type of interpolation.

Available values are:

no no interpolation

linear

linear only horizontal

bilinear

linear in both directions (default)

bicubic

cubic in both directions (slow)

tripod

Enable virtual tripod mode if set to 1, which is equivalent to "relative=0:smoothing=0". Default value is 0.

Use also "tripod" option of **vidstabdetect**.

debug

Increase log verbosity if set to 1. Also the detected global motions are written to the temporary file *global_motions.trf*. Default value is 0.

Examples

- ⊕ Use **ffmpeg** for a typical stabilization with default values:

```
ffmpeg -i inp.mpeg -vf vidstabtransform,unsharp=5:5:0.8:3:3:0.4 inp_stabilized.mpeg
```

Note the use of the **unsharp** filter which is always recommended.

- ⊕ Zoom in a bit more and load transform data from a given file:

```
vidstabtransform=zoom=5:input="mytransforms.trf"
```

- ⊕ Smoothen the video even more:

```
vidstabtransform=smoothing=30
```

vflip

Flip the input video vertically.

For example, to vertically flip a video with **ffmpeg**:

```
ffmpeg -i in.avi -vf "vflip" out.avi
```

vfrdet

Detect variable frame rate video.

This filter tries to detect if the input is variable or constant frame rate.

At end it will output number of frames detected as having variable delta pts, and ones with constant delta pts. If there was frames with variable delta, than it will also show min, max and average delta encountered.

vibrance

Boost or alter saturation.

The filter accepts the following options:

intensity

Set strength of boost if positive value or strength of alter if negative value. Default is 0. Allowed range is from -2 to 2.

rbal Set the red balance. Default is 1. Allowed range is from -10 to 10.

gbal Set the green balance. Default is 1. Allowed range is from -10 to 10.

bbal Set the blue balance. Default is 1. Allowed range is from -10 to 10.

rlum

Set the red luma coefficient.

glum

Set the green luma coefficient.

blum

Set the blue luma coefficient.

alternate

If "intensity" is negative and this is set to 1, colors will change, otherwise colors will be less saturated, more towards gray.

Commands

This filter supports the all above options as **commands**.

vif

Obtain the average VIF (Visual Information Fidelity) between two input videos.

This filter takes two input videos.

Both input videos must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained average VIF score is printed through the logging system.

The filter stores the calculated VIF score of each frame.

This filter also supports the **framesync** options.

In the below example the input file *main.mpg* being processed is compared with the reference file *ref.mpg*.

```
ffmpeg -i main.mpg -i ref.mpg -lavfi vif -f null -
```

vignette

Make or reverse a natural vignetting effect.

The filter accepts the following options:

angle, a

Set lens angle expression as a number of radians.

The value is clipped in the "[0,PI/2]" range.

Default value: "PI/5"

x0

y0 Set center coordinates expressions. Respectively "w/2" and "h/2" by default.

mode

Set forward/backward mode.

Available modes are:

forward

The larger the distance from the central point, the darker the image becomes.

backward

The larger the distance from the central point, the brighter the image becomes. This can be used to reverse a vignette effect, though there is no automatic detection to extract the lens **angle** and other settings (yet). It can also be used to create a burning effect.

Default value is **forward**.

eval Set evaluation mode for the expressions (**angle**, **x0**, **y0**).

It accepts the following values:

init Evaluate expressions only once during the filter initialization.

frame

Evaluate expressions for each incoming frame. This is way slower than the **init** mode since it

requires all the scalars to be re-computed, but it allows advanced dynamic expressions.

Default value is **init**.

dither

Set dithering to reduce the circular banding effects. Default is 1 (enabled).

aspect

Set vignette aspect. This setting allows one to adjust the shape of the vignette. Setting this value to the SAR of the input will make a rectangular vignetting following the dimensions of the video.

Default is "1/1".

Expressions

The **alpha**, **x0** and **y0** expressions can contain the following parameters.

w

h input width and height

n the number of input frame, starting from 0

pts the PTS (Presentation TimeStamp) time of the filtered video frame, expressed in *TB* units, NAN if undefined

r frame rate of the input video, NAN if the input frame rate is unknown

t the PTS (Presentation TimeStamp) of the filtered video frame, expressed in seconds, NAN if undefined

tb time base of the input video

Examples

⊕ Apply simple strong vignetting effect:

vignette=PI/4

⊕ Make a flickering vignetting:

```
vignette='PI/4+random(1)*PI/50':eval=frame
```

vmafmotion

Obtain the average VMAF motion score of a video. It is one of the component metrics of VMAF.

The obtained average motion score is printed through the logging system.

The filter accepts the following options:

stats_file

If specified, the filter will use the named file to save the motion score of each frame with respect to the previous frame. When filename equals "-" the data is sent to standard output.

Example:

```
ffmpeg -i ref.mpg -vf vmafmotion -f null -
```

vstack

Stack input videos vertically.

All streams must be of same pixel format and of same width.

Note that this filter is faster than using **overlay** and **pad** filter to create same output.

The filter accepts the following options:

inputs

Set number of input streams. Default is 2.

shortest

If set to 1, force the output to terminate when the shortest input terminates. Default value is 0.

w3fdif

Deinterlace the input video ("w3fdif" stands for "Weston 3 Field Deinterlacing Filter").

Based on the process described by Martin Weston for BBC R&D, and implemented based on the deinterlace algorithm written by Jim Easterbrook for BBC R&D, the Weston 3 field deinterlacing filter uses filter coefficients calculated by BBC R&D.

This filter uses field-dominance information in frame to decide which of each pair of fields to place

first in the output. If it gets it wrong use **setfield** filter before "w3fdif" filter.

There are two sets of filter coefficients, so called "simple" and "complex". Which set of filter coefficients is used can be set by passing an optional parameter:

filter

Set the interlacing filter coefficients. Accepts one of the following values:

simple

Simple filter coefficient set.

complex

More-complex filter coefficient set.

Default value is **complex**.

mode

The interlacing mode to adopt. It accepts one of the following values:

frame

Output one frame for each frame.

field

Output one frame for each field.

The default value is "field".

parity

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

tff Assume the top field is first.

bff Assume the bottom field is first.

auto Enable automatic detection of field parity.

The default value is "auto". If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

deint

Specify which frames to deinterlace. Accepts one of the following values:

all Deinterlace all frames,

interlaced

Only deinterlace frames marked as interlaced.

Default value is **all**.

Commands

This filter supports same **commands** as options.

waveform

Video waveform monitor.

The waveform monitor plots color component intensity. By default luminance only. Each column of the waveform corresponds to a column of pixels in the source video.

It accepts the following options:

mode, m

Can be either "row", or "column". Default is "column". In row mode, the graph on the left side represents color component value 0 and the right side represents value = 255. In column mode, the top side represents color component value = 0 and bottom side represents value = 255.

intensity, i

Set intensity. Smaller values are useful to find out how many values of the same luminance are distributed across input rows/columns. Default value is 0.04. Allowed range is [0, 1].

mirror, r

Set mirroring mode. 0 means unmirrored, 1 means mirrored. In mirrored mode, higher values will be represented on the left side for "row" mode and at the top for "column" mode. Default is 1 (mirrored).

display, d

Set display mode. It accepts the following values:

overlay

Presents information identical to that in the "parade", except that the graphs representing color components are superimposed directly over one another.

This display mode makes it easier to spot relative differences or similarities in overlapping areas of the color components that are supposed to be identical, such as neutral whites, grays, or blacks.

stack

Display separate graph for the color components side by side in "row" mode or one below the other in "column" mode.

parade

Display separate graph for the color components side by side in "column" mode or one below the other in "row" mode.

Using this display mode makes it easy to spot color casts in the highlights and shadows of an image, by comparing the contours of the top and the bottom graphs of each waveform. Since whites, grays, and blacks are characterized by exactly equal amounts of red, green, and blue, neutral areas of the picture should display three waveforms of roughly equal width/height. If not, the correction is easy to perform by making level adjustments the three waveforms.

Default is "stack".

components, c

Set which color components to display. Default is 1, which means only luminance or red color component if input is in RGB colorspace. If is set for example to 7 it will display all 3 (if) available color components.

envelope, e**none**

No envelope, this is default.

instant

Instant envelope, minimum and maximum values presented in graph will be easily visible even with small "step" value.

peak

Hold minimum and maximum values presented in graph across time. This way you can still spot out of range values without constantly looking at waveforms.

peak+instant

Peak and instant envelope combined together.

filter, f**lowpass**

No filtering, this is default.

flat Luma and chroma combined together.

aflat

Similar as above, but shows difference between blue and red chroma.

xflat

Similar as above, but use different colors.

yflat

Similar as above, but again with different colors.

chroma

Displays only chroma.

color

Displays actual color value on waveform.

acolor

Similar as above, but with luma showing frequency of chroma values.

graticule, g

Set which graticule to display.

none

Do not display graticule.

green

Display green graticule showing legal broadcast ranges.

orange

Display orange graticule showing legal broadcast ranges.

invert

Display invert graticule showing legal broadcast ranges.

opacity, o

Set graticule opacity.

flags, fl

Set graticule flags.

numbers

Draw numbers above lines. By default enabled.

dots Draw dots instead of lines.

scale, s

Set scale used for displaying graticule.

digital**millivolts****ire**

Default is digital.

bgopacity, b

Set background opacity.

tint0, t0**tint1, t1**

Set tint for output. Only used with lowpass filter and when display is not overlay and input pixel formats are not RGB.

fitmode, fm

Set sample aspect ratio of video output frames. Can be used to configure waveform so it is not stretched too much in one of directions.

none

Set sample aspect ration to 1/1.

size Set sample aspect ratio to match input size of video

Default is **none**.

weave, doubleweave

The "weave" takes a field-based video input and join each two sequential fields into single frame, producing a new double height clip with half the frame rate and half the frame count.

The "doubleweave" works same as "weave" but without halving frame rate and frame count.

It accepts the following option:

first_field

Set first field. Available values are:

top, t

Set the frame as top-field-first.

bottom, b

Set the frame as bottom-field-first.

Examples

- ⊕ Interlace video using **select** and **separatefields** filter:

```
separatefields,select=eq(mod(n,4),0)+eq(mod(n,4),3),weave
```

xbr

Apply the xBR high-quality magnification filter which is designed for pixel art. It follows a set of edge-detection rules, see <<https://forums.libretro.com/t/xbr-algorithm-tutorial/123>>.

It accepts the following option:

- n** Set the scaling dimension: 2 for "2xBR", 3 for "3xBR" and 4 for "4xBR". Default is 3.

xcorrelate

Apply normalized cross-correlation between first and second input video stream.

Second input video stream dimensions must be lower than first input video stream.

The filter accepts the following options:

planes

Set which planes to process.

secondary

Set which secondary video frames will be processed from second input video stream, can be *first* or *all*. Default is *all*.

The "xcorrelate" filter also supports the **framesync** options.

xfade

Apply cross fade from one input video stream to another input video stream. The cross fade is applied for specified duration.

Both inputs must be constant frame-rate and have the same resolution, pixel format, frame rate and timebase.

The filter accepts the following options:

transition

Set one of available transition effects:

custom

fade

wipeleft

wiperight

wipeup

wipedown

slideleft

slideright

slideup

slidedown

circlecrop

rectcrop

distance

fadeblack

fadewhite

radial

smoothleft

smoothright

smoothup

smoothdown

circleopen

circleclose

vertopen
vertclose
horzopen
horzclose
dissolve
pixelize
diagtl
diagtr
diagbl
diagbr
hlslice
hrslice
vuslice
vdslice
hblur
fadegrays
wipetl
wipetr
wipebl
wipebr
squeezeh
squeezev
zoomin
fadefast
fadeslow

Default transition effect is fade.

duration

Set cross fade duration in seconds. Range is 0 to 60 seconds. Default duration is 1 second.

offset

Set cross fade start relative to first input stream in seconds. Default offset is 0.

expr

Set expression for custom transition effect.

The expressions can use the following variables and functions:

X

Y The coordinates of the current sample.

W

H The width and height of the image.

P Progress of transition effect.

PLANE

Currently processed plane.

A Return value of first input at current location and plane.

B Return value of second input at current location and plane.

a0(x, y)

a1(x, y)

a2(x, y)

a3(x, y)

Return the value of the pixel at location (x,y) of the first/second/third/fourth component of first input.

b0(x, y)

b1(x, y)

b2(x, y)

b3(x, y)

Return the value of the pixel at location (x,y) of the first/second/third/fourth component of second input.

Examples

- ⊕ Cross fade from one input video to another input video, with fade transition and duration of transition of 2 seconds starting at offset of 5 seconds:

```
ffmpeg -i first.mp4 -i second.mp4 -filter_complex xfade=transition=fade:duration=2:offset=5 output.mp4
```

xmedian

Pick median pixels from several input videos.

The filter accepts the following options:

inputs

Set number of inputs. Default is 3. Allowed range is from 3 to 255. If number of inputs is even number, than result will be mean value between two median values.

planes

Set which planes to filter. Default value is 15, by which all planes are processed.

percentile

Set median percentile. Default value is 0.5. Default value of 0.5 will pick always median values, while 0 will pick minimum values, and 1 maximum values.

Commands

This filter supports all above options as **commands**, excluding option "inputs".

xstack

Stack video inputs into custom layout.

All streams must be of same pixel format.

The filter accepts the following options:

inputs

Set number of input streams. Default is 2.

layout

Specify layout of inputs. This option requires the desired layout configuration to be explicitly set by the user. This sets position of each video input in output. Each input is separated by '|'. The first number represents the column, and the second number represents the row. Numbers start at 0 and are separated by '_'. Optionally one can use wX and hX, where X is video input from which to take width or height. Multiple values can be used when separated by '+'. In such case values are summed together.

Note that if inputs are of different sizes gaps may appear, as not all of the output video frame will be filled. Similarly, videos can overlap each other if their position doesn't leave enough space for the full frame of adjoining videos.

For 2 inputs, a default layout of "0_0|w0_0" (equivalent to "grid=2x1") is set. In all other cases, a layout or a grid must be set by the user. Either "grid" or "layout" can be specified at a time. Specifying both will result in an error.

grid Specify a fixed size grid of inputs. This option is used to create a fixed size grid of the input streams. Set the grid size in the form "COLUMNSxROWS". There must be "ROWS * COLUMNS" input streams and they will be arranged as a grid with "ROWS" rows and "COLUMNS" columns. When using this option, each input stream within a row must have the same height and all the rows must have the same width.

If "grid" is set, then "inputs" option is ignored and is implicitly set to "ROWS * COLUMNS".

For 2 inputs, a default grid of "2x1" (equivalent to "layout=0_0|w0_0") is set. In all other cases, a layout or a grid must be set by the user. Either "grid" or "layout" can be specified at a time. Specifying both will result in an error.

shortest

If set to 1, force the output to terminate when the shortest input terminates. Default value is 0.

fill If set to valid color, all unused pixels will be filled with that color. By default fill is set to none, so it is disabled.

Examples

- ⊕ Display 4 inputs into 2x2 grid.

Layout:

```
input1(0, 0) | input3(w0, 0)
input2(0, h0) | input4(w0, h0)
```

```
xstack=inputs=4:layout=0_0|0_h0|w0_0|w0_h0
```

Note that if inputs are of different sizes, gaps or overlaps may occur.

- ⊕ Display 4 inputs into 1x4 grid.

Layout:

```
input1(0, 0)
input2(0, h0)
input3(0, h0+h1)
```

```
input4(0, h0+h1+h2)
```

```
xstack=inputs=4:layout=0_0|0_h0|0_h0+h1|0_h0+h1+h2
```

Note that if inputs are of different widths, unused space will appear.

- ⊕ Display 9 inputs into 3x3 grid.

Layout:

```
input1(0, 0)   | input4(w0, 0)   | input7(w0+w3, 0)
input2(0, h0)  | input5(w0, h0)  | input8(w0+w3, h0)
input3(0, h0+h1) | input6(w0, h0+h1) | input9(w0+w3, h0+h1)
```

```
xstack=inputs=9:layout=0_0|0_h0|0_h0+h1|w0_0|w0_h0|w0_h0+h1|w0+w3_0|w0+w3_h0|w0+w3_h0+h1
```

Note that if inputs are of different sizes, gaps or overlaps may occur.

- ⊕ Display 16 inputs into 4x4 grid.

Layout:

```
input1(0, 0)   | input5(w0, 0)   | input9(w0+w4, 0)   | input13(w0+w4+w8, 0)
input2(0, h0)  | input6(w0, h0)  | input10(w0+w4, h0) | input14(w0+w4+w8, h0)
input3(0, h0+h1) | input7(w0, h0+h1) | input11(w0+w4, h0+h1) | input15(w0+w4+w8, h0+h1)
input4(0, h0+h1+h2) | input8(w0, h0+h1+h2) | input12(w0+w4, h0+h1+h2) | input16(w0+w4+w8, h0+h1+h2)
```

```
xstack=inputs=16:layout=0_0|0_h0|0_h0+h1|0_h0+h1+h2|w0_0|w0_h0|w0_h0+h1|w0_h0+h1+h2|w0+w4_0|
w0+w4_h0|w0+w4_h0+h1|w0+w4_h0+h1+h2|w0+w4+w8_0|w0+w4+w8_h0|w0+w4+w8_h0+h1|w0+w4+w8_h0+h1+h2
```

Note that if inputs are of different sizes, gaps or overlaps may occur.

yadif

Deinterlace the input video ("yadif" means "yet another deinterlacing filter").

It accepts the following parameters:

mode

The interlacing mode to adopt. It accepts one of the following values:

0, send_frame

Output one frame for each frame.

1, send_field

Output one frame for each field.

2, send_frame_nospatial

Like "send_frame", but it skips the spatial interlacing check.

3, send_field_nospatial

Like "send_field", but it skips the spatial interlacing check.

The default value is "send_frame".

parity

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

0, tff

Assume the top field is first.

1, bff

Assume the bottom field is first.

-1, auto

Enable automatic detection of field parity.

The default value is "auto". If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

deint

Specify which frames to deinterlace. Accepts one of the following values:

0, all

Deinterlace all frames.

1, interlaced

Only deinterlace frames marked as interlaced.

The default value is "all".

yadif_cuda

Deinterlace the input video using the **yadif** algorithm, but implemented in CUDA so that it can work as part of a GPU accelerated pipeline with nvdec and/or nvenc.

It accepts the following parameters:

mode

The interlacing mode to adopt. It accepts one of the following values:

0, send_frame

Output one frame for each frame.

1, send_field

Output one frame for each field.

2, send_frame_nospatial

Like "send_frame", but it skips the spatial interlacing check.

3, send_field_nospatial

Like "send_field", but it skips the spatial interlacing check.

The default value is "send_frame".

parity

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

0, tff

Assume the top field is first.

1, bff

Assume the bottom field is first.

-1, auto

Enable automatic detection of field parity.

The default value is "auto". If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

deint

Specify which frames to deinterlace. Accepts one of the following values:

0, all

Deinterlace all frames.

1, interlaced

Only deinterlace frames marked as interlaced.

The default value is "all".

yaepblur

Apply blur filter while preserving edges ("yaepblur" means "yet another edge preserving blur filter"). The algorithm is described in "J. S. Lee, Digital image enhancement and noise filtering by use of local statistics, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-2, 1980."

It accepts the following parameters:

radius, r

Set the window radius. Default value is 3.

planes, p

Set which planes to filter. Default is only the first plane.

sigma, s

Set blur strength. Default value is 128.

Commands

This filter supports same **commands** as options.

zoompan

Apply Zoom & Pan effect.

This filter accepts the following options:

zoom, z

Set the zoom expression. Range is 1-10. Default is 1.

x

y Set the x and y expression. Default is 0.

d Set the duration expression in number of frames. This sets for how many number of frames effect will last for single input image. Default is 90.

s Set the output image size, default is 'hd720'.

fps Set the output frame rate, default is '25'.

Each expression can contain the following constants:

in_w, iw

Input width.

in_h, ih

Input height.

out_w, ow

Output width.

out_h, oh

Output height.

in Input frame count.

on Output frame count.

in_time, it

The input timestamp expressed in seconds. It's NAN if the input timestamp is unknown.

out_time, time, ot

The output timestamp expressed in seconds.

x

y Last calculated 'x' and 'y' position from 'x' and 'y' expression for current input frame.

px

py 'x' and 'y' of last output frame of previous input frame or 0 when there was not yet such frame (first input frame).

zoom

Last calculated zoom from 'z' expression for current input frame.

pzoom

Last calculated zoom of last output frame of previous input frame.

duration

Number of output frames for current input frame. Calculated from 'd' expression for each input frame.

pduration

number of output frames created for previous input frame

a Rational number: input width / input height

sar sample aspect ratio

dar display aspect ratio

Examples

⊕ Zoom in up to 1.5x and pan at same time to some spot near center of picture:

```
zoompan=z='min(zoom+0.0015,1.5)':d=700:x='if(ge(zoom,1.5),x,x+1/a)':y='if(ge(zoom,1.5),y,y+1)':s=64
```

⊕ Zoom in up to 1.5x and pan always at center of picture:

```
zoompan=z='min(zoom+0.0015,1.5)':d=700:x='iw/2-(iw/zoom/2)':y='ih/2-(ih/zoom/2)'
```

⊕ Same as above but without pausing:

```
zoompan=z='min(max(zoom,pzoom)+0.0015,1.5)':d=1:x='iw/2-(iw/zoom/2)':y='ih/2-(ih/zoom/2)'
```

⊕ Zoom in 2x into center of picture only for the first second of the input video:

```
zoompan=z='if(between(in_time,0,1),2,1)':d=1:x='iw/2-(iw/zoom/2)':y='ih/2-(ih/zoom/2)'
```

zscale

Scale (resize) the input video, using the z.lib library: <<https://github.com/sekrit-twc/zimg>>. To enable compilation of this filter, you need to configure FFmpeg with "--enable-libzimg".

The zscale filter forces the output display aspect ratio to be the same as the input, by changing the output sample aspect ratio.

If the input image format is different from the format requested by the next filter, the zscale filter will convert the input to the requested format.

Options

The filter accepts the following options.

width, w**height, h**

Set the output video dimension expression. Default value is the input dimension.

If the *width* or *w* value is 0, the input width is used for the output. If the *height* or *h* value is 0, the input height is used for the output.

If one and only one of the values is $-n$ with $n \geq 1$, the zscale filter will use a value that maintains the aspect ratio of the input image, calculated from the other specified dimension. After that it will, however, make sure that the calculated dimension is divisible by n and adjust the value if necessary.

If both values are $-n$ with $n \geq 1$, the behavior will be identical to both values being set to 0 as previously detailed.

See below for the list of accepted constants for use in the dimension expression.

size, s

Set the video size. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

dither, d

Set the dither type.

Possible values are:

none
ordered
random
error_diffusion

Default is none.

filter, f

Set the resize filter type.

Possible values are:

point
bilinear
bicubic
spline16
spline36
lanczos

Default is bilinear.

range, r

Set the color range.

Possible values are:

input
limited
full

Default is same as input.

primaries, p

Set the color primaries.

Possible values are:

input
709
unspecified

170m

240m

2020

Default is same as input.

transfer, t

Set the transfer characteristics.

Possible values are:

input

709

unspecified

601

linear

2020_10

2020_12

smpte2084

iec61966-2-1

arib-std-b67

Default is same as input.

matrix, m

Set the colorspace matrix.

Possible value are:

input

709

unspecified

470bg

170m

2020_ncl

2020_cl

Default is same as input.

rangein, rin

Set the input color range.

Possible values are:

input
limited
full

Default is same as input.

primariesin, pin

Set the input color primaries.

Possible values are:

input
709
unspecified
170m
240m
2020

Default is same as input.

transferin, tin

Set the input transfer characteristics.

Possible values are:

input
709
unspecified
601
linear
2020_10
2020_12

Default is same as input.

matrixin, min

Set the input colorspace matrix.

Possible value are:

input
709
unspecified
470bg
170m
2020_ncl
2020_cl

chromal, c

Set the output chroma location.

Possible values are:

input
left
center
topleft
top
bottomleft
bottom

chromalin, cin

Set the input chroma location.

Possible values are:

input
left
center
topleft
top
bottomleft
bottom

npl Set the nominal peak luminance.

param_a

Parameter A for scaling filters. Parameter "b" for bicubic, and the number of filter taps for lanczos.

param_b

Parameter B for scaling filters. Parameter "c" for bicubic.

The values of the **w** and **h** options are expressions containing the following constants:

in_w

in_h

The input width and height

iw

ih These are the same as *in_w* and *in_h*.

out_w

out_h

The output (scaled) width and height

ow

oh These are the same as *out_w* and *out_h*

a The same as iw / ih

sar input sample aspect ratio

dar The input display aspect ratio. Calculated from $(iw / ih) * sar$.

hsub

vsub

horizontal and vertical input chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

ohsub

ovsub

horizontal and vertical output chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

Commands

This filter supports the following commands:

width, w

height, h

Set the output video dimension expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

OPENCL VIDEO FILTERS

Below is a description of the currently available OpenCL video filters.

To enable compilation of these filters you need to configure FFmpeg with "--enable-openc1".

Running OpenCL filters requires you to initialize a hardware device and to pass that device to all filters in any filter graph.

-init_hw_device openc1[=*name*][:*device*[,*key*=*value*...]]

Initialise a new hardware device of type *openc1* called *name*, using the given device parameters.

-filter_hw_device *name*

Pass the hardware device called *name* to all filters in any filter graph.

For more detailed information see <<https://www.ffmpeg.org/ffmpeg.html#Advanced-Video-options>>

- ⊕ Example of choosing the first device on the second platform and running `avgblur_openc1` filter with default parameters on it.

```
-init_hw_device openc1=gpu:1.0 -filter_hw_device gpu -i INPUT -vf "hwupload, avgblur_openc1, hwdownload"
```

Since OpenCL filters are not able to access frame data in normal memory, all frame data needs to be uploaded(**hwupload**) to hardware surfaces connected to the appropriate device before being used and then downloaded(**hwdownload**) back to normal memory. Note that **hwupload** will upload to a surface with the same layout as the software frame, so it may be necessary to add a **format** filter immediately before to get the input into the right format and **hwdownload** does not support all formats on the output - it may be necessary to insert an additional **format** filter immediately following in the graph to get the output in a supported format.

avgblur_openc1

Apply average blur filter.

The filter accepts the following options:

sizeX

Set horizontal radius size. Range is "[1, 1024]" and default value is 1.

planes

Set which planes to filter. Default value is 0xf, by which all planes are processed.

sizeY

Set vertical radius size. Range is "[1, 1024]" and default value is 0. If zero, "sizeX" value will be used.

Example

- ⊕ Apply average blur filter with horizontal and vertical size of 3, setting each pixel of the output to the average value of the 7x7 region centered on it in the input. For pixels on the edges of the image, the region does not extend beyond the image boundaries, and so out-of-range coordinates are not used in the calculations.

```
-i INPUT -vf "hwupload, avgbblur_openc1=3, hwdownload" OUTPUT
```

boxblur_openc1

Apply a boxblur algorithm to the input video.

It accepts the following parameters:

luma_radius, lr

luma_power, lp

chroma_radius, cr

chroma_power, cp

alpha_radius, ar

alpha_power, ap

A description of the accepted options follows.

luma_radius, lr

chroma_radius, cr

alpha_radius, ar

Set an expression for the box radius in pixels used for blurring the corresponding input plane.

The radius value must be a non-negative number, and must not be greater than the value of the expression "min(w,h)/2" for the luma and alpha planes, and of "min(cw,ch)/2" for the chroma

planes.

Default value for **luma_radius** is "2". If not specified, **chroma_radius** and **alpha_radius** default to the corresponding value set for **luma_radius**.

The expressions can contain the following constants:

w

h The input width and height in pixels.

cw

ch The input chroma image width and height in pixels.

hsub

vsub

The horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p", *hsub* is 2 and *vsub* is 1.

luma_power, lp

chroma_power, cp

alpha_power, ap

Specify how many times the boxblur filter is applied to the corresponding plane.

Default value for **luma_power** is 2. If not specified, **chroma_power** and **alpha_power** default to the corresponding value set for **luma_power**.

A value of 0 will disable the effect.

Examples

Apply boxblur filter, setting each pixel of the output to the average value of box-radiuses *luma_radius*, *chroma_radius*, *alpha_radius* for each plane respectively. The filter will apply *luma_power*, *chroma_power*, *alpha_power* times onto the corresponding plane. For pixels on the edges of the image, the radius does not extend beyond the image boundaries, and so out-of-range coordinates are not used in the calculations.

- ⊕ Apply a boxblur filter with the luma, chroma, and alpha radius set to 2 and luma, chroma, and alpha power set to 3. The filter will run 3 times with box-radius set to 2 for every plane of the image.

```
-i INPUT -vf "hwupload, boxblur_openc1=luma_radius=2:luma_power=3, hwdownload" OUTPUT
-i INPUT -vf "hwupload, boxblur_openc1=2:3, hwdownload" OUTPUT
```

- ⊕ Apply a boxblur filter with luma radius set to 2, luma_power to 1, chroma_radius to 4, chroma_power to 5, alpha_radius to 3 and alpha_power to 7.

For the luma plane, a 2x2 box radius will be run once.

For the chroma plane, a 4x4 box radius will be run 5 times.

For the alpha plane, a 3x3 box radius will be run 7 times.

```
-i INPUT -vf "hwupload, boxblur_openc1=2:1:4:5:3:7, hwdownload" OUTPUT
```

colorkey_openc1

RGB colorspace color keying.

The filter accepts the following options:

color

The color which will be replaced with transparency.

similarity

Similarity percentage with the key color.

0.01 matches only the exact key color, while 1.0 matches everything.

blend

Blend percentage.

0.0 makes pixels either fully transparent, or not transparent at all.

Higher values result in semi-transparent pixels, with a higher transparency the more similar the pixels color is to the key color.

Examples

- ⊕ Make every semi-green pixel in the input transparent with some slight blending:

```
-i INPUT -vf "hwupload, colorkey_openc1=green:0.3:0.1, hwdownload" OUTPUT
```

convolution_openc1

Apply convolution of 3x3, 5x5, 7x7 matrix.

The filter accepts the following options:

0m

1m

2m

3m Set matrix for each plane. Matrix is sequence of 9, 25 or 49 signed numbers. Default value for each plane is "0 0 0 0 1 0 0 0 0".

0rdiv

1rdiv

2rdiv

3rdiv

Set multiplier for calculated value for each plane. If unset or 0, it will be sum of all matrix elements. The option value must be a float number greater or equal to 0.0. Default value is 1.0.

0bias

1bias

2bias

3bias

Set bias for each plane. This value is added to the result of the multiplication. Useful for making the overall image brighter or darker. The option value must be a float number greater or equal to 0.0. Default value is 0.0.

Examples

⊕ Apply sharpen:

```
-i INPUT -vf "hwupload, convolution_openc1=0 -1 0 -1 5 -1 0 -1 0:0 -1 0 -1 5 -1 0 -1 0:0 -1 0 -1 5 -1 0 -1 0:0"
```

⊕ Apply blur:

```
-i INPUT -vf "hwupload, convolution_openc1=1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1:1 1 1 1 1 1 1 1"
```

⊕ Apply edge enhance:

```
-i INPUT -vf "hwupload, convolution_openc1=0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0:0 0 0 -1 1 0 0 0 0"
```


- ⊕ Apply erosion filter with threshold0 set to 30, threshold1 set 40, threshold2 set to 50 and coordinates set to 231, setting each pixel of the output to the local minimum between pixels: 1, 2, 3, 6, 7, 8 of the 3x3 region centered on it in the input. If the difference between input pixel and local minimum is more then threshold of the corresponding plane, output pixel will be set to input pixel - threshold of corresponding plane.

```
-i INPUT -vf "hwupload, erosion_openc1=30:40:50:coordinates=231, hwdownload" OUTPUT
```

deshake_openc1

Feature-point based video stabilization filter.

The filter accepts the following options:

tripod

Simulates a tripod by preventing any camera movement whatsoever from the original frame.

Defaults to 0.

debug

Whether or not additional debug info should be displayed, both in the processed output and in the console.

Note that in order to see console debug output you will also need to pass "-v verbose" to ffmpeg.

Viewing point matches in the output video is only supported for RGB input.

Defaults to 0.

adaptive_crop

Whether or not to do a tiny bit of cropping at the borders to cut down on the amount of mirrored pixels.

Defaults to 1.

refine_features

Whether or not feature points should be refined at a sub-pixel level.

This can be turned off for a slight performance gain at the cost of precision.

Defaults to 1.

smooth_strength

The strength of the smoothing applied to the camera path from 0.0 to 1.0.

1.0 is the maximum smoothing strength while values less than that result in less smoothing.

0.0 causes the filter to adaptively choose a smoothing strength on a per-frame basis.

Defaults to 0.0.

smooth_window_multiplier

Controls the size of the smoothing window (the number of frames buffered to determine motion information from).

The size of the smoothing window is determined by multiplying the framerate of the video by this number.

Acceptable values range from 0.1 to 10.0.

Larger values increase the amount of motion data available for determining how to smooth the camera path, potentially improving smoothness, but also increase latency and memory usage.

Defaults to 2.0.

Examples

- ⊕ Stabilize a video with a fixed, medium smoothing strength:

```
-i INPUT -vf "hwupload, deshake_openc1=smooth_strength=0.5, hwdownload" OUTPUT
```

- ⊕ Stabilize a video with debugging (both in console and in rendered video):

```
-i INPUT -filter_complex "[0:v]format=rgba, hwupload, deshake_openc1=debug=1, hwdownload, format=rgb
```

dilation_openc1

Apply dilation effect to the video.

This filter replaces the pixel by the local(3x3) maximum.

It accepts the following options:

threshold0**threshold1****threshold2****threshold3**

Limit the maximum change for each plane. Range is "[0, 65535]" and default value is 65535. If 0, plane will remain unchanged.

coordinates

Flag which specifies the pixel to refer to. Range is "[0, 255]" and default value is 255, i.e. all eight pixels are used.

Flags to local 3x3 coordinates region centered on "x":

1 2 3

4 x 5

6 7 8

Example

- ⊕ Apply dilation filter with threshold0 set to 30, threshold1 set 40, threshold2 set to 50 and coordinates set to 231, setting each pixel of the output to the local maximum between pixels: 1, 2, 3, 6, 7, 8 of the 3x3 region centered on it in the input. If the difference between input pixel and local maximum is more then threshold of the corresponding plane, output pixel will be set to input pixel + threshold of corresponding plane.

```
-i INPUT -vf "hwupload, dilation_openc1=30:40:50:coordinates=231, hwdownload" OUTPUT
```

nlmeans_openc1

Non-local Means denoise filter through OpenCL, this filter accepts same options as **nlmeans**.

overlay_openc1

Overlay one video on top of another.

It takes two inputs and has one output. The first input is the "main" video on which the second input is overlaid. This filter requires same memory layout for all the inputs. So, format conversion may be needed.

The filter accepts the following options:

x Set the x coordinate of the overlaid video on the main video. Default value is 0.

y Set the y coordinate of the overlaid video on the main video. Default value is 0.

Examples

- ⊕ Overlay an image LOGO at the top-left corner of the INPUT video. Both inputs are yuv420p format.

```
-i INPUT -i LOGO -filter_complex "[0:v]hwupload[a], [1:v]format=yuv420p, hwupload[b], [a][b]overlay_0"
```

- ⊕ The inputs have same memory layout for color channels , the overlay has additional alpha plane, like INPUT is yuv420p, and the LOGO is yuva420p.

```
-i INPUT -i LOGO -filter_complex "[0:v]hwupload[a], [1:v]format=yuva420p, hwupload[b], [a][b]overlay_0"
```

pad_openc1

Add paddings to the input image, and place the original input at the provided *x*, *y* coordinates.

It accepts the following options:

width, w

height, h

Specify an expression for the size of the output image with the paddings added. If the value for *width* or *height* is 0, the corresponding input size is used for the output.

The *width* expression can reference the value set by the *height* expression, and vice versa.

The default value of *width* and *height* is 0.

x

y Specify the offsets to place the input image at within the padded area, with respect to the top/left border of the output image.

The *x* expression can reference the value set by the *y* expression, and vice versa.

The default value of *x* and *y* is 0.

If *x* or *y* evaluate to a negative number, they'll be changed so the input image is centered on the padded area.

color

Specify the color of the padded area. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**.

aspect

Pad to an aspect instead to a resolution.

The value for the *width*, *height*, *x*, and *y* options are expressions containing the following constants:

in_w**in_h**

The input video width and height.

iw

ih These are the same as *in_w* and *in_h*.

out_w**out_h**

The output width and height (the size of the padded area), as specified by the *width* and *height* expressions.

ow

oh These are the same as *out_w* and *out_h*.

x

y The x and y offsets as specified by the *x* and *y* expressions, or NAN if not yet specified.

a same as *iw / ih*

sar input sample aspect ratio

dar input display aspect ratio, it is the same as $(iw / ih) * sar$

prewitt_openc1

Apply the Prewitt operator (<https://en.wikipedia.org/wiki/Prewitt_operator>) to input video stream.

The filter accepts the following option:

planes

Set which planes to filter. Default value is 0xf, by which all planes are processed.

scale

Set value which will be multiplied with filtered result. Range is "[0.0, 65535]" and default value is 1.0.

delta

Set value which will be added to filtered result. Range is "[-65535, 65535]" and default value is 0.0.

Example

- ⊕ Apply the Prewitt operator with scale set to 2 and delta set to 10.

```
-i INPUT -vf "hwupload, prewitt_opencl=scale=2:delta=10, hwdownload" OUTPUT
```

program_opencl

Filter video using an OpenCL program.

source

OpenCL program source file.

kernel

Kernel name in program.

inputs

Number of inputs to the filter. Defaults to 1.

size, s

Size of output frames. Defaults to the same as the first input.

The "program_opencl" filter also supports the **framesync** options.

The program source file must contain a kernel function with the given name, which will be run once for each plane of the output. Each run on a plane gets enqueued as a separate 2D global NDRange with one work-item for each pixel to be generated. The global ID offset for each work-item is therefore the coordinates of a pixel in the destination image.

The kernel function needs to take the following arguments:

- ⊕ Destination image, *__write_only image2d_t*.

This image will become the output; the kernel should write all of it.

- ⊕ Frame index, *unsigned int*.

This is a counter starting from zero and increasing by one for each frame.

- ⊕ Source images, *__read_only image2d_t*.

These are the most recent images on each input. The kernel may read from them to generate the output, but they can't be written to.

Example programs:

- ⊕ Copy the input to the output (output must be the same size as the input).

```
__kernel void copy(__write_only image2d_t destination,
                  unsigned int index,
                  __read_only image2d_t source)
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE;

    int2 location = (int2)(get_global_id(0), get_global_id(1));

    float4 value = read_imagef(source, sampler, location);

    write_imagef(destination, location, value);
}
```

- ⊕ Apply a simple transformation, rotating the input by an amount increasing with the index counter. Pixel values are linearly interpolated by the sampler, and the output need not have the same dimensions as the input.

```
__kernel void rotate_image(__write_only image2d_t dst,
                          unsigned int index,
                          __read_only image2d_t src)
{
    const sampler_t sampler = (CLK_NORMALIZED_COORDS_FALSE |
                              CLK_FILTER_LINEAR);

    float angle = (float)index / 100.0f;
```



```

float2 dst_dim = convert_float2(get_image_dim(dst));
float2 src_dim = convert_float2(get_image_dim(src));

float2 dst_cen = dst_dim / 2.0f;
float2 src_cen = src_dim / 2.0f;

int2  dst_loc = (int2)(get_global_id(0), get_global_id(1));

float2 dst_pos = convert_float2(dst_loc) - dst_cen;
float2 src_pos = {
    cos(angle) * dst_pos.x - sin(angle) * dst_pos.y,
    sin(angle) * dst_pos.x + cos(angle) * dst_pos.y
};
src_pos = src_pos * src_dim / dst_dim;

float2 src_loc = src_pos + src_cen;

if (src_loc.x < 0.0f || src_loc.y < 0.0f ||
    src_loc.x > src_dim.x || src_loc.y > src_dim.y)
    write_imagef(dst, dst_loc, 0.5f);
else
    write_imagef(dst, dst_loc, read_imagef(src, sampler, src_loc));
}

```

- ⊕ Blend two inputs together, with the amount of each input used varying with the index counter.

```

__kernel void blend_images(__write_only image2d_t dst,
                          unsigned int index,
                          __read_only image2d_t src1,
                          __read_only image2d_t src2)
{
    const sampler_t sampler = (CLK_NORMALIZED_COORDS_FALSE |
                              CLK_FILTER_LINEAR);

    float blend = (cos((float)index / 50.0f) + 1.0f) / 2.0f;

    int2 dst_loc = (int2)(get_global_id(0), get_global_id(1));
    int2 src1_loc = dst_loc * get_image_dim(src1) / get_image_dim(dst);
    int2 src2_loc = dst_loc * get_image_dim(src2) / get_image_dim(dst);
}

```

```

float4 val1 = read_imagef(src1, sampler, src1_loc);
float4 val2 = read_imagef(src2, sampler, src2_loc);

write_imagef(dst, dst_loc, val1 * blend + val2 * (1.0f - blend));
}

```

remap_openc1

Remap pixels using 2nd: Xmap and 3rd: Ymap input video stream.

Destination pixel at position (X, Y) will be picked from source (x, y) position where $x = Xmap(X, Y)$ and $y = Ymap(X, Y)$. If mapping values are out of range, zero value for pixel will be used for destination pixel.

Xmap and Ymap input video streams must be of same dimensions. Output video stream will have Xmap/Ymap video stream dimensions. Xmap and Ymap input video streams are 32bit float pixel format, single channel.

interp

Specify interpolation used for remapping of pixels. Allowed values are "near" and "linear". Default value is "linear".

fill Specify the color of the unmapped pixels. For the syntax of this option, check the "**Color**" section in the **ffmpeg-utils manual**. Default color is "black".

roberts_openc1

Apply the Roberts cross operator (<https://en.wikipedia.org/wiki/Roberts_cross>) to input video stream.

The filter accepts the following option:

planes

Set which planes to filter. Default value is 0xf, by which all planes are processed.

scale

Set value which will be multiplied with filtered result. Range is "[0.0, 65535]" and default value is 1.0.

delta

Set value which will be added to filtered result. Range is "[-65535, 65535]" and default value is 0.0.

Example

- ⊕ Apply the Roberts cross operator with scale set to 2 and delta set to 10

```
-i INPUT -vf "hwupload, roberts_openc1=scale=2:delta=10, hwdownload" OUTPUT
```

sobel_openc1

Apply the Sobel operator (<https://en.wikipedia.org/wiki/Sobel_operator>) to input video stream.

The filter accepts the following option:

planes

Set which planes to filter. Default value is 0xf, by which all planes are processed.

scale

Set value which will be multiplied with filtered result. Range is "[0.0, 65535]" and default value is 1.0.

delta

Set value which will be added to filtered result. Range is "[-65535, 65535]" and default value is 0.0.

Example

- ⊕ Apply sobel operator with scale set to 2 and delta set to 10

```
-i INPUT -vf "hwupload, sobel_openc1=scale=2:delta=10, hwdownload" OUTPUT
```

tonemap_openc1

Perform HDR(PQ/HLG) to SDR conversion with tone-mapping.

It accepts the following parameters:

tonemap

Specify the tone-mapping operator to be used. Same as tonemap option in **tonemap**.

param

Tune the tone mapping algorithm. same as param option in **tonemap**.

desat

Apply desaturation for highlights that exceed this level of brightness. The higher the parameter, the more color information will be preserved. This setting helps prevent unnaturally blown-out colors for super-highlights, by (smoothly) turning into white instead. This makes images feel more natural, at the cost of reducing information about out-of-range colors.

The default value is 0.5, and the algorithm here is a little different from the cpu version tonemap currently. A setting of 0.0 disables this option.

threshold

The tonemapping algorithm parameters is fine-tuned per each scene. And a threshold is used to detect whether the scene has changed or not. If the distance between the current frame average brightness and the current running average exceeds a threshold value, we would re-calculate scene average and peak brightness. The default value is 0.2.

format

Specify the output pixel format.

Currently supported formats are:

p010

nv12

range, r

Set the output color range.

Possible values are:

tv/mpeg

pc/jpeg

Default is same as input.

primaries, p

Set the output color primaries.

Possible values are:

bt709

bt2020

Default is same as input.

transfer, t

Set the output transfer characteristics.

Possible values are:

bt709

bt2020

Default is bt709.

matrix, m

Set the output colorspace matrix.

Possible value are:

bt709

bt2020

Default is same as input.

Example

- ⊕ Convert HDR(PQ/HLG) video to bt2020-transfer-characteristic p010 format using linear operator.

```
-i INPUT -vf "format=p010,hwupload,tonemap_openc1=t=bt2020:tonemap=linear:format=p010,hwdownload"
```

unsharp_openc1

Sharpen or blur the input video.

It accepts the following parameters:

luma_msize_x, lx

Set the luma matrix horizontal size. Range is "[1, 23]" and default value is 5.

luma_msize_y, ly

Set the luma matrix vertical size. Range is "[1, 23]" and default value is 5.

luma_amount, la

Set the luma effect strength. Range is "[-10, 10]" and default value is 1.0.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

chroma_msize_x, cx

Set the chroma matrix horizontal size. Range is "[1, 23]" and default value is 5.

chroma_msize_y, cy

Set the chroma matrix vertical size. Range is "[1, 23]" and default value is 5.

chroma_amount, ca

Set the chroma effect strength. Range is "[-10, 10]" and default value is 0.0.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

All parameters are optional and default to the equivalent of the string '5:5:1.0:5:5:0.0'.

Examples

- ⊕ Apply strong luma sharpen effect:

```
-i INPUT -vf "hwupload, unsharp_opencl=luma_msize_x=7:luma_msize_y=7:luma_amount=2.5, hwdownload"
```

- ⊕ Apply a strong blur of both luma and chroma parameters:

```
-i INPUT -vf "hwupload, unsharp_opencl=7:7:-2:7:7:-2, hwdownload" OUTPUT
```

xfade_opencl

Cross fade two videos with custom transition effect by using OpenCL.

It accepts the following options:

transition

Set one of possible transition effects.

custom

Select custom transition effect, the actual transition description will be picked from source and kernel options.

fade

wipeleft
wiperight
wipeup
wipedown
slideleft
slideright
slideup
slidedown

Default transition is fade.

source

OpenCL program source file for custom transition.

kernel

Set name of kernel to use for custom transition from program source file.

duration

Set duration of video transition.

offset

Set time of start of transition relative to first video.

The program source file must contain a kernel function with the given name, which will be run once for each plane of the output. Each run on a plane gets enqueued as a separate 2D global NDRange with one work-item for each pixel to be generated. The global ID offset for each work-item is therefore the coordinates of a pixel in the destination image.

The kernel function needs to take the following arguments:

- ⊕ Destination image, *__write_only image2d_t*.

This image will become the output; the kernel should write all of it.

- ⊕ First Source image, *__read_only image2d_t*. Second Source image, *__read_only image2d_t*.

These are the most recent images on each input. The kernel may read from them to generate the output, but they can't be written to.

- ⊕ Transition progress, *float*. This value is always between 0 and 1 inclusive.

Example programs:

- ⊕ Apply dots curtain transition effect:

```
__kernel void blend_images(__write_only image2d_t dst,
                          __read_only image2d_t src1,
                          __read_only image2d_t src2,
                          float progress)
{
    const sampler_t sampler = (CLK_NORMALIZED_COORDS_FALSE |
                              CLK_FILTER_LINEAR);
    int2 p = (int2)(get_global_id(0), get_global_id(1));
    float2 rp = (float2)(get_global_id(0), get_global_id(1));
    float2 dim = (float2)(get_image_dim(src1).x, get_image_dim(src1).y);
    rp = rp / dim;

    float2 dots = (float2)(20.0, 20.0);
    float2 center = (float2)(0,0);
    float2 unused;

    float4 val1 = read_imagef(src1, sampler, p);
    float4 val2 = read_imagef(src2, sampler, p);
    bool next = distance(fract(rp * dots, &unused), (float2)(0.5, 0.5)) < (progress / distance(rp, center));

    write_imagef(dst, p, next ? val1 : val2);
}
```

VAAPI VIDEO FILTERS

VAAPI Video filters are usually used with VAAPI decoder and VAAPI encoder. Below is a description of VAAPI video filters.

To enable compilation of these filters you need to configure FFmpeg with "--enable-vaapi".

To use vaapi filters, you need to setup the vaapi device correctly. For more information, please read <<https://trac.ffmpeg.org/wiki/Hardware/VAAPI>>

overlay_vaapi

Overlay one video on the top of another.

It takes two inputs and has one output. The first input is the "main" video on which the second input is

overlaid.

The filter accepts the following options:

x

y Set expressions for the x and y coordinates of the overlaid video on the main video.

Default value is "0" for both expressions.

w

h Set expressions for the width and height the overlaid video on the main video.

Default values are 'overlay_iw' for 'w' and 'overlay_ih*w/overlay_iw' for 'h'.

The expressions can contain the following parameters:

main_w, W

main_h, H

The main input width and height.

overlay_iw

overlay_ih

The overlay input width and height.

overlay_w, w

overlay_h, h

The overlay output width and height.

overlay_x, x

overlay_y, y

Position of the overlay layer inside of main

alpha

Set transparency of overlaid video. Allowed range is 0.0 to 1.0. Higher value means lower transparency. Default value is 1.0.

eof_action

See **framesync**.

shortest

See **framesync**.

repeatlast

See **framesync**.

This filter also supports the **framesync** options.

Examples

- ⊕ Overlay an image LOGO at the top-left corner of the INPUT video. Both inputs for this filter are yuv420p format.

```
-i INPUT -i LOGO -filter_complex "[0:v]hwupload[a], [1:v]format=yuv420p, hwupload[b], [a][b]overlay_va
```

- ⊕ Overlay an image LOGO at the offset (200, 100) from the top-left corner of the INPUT video. The inputs have same memory layout for color channels, the overlay has additional alpha plane, like INPUT is yuv420p, and the LOGO is yuva420p.

```
-i INPUT -i LOGO -filter_complex "[0:v]hwupload[a], [1:v]format=yuva420p, hwupload[b], [a][b]overlay_v
```

tonemap_vaapi

Perform HDR(High Dynamic Range) to SDR(Standard Dynamic Range) conversion with tone-mapping. It maps the dynamic range of HDR10 content to the SDR content. It currently only accepts HDR10 as input.

It accepts the following parameters:

format

Specify the output pixel format.

Currently supported formats are:

p010

nv12

Default is nv12.

primaries, p

Set the output color primaries.

Default is same as input.

transfer, t

Set the output transfer characteristics.

Default is bt709.

matrix, m

Set the output colorspace matrix.

Default is same as input.

Example

- ⊕ Convert HDR(HDR10) video to bt2020-transfer-characteristic p010 format

```
tonemap_vaapi=format=p010:t=bt2020-10
```

hstack_vaapi

Stack input videos horizontally.

This is the VA-API variant of the **hstack** filter, each input stream may have different height, this filter will scale down/up each input stream while keeping the original aspect.

It accepts the following options:

inputs

See **hstack**.

shortest

See **hstack**.

height

Set height of output. If set to 0, this filter will set height of output to height of the first input stream. Default value is 0.

vstack_vaapi

Stack input videos vertically.

This is the VA-API variant of the **vstack** filter, each input stream may have different width, this filter

will scale down/up each input stream while keeping the original aspect.

It accepts the following options:

inputs

See **vstack**.

shortest

See **vstack**.

width

Set width of output. If set to 0, this filter will set width of output to width of the first input stream. Default value is 0.

xstack_vaapi

Stack video inputs into custom layout.

This is the VA-API variant of the **xstack** filter, each input stream may have different size, this filter will scale down/up each input stream to the given output size, or the size of the first input stream.

It accepts the following options:

inputs

See **xstack**.

shortest

See **xstack**.

layout

See **xstack**. Moreover, this permits the user to supply output size for each input stream.

```
xstack_vaapi=inputs=4:layout=0_0_1920x1080|0_h0_1920x1080|w0_0_1920x1080|w0_h0_1920x1080
```

grid See **xstack**.

grid_tile_size

Set output size for each input stream when **grid** is set. If this option is not set, this filter will set output size by default to the size of the first input stream. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

fill See **xstack**.

QSV VIDEO FILTERS

Below is a description of the currently available QSV video filters.

To enable compilation of these filters you need to configure FFmpeg with "--enable-libmfx" or "--enable-libvpl".

To use QSV filters, you need to setup the QSV device correctly. For more information, please read <<https://trac.ffmpeg.org/wiki/Hardware/QuickSync>>

hstack_qsv

Stack input videos horizontally.

This is the QSV variant of the **hstack** filter, each input stream may have different height, this filter will scale down/up each input stream while keeping the original aspect.

It accepts the following options:

inputs

See **hstack**.

shortest

See **hstack**.

height

Set height of output. If set to 0, this filter will set height of output to height of the first input stream. Default value is 0.

vstack_qsv

Stack input videos vertically.

This is the QSV variant of the **vstack** filter, each input stream may have different width, this filter will scale down/up each input stream while keeping the original aspect.

It accepts the following options:

inputs

See **vstack**.

shortest

See **vstack**.

width

Set width of output. If set to 0, this filter will set width of output to width of the first input stream. Default value is 0.

xstack_qsv

Stack video inputs into custom layout.

This is the QSV variant of the **xstack** filter.

It accepts the following options:

inputs

See **xstack**.

shortest

See **xstack**.

layout

See **xstack**. Moreover, this permits the user to supply output size for each input stream.

```
xstack_qsv=inputs=4:layout=0_0_1920x1080|0_h0_1920x1080|w0_0_1920x1080|w0_h0_1920x1080
```

grid See **xstack**.

grid_tile_size

Set output size for each input stream when **grid** is set. If this option is not set, this filter will set output size by default to the size of the first input stream. For the syntax of this option, check the **"Video size" section in the ffmpeg-utils manual**.

fill See **xstack**.

VIDEO SOURCES

Below is a description of the currently available video sources.

buffer

Buffer video frames, and make them available to the filter chain.

This source is mainly intended for a programmatic use, in particular through the interface defined in *libavfilter/buffersrc.h*.

It accepts the following parameters:

video_size

Specify the size (width and height) of the buffered video frames. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

width

The input video width.

height

The input video height.

pix_fmt

A string representing the pixel format of the buffered video frames. It may be a number corresponding to a pixel format, or a pixel format name.

time_base

Specify the timebase assumed by the timestamps of the buffered frames.

frame_rate

Specify the frame rate expected for the video stream.

pixel_aspect, sar

The sample (pixel) aspect ratio of the input video.

hw_frames_ctx

When using a hardware pixel format, this should be a reference to an AVHWFramesContext describing input frames.

For example:

```
buffer=width=320:height=240:pix_fmt=yuv410p:time_base=1/24:sar=1
```

will instruct the source to accept video frames with size 320x240 and with format "yuv410p", assuming 1/24 as the timestamps timebase and square pixels (1:1 sample aspect ratio). Since the pixel format with name "yuv410p" corresponds to the number 6 (check the enum AVPixelFormat definition in *libavutil/pixfmt.h*), this example corresponds to:

```
buffer=size=320x240:pixfmt=6:time_base=1/24:pixel_aspect=1/1
```

Alternatively, the options can be specified as a flat string, but this syntax is deprecated:

```
width:height:pix_fmt:time_base.num:time_base.den:pixel_aspect.num:pixel_aspect.den
```

cellauto

Create a pattern generated by an elementary cellular automaton.

The initial state of the cellular automaton can be defined through the **filename** and **pattern** options. If such options are not specified an initial state is created randomly.

At each new frame a new row in the video is filled with the result of the cellular automaton next generation. The behavior when the whole frame is filled is defined by the **scroll** option.

This source accepts the following options:

filename, f

Read the initial cellular automaton state, i.e. the starting row, from the specified file. In the file, each non-whitespace character is considered an alive cell, a newline will terminate the row, and further characters in the file will be ignored.

pattern, p

Read the initial cellular automaton state, i.e. the starting row, from the specified string.

Each non-whitespace character in the string is considered an alive cell, a newline will terminate the row, and further characters in the string will be ignored.

rate, r

Set the video rate, that is the number of frames generated per second. Default is 25.

random_fill_ratio, ratio

Set the random fill ratio for the initial cellular automaton row. It is a floating point number value ranging from 0 to 1, defaults to 1/PHI.

This option is ignored when a file or a pattern is specified.

random_seed, seed

Set the seed for filling randomly the initial row, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random

seed on a best effort basis.

rule Set the cellular automaton rule, it is a number ranging from 0 to 255. Default value is 110.

size, s

Set the size of the output video. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

If **filename** or **pattern** is specified, the size is set by default to the width of the specified initial state row, and the height is set to *width* * PHI.

If **size** is set, it must contain the width of the specified pattern string, and the specified pattern will be centered in the larger row.

If a filename or a pattern string is not specified, the size value defaults to "320x518" (used for a randomly generated initial state).

scroll

If set to 1, scroll the output upward when all the rows in the output have been already filled. If set to 0, the new generated row will be written over the top row just after the bottom row is filled. Defaults to 1.

start_full, full

If set to 1, completely fill the output with generated rows before outputting the first frame. This is the default behavior, for disabling set the value to 0.

stitch

If set to 1, stitch the left and right row edges together. This is the default behavior, for disabling set the value to 0.

Examples

- ⊕ Read the initial state from *pattern*, and specify an output of size 200x400.

```
cellauto=f=pattern:s=200x400
```

- ⊕ Generate a random initial row with a width of 200 cells, with a fill ratio of 2/3:

```
cellauto=ratio=2/3:s=200x200
```

- ⊕ Create a pattern generated by rule 18 starting by a single alive cell centered on an initial row with width 100:

```
cellauto=p=@s=100x400:full=0:rule=18
```

- ⊕ Specify a more elaborated initial pattern:

```
cellauto=p='@@ @ @@@':s=100x400:full=0:rule=18
```

coreimagesrc

Video source generated on GPU using Apple's CoreImage API on OSX.

This video source is a specialized version of the **coreimage** video filter. Use a core image generator at the beginning of the applied filterchain to generate the content.

The coreimagesrc video source accepts the following options:

list_generators

List all available generators along with all their respective options as well as possible minimum and maximum values along with the default values.

```
list_generators=true
```

size, s

Specify the size of the sourced video. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. The default value is "320x240".

rate, r

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame_rate_num/frame_rate_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

sar Set the sample aspect ratio of the sourced video.

duration, d

Set the duration of the sourced video. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

Additionally, all options of the **coreimage** video filter are accepted. A complete filterchain can be used for further processing of the generated input without CPU-HOST transfer. See **coreimage** documentation and examples for details.

Examples

- ⊕ Use CIQRCodeGenerator to create a QR code for the FFmpeg homepage, given as complete and escaped command-line for Apple's standard bash shell:

```
ffmpeg -f lavfi -i coreimagesrc=s=100x100:filter=CIQRCodeGenerator@inputMessage=https://FFmpeg.org
```

This example is equivalent to the QRCode example of **coreimage** without the need for a nullsrc video source.

ddagrab

Captures the Windows Desktop via Desktop Duplication API.

The filter exclusively returns D3D11 Hardware Frames, for on-gpu encoding or processing. So an explicit **hwdownload** is needed for any kind of software processing.

It accepts the following options:

output_idx

DXGI Output Index to capture.

Usually corresponds to the index Windows has given the screen minus one, so it's starting at 0.

Defaults to output 0.

draw_mouse

Whether to draw the mouse cursor.

Defaults to true.

Only affects hardware cursors. If a game or application renders its own cursor, it'll always be captured.

framerate

Framerate at which the desktop will be captured.

Defaults to 30 FPS.

video_size

Specify the size of the captured video.

Defaults to the full size of the screen.

Cropped from the bottom/right if smaller than screen size.

offset_x

Horizontal offset of the captured video.

offset_y

Vertical offset of the captured video.

output_fmt

Desired filter output format. Defaults to 8 Bit BGRA.

It accepts the following values:

auto Passes all supported output formats to DDA and returns what DDA decides to use.

8bit**bgra**

8 Bit formats always work, and DDA will convert to them if necessary.

10bit**x2bgr10**

Filter initialization will fail if 10 bit format is requested but unavailable.

Examples

Capture primary screen and encode using nvenc:

```
ffmpeg -f lavfi -i ddagrab -c:v h264_nvenc -cq 18 output.mp4
```

You can also skip the lavfi device and directly use the filter. Also demonstrates downloading the frame and encoding with libx264. Explicit output format specification is required in this case:

```
ffmpeg -filter_complex ddagrab=output_idx=1:framerate=60,hwdownload,format=bgra -c:v libx264 -crf 18 output.mp4
```

If you want to capture only a subsection of the desktop, this can be achieved by specifying a smaller size and its offsets into the screen:

```
ddagrab=video_size=800x600:offset_x=100:offset_y=100
```

gradients

Generate several gradients.

size, s

Set frame size. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "640x480".

rate, r

Set frame rate, expressed as number of frames per second. Default value is "25".

c0, c1, c2, c3, c4, c5, c6, c7

Set 8 colors. Default values for colors is to pick random one.

x0, y0, y0, y1

Set gradient line source and destination points. If negative or out of range, random ones are picked.

nb_colors, n

Set number of colors to use at once. Allowed range is from 2 to 8. Default value is 2.

seed

Set seed for picking gradient line points.

duration, d

Set the duration of the sourced video. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

speed

Set speed of gradients rotation.

type, t

Set type of gradients, can be "linear" or "radial" or "circular" or "spiral".

mandelbrot

Generate a Mandelbrot set fractal, and progressively zoom towards the point specified with *start_x* and *start_y*.

This source accepts the following options:

end_pts

Set the terminal pts value. Default value is 400.

end_scale

Set the terminal scale value. Must be a floating point value. Default value is 0.3.

inner

Set the inner coloring mode, that is the algorithm used to draw the Mandelbrot fractal internal region.

It shall assume one of the following values:

black

Set black mode.

convergence

Show time until convergence.

mincol

Set color based on point closest to the origin of the iterations.

period

Set period mode.

Default value is *mincol*.

bailout

Set the bailout value. Default value is 10.0.

maxiter

Set the maximum of iterations performed by the rendering algorithm. Default value is 7189.

outer

Set outer coloring mode. It shall assume one of following values:

iteration_count

Set iteration count mode.

normalized_iteration_count

set normalized iteration count mode.

Default value is *normalized_iteration_count*.

rate, r

Set frame rate, expressed as number of frames per second. Default value is "25".

size, s

Set frame size. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "640x480".

start_scale

Set the initial scale value. Default value is 3.0.

start_x

Set the initial x position. Must be a floating point value between -100 and 100. Default value is -0.743643887037158704752191506114774.

start_y

Set the initial y position. Must be a floating point value between -100 and 100. Default value is -0.131825904205311970493132056385139.

mptestsrc

Generate various test patterns, as generated by the MPlayer test filter.

The size of the generated video is fixed, and is 256x256. This source is useful in particular for testing encoding features.

This source accepts the following options:

rate, r

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame_rate_num/frame_rate_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

duration, d

Set the duration of the sourced video. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

test, t

Set the number or the name of the test to perform. Supported tests are:

dc_luma

dc_chroma

freq_luma

freq_chroma

amp_luma

amp_chroma

cbp

mv

ring1

ring2

all

max_frames, m

Set the maximum number of frames generated for each test, default value is 30.

Default value is "all", which will cycle through the list of all tests.

Some examples:

```
mptestsrc=t=dc_luma
```

will generate a "dc_luma" test pattern.

frei0r_src

Provide a frei0r source.

To enable compilation of this filter you need to install the frei0r header and configure FFmpeg with "--enable-frei0r".

This source accepts the following parameters:

size The size of the video to generate. For the syntax of this option, check the **"Video size" section in**

the ffmpeg-utils manual.**framerate**

The framerate of the generated video. It may be a string of the form *num/den* or a frame rate abbreviation.

filter_name

The name to the frei0r source to load. For more information regarding frei0r and how to set the parameters, read the **frei0r** section in the video filters documentation.

filter_params

A '|' -separated list of parameters to pass to the frei0r source.

For example, to generate a frei0r partik0l source with size 200x200 and frame rate 10 which is overlaid on the overlay filter main input:

```
frei0r_src=size=200x200:framerate=10:filter_name=partik0l:filter_params=1234 [overlay]; [in][overlay] overlay
```

life

Generate a life pattern.

This source is based on a generalization of John Conway's life game.

The sourced input represents a life grid, each pixel represents a cell which can be in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent.

At each interaction the grid evolves according to the adopted rule, which specifies the number of neighbor alive cells which will make a cell stay alive or born. The **rule** option allows one to specify the rule to adopt.

This source accepts the following options:

filename, f

Set the file from which to read the initial grid state. In the file, each non-whitespace character is considered an alive cell, and newline is used to delimit the end of each row.

If this option is not specified, the initial grid is generated randomly.

rate, r

Set the video rate, that is the number of frames generated per second. Default is 25.

random_fill_ratio, ratio

Set the random fill ratio for the initial random grid. It is a floating point number value ranging from 0 to 1, defaults to 1/PHI. It is ignored when a file is specified.

random_seed, seed

Set the seed for filling the initial random grid, must be an integer included between 0 and UINT32_MAX. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

rule Set the life rule.

A rule can be specified with a code of the kind "SNS/BNB", where *NS* and *NB* are sequences of numbers in the range 0-8, *NS* specifies the number of alive neighbor cells which make a live cell stay alive, and *NB* the number of alive neighbor cells which make a dead cell to become alive (i.e. to "born"). "s" and "b" can be used in place of "S" and "B", respectively.

Alternatively a rule can be specified by an 18-bits integer. The 9 high order bits are used to encode the next cell state if it is alive for each number of neighbor alive cells, the low order bits specify the rule for "borning" new cells. Higher order bits encode for an higher number of neighbor cells. For example the number 6153 = "(12<<9)+9" specifies a stay alive rule of 12 and a born rule of 9, which corresponds to "S23/B03".

Default value is "S23/B3", which is the original Conway's game of life rule, and will keep a cell alive if it has 2 or 3 neighbor alive cells, and will born a new cell if there are three alive cells around a dead cell.

size, s

Set the size of the output video. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**.

If **filename** is specified, the size is set by default to the same size of the input file. If **size** is set, it must contain the size specified in the input file, and the initial grid defined in that file is centered in the larger resulting area.

If a filename is not specified, the size value defaults to "320x240" (used for a randomly generated initial grid).

stitch

If set to 1, stitch the left and right grid edges together, and the top and bottom edges also. Defaults to 1.

mold

Set cell mold speed. If set, a dead cell will go from **death_color** to **mold_color** with a step of **mold**. **mold** can have a value from 0 to 255.

life_color

Set the color of living (or new born) cells.

death_color

Set the color of dead cells. If **mold** is set, this is the first color used to represent a dead cell.

mold_color

Set mold color, for definitely dead and moldy cells.

For the syntax of these 3 color options, check the "**Color**" section in the **ffmpeg-utils manual**.

Examples

- ⊕ Read a grid from *pattern*, and center it on a grid of size 300x300 pixels:

```
life=f=pattern:s=300x300
```

- ⊕ Generate a random grid of size 200x200, with a fill ratio of 2/3:

```
life=ratio=2/3:s=200x200
```

- ⊕ Specify a custom rule for evolving a randomly generated grid:

```
life=rule=S14/B34
```

- ⊕ Full example with slow death effect (mold) using **ffplay**:

```
ffplay -f lavfi life=s=300x200:mold=10:r=60:ratio=0.1:death_color=#C83232:life_color=#00ff00,scale=1200
```

allrgb, allyuv, color, colorchart, colorspectrum, haldclutsrc, nullsrc, pal75bars, pal100bars, rgbtestsrc, smptebars, smptehdbars, testsrc, testsrc2, yuvtestsrc

The "allrgb" source returns frames of size 4096x4096 of all rgb colors.

The "allyuv" source returns frames of size 4096x4096 of all yuv colors.

The "color" source provides an uniformly colored input.

The "colorchart" source provides a colors checker chart.

The "colourspectrum" source provides a color spectrum input.

The "haldclutsrc" source provides an identity Hald CLUT. See also **haldclut** filter.

The "nullsrc" source returns unprocessed video frames. It is mainly useful to be employed in analysis / debugging tools, or as the source for filters which ignore the input data.

The "pal75bars" source generates a color bars pattern, based on EBU PAL recommendations with 75% color levels.

The "pal100bars" source generates a color bars pattern, based on EBU PAL recommendations with 100% color levels.

The "rgbtstsrc" source generates an RGB test pattern useful for detecting RGB vs BGR issues. You should see a red, green and blue stripe from top to bottom.

The "smpthebars" source generates a color bars pattern, based on the SMPTE Engineering Guideline EG 1-1990.

The "smptehdbars" source generates a color bars pattern, based on the SMPTE RP 219-2002.

The "testsrc" source generates a test video pattern, showing a color pattern, a scrolling gradient and a timestamp. This is mainly intended for testing purposes.

The "testsrc2" source is similar to testsrc, but supports more pixel formats instead of just "rgb24". This allows using it as an input for other tests without requiring a format conversion.

The "yuvtestsrc" source generates an YUV test pattern. You should see a y, cb and cr stripe from top to bottom.

The sources accept the following parameters:

level

Specify the level of the Hald CLUT, only available in the "haldclutsrc" source. A level of "N"

generates a picture of "N*N*N" by "N*N*N" pixels to be used as identity matrix for 3D lookup tables. Each component is coded on a "1/(N*N)" scale.

color, c

Specify the color of the source, only available in the "color" source. For the syntax of this option, check the **"Color" section in the ffmpeg-utils manual**.

size, s

Specify the size of the sourced video. For the syntax of this option, check the **"Video size" section in the ffmpeg-utils manual**. The default value is "320x240".

This option is not available with the "allrgb", "allyuv", and "haldclutsrc" filters.

rate, r

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame_rate_num/frame_rate_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

duration, d

Set the duration of the sourced video. See **the Time duration section in the ffmpeg-utils(1) manual** for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

Since the frame rate is used as time base, all frames including the last one will have their full duration. If the specified duration is not a multiple of the frame duration, it will be rounded up.

sar Set the sample aspect ratio of the sourced video.

alpha

Specify the alpha (opacity) of the background, only available in the "testsrc2" source. The value must be between 0 (fully transparent) and 255 (fully opaque, the default).

decimals, n

Set the number of decimals to show in the timestamp, only available in the "testsrc" source.

The displayed timestamp value will correspond to the original timestamp value multiplied by the power of 10 of the specified value. Default value is 0.

type Set the type of the color spectrum, only available in the "colourspectrum" source. Can be one of the following:

black

white

all

patch_size

Set patch size of single color patch, only available in the "colorchart" source. Default is "64x64".

preset

Set colorchecker colors preset, only available in the "colorchart" source.

Available values are:

reference

skintones

Default value is "reference".

Examples

- ⊕ Generate a video with a duration of 5.3 seconds, with size 176x144 and a frame rate of 10 frames per second:

```
testsrc=duration=5.3:size=qcif:rate=10
```

- ⊕ The following graph description will generate a red source with an opacity of 0.2, with size "qcif" and a frame rate of 10 frames per second:

```
color=c=red@0.2:s=qcif:r=10
```

- ⊕ If the input content is to be ignored, "nullsrc" can be used. The following command generates noise in the luminance plane by employing the "geq" filter:

```
nullsrc=s=256x256, geq=random(1)*255:128:128
```

Commands

The "color" source supports the following commands:

c, color

Set the color of the created image. Accepts the same syntax of the corresponding **color** option.

openclsrc

Generate video using an OpenCL program.

source

OpenCL program source file.

kernel

Kernel name in program.

size, s

Size of frames to generate. This must be set.

format

Pixel format to use for the generated frames. This must be set.

rate, r

Number of frames generated every second. Default value is '25'.

For details of how the program loading works, see the **program_openc1** filter.

Example programs:

- ⊕ Generate a colour ramp by setting pixel values from the position of the pixel in the output image. (Note that this will work with all pixel formats, but the generated output will not be the same.)

```
__kernel void ramp(__write_only image2d_t dst,
                  unsigned int index)
{
    int2 loc = (int2)(get_global_id(0), get_global_id(1));

    float4 val;
    val.xy = val.zw = convert_float2(loc) / convert_float2(get_image_dim(dst));

    write_imagef(dst, loc, val);
}
```

- ⊕ Generate a Sierpinski carpet pattern, panning by a single pixel each frame.

```

__kernel void sierpinski_carpet(__write_only image2d_t dst,
                                unsigned int index)
{
    int2 loc = (int2)(get_global_id(0), get_global_id(1));

    float4 value = 0.0f;
    int x = loc.x + index;
    int y = loc.y + index;
    while (x > 0 || y > 0) {
        if (x % 3 == 1 && y % 3 == 1) {
            value = 1.0f;
            break;
        }
        x /= 3;
        y /= 3;
    }

    write_imagef(dst, loc, value);
}

```

sierpinski

Generate a Sierpinski carpet/triangle fractal, and randomly pan around.

This source accepts the following options:

size, s

Set frame size. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "640x480".

rate, r

Set frame rate, expressed as number of frames per second. Default value is "25".

seed

Set seed which is used for random panning.

jump

Set max jump for single pan destination. Allowed range is from 1 to 10000.

type Set fractal type, can be default "carpet" or "triangle".

VIDEO SINKS

Below is a description of the currently available video sinks.

buffersink

Buffer video frames, and make them available to the end of the filter graph.

This sink is mainly intended for programmatic use, in particular through the interface defined in *libavfilter/buffersink.h* or the options system.

It accepts a pointer to an AVBufferSinkContext structure, which defines the incoming buffers' formats, to be passed as the opaque parameter to "avfilter_init_filter" for initialization.

nullsink

Null video sink: do absolutely nothing with the input video. It is mainly useful as a template and for use in analysis / debugging tools.

MULTIMEDIA FILTERS

Below is a description of the currently available multimedia filters.

a3dscope

Convert input audio to 3d scope video output.

The filter accepts the following options:

rate, r

Set frame rate, expressed as number of frames per second. Default value is "25".

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "hd720".

fov Set the camera field of view. Default is 90 degrees. Allowed range is from 40 to 150.

roll Set the camera roll.

pitch

Set the camera pitch.

yaw Set the camera yaw.

xzoom

Set the camera zoom on X-axis.

yzoom

Set the camera zoom on Y-axis.

zzoom

Set the camera zoom on Z-axis.

xpos

Set the camera position on X-axis.

ypos

Set the camera position on Y-axis.

zpos

Set the camera position on Z-axis.

length

Set the length of displayed audio waves in number of frames.

Commands

Filter supports the some above options as **commands**.

abitscope

Convert input audio to a video output, displaying the audio bit scope.

The filter accepts the following options:

rate, r

Set frame rate, expressed as number of frames per second. Default value is "25".

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "1024x256".

colors

Specify list of colors separated by space or by `|` which will be used to draw channels. Unrecognized or missing colors will be replaced by white color.

mode, m

Set output mode. Can be "bars" or "trace". Default is "bars".

adrawgraph

Draw a graph using input audio metadata.

See **drawgraph**

agraphmonitor

See **graphmonitor**.

ahistogram

Convert input audio to a video output, displaying the volume histogram.

The filter accepts the following options:

dmode

Specify how histogram is calculated.

It accepts the following values:

single

Use single histogram for all channels.

separate

Use separate histogram for each channel.

Default is "single".

rate, r

Set frame rate, expressed as number of frames per second. Default value is "25".

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "hd720".

scale

Set display scale.

It accepts the following values:

log logarithmic

sqrt square root

cbrt cubic root

lin linear

rlog reverse logarithmic

Default is "log".

ascale

Set amplitude scale.

It accepts the following values:

log logarithmic

lin linear

Default is "log".

acount

Set how much frames to accumulate in histogram. Default is 1. Setting this to -1 accumulates all frames.

rheight

Set histogram ratio of window height.

slide

Set sonogram sliding.

It accepts the following values:

replace

replace old rows with new ones.

scroll

scroll from top to bottom.

Default is "replace".

hmode

Set histogram mode.

It accepts the following values:

abs Use absolute values of samples.

sign Use untouched values of samples.

Default is "abs".

aphasemeter

Measures phase of input audio, which is exported as metadata "lavfi.aphasemeter.phase", representing mean phase of current audio frame. A video output can also be produced and is enabled by default. The audio is passed through as first output.

Audio will be rematrixed to stereo if it has a different channel layout. Phase value is in range "[-1, 1]" where "-1" means left and right channels are completely out of phase and 1 means channels are in phase.

The filter accepts the following options, all related to its video output:

rate, r

Set the output frame rate. Default value is 25.

size, s

Set the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "800x400".

rc**gc**

bc Specify the red, green, blue contrast. Default values are 2, 7 and 1. Allowed range is "[0, 255]".

mpc Set color which will be used for drawing median phase. If color is "none" which is default, no median phase value will be drawn.

video

Enable video output. Default is enabled.

phasing detection

The filter also detects out of phase and mono sequences in stereo streams. It logs the sequence start, end and duration when it lasts longer or as long as the minimum set.

The filter accepts the following options for this detection:

phasing

Enable mono and out of phase detection. Default is disabled.

tolerance, t

Set phase tolerance for mono detection, in amplitude ratio. Default is 0. Allowed range is "[0, 1)".

angle, a

Set angle threshold for out of phase detection, in degree. Default is 170. Allowed range is "[90, 180)".

duration, d

Set mono or out of phase duration until notification, expressed in seconds. Default is 2.

Examples

- ⊕ Complete example with **ffmpeg** to detect 1 second of mono with 0.001 phase tolerance:

```
ffmpeg -i stereo.wav -af aphasemeter=video=0:phasing=1:duration=1:tolerance=0.001 -f null -
```

avectorscope

Convert input audio to a video output, representing the audio vector scope.

The filter is used to measure the difference between channels of stereo audio stream. A monaural signal, consisting of identical left and right signal, results in straight vertical line. Any stereo separation is visible as a deviation from this line, creating a Lissajous figure. If the straight (or deviation from it) but horizontal line appears this indicates that the left and right channels are out of phase.

The filter accepts the following options:

mode, m

Set the vectorscope mode.

Available values are:

lissajous

Lissajous rotated by 45 degrees.

lissajous_xy

Same as above but not rotated.

polar

Shape resembling half of circle.

Default value is **lissajous**.

size, s

Set the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "400x400".

rate, r

Set the output frame rate. Default value is 25.

rc**gc****bc**

ac Specify the red, green, blue and alpha contrast. Default values are 40, 160, 80 and 255. Allowed range is "[0, 255]".

rf**gf****bf**

af Specify the red, green, blue and alpha fade. Default values are 15, 10, 5 and 5. Allowed range is "[0, 255]".

zoom

Set the zoom factor. Default value is 1. Allowed range is "[0, 10]". Values lower than 1 will auto adjust zoom factor to maximal possible value.

draw

Set the vectorscope drawing mode.

Available values are:

dot Draw dot for each sample.

line Draw line between previous and current sample.

aaline

Draw anti-aliased line between previous and current sample.

Default value is **dot**.

scale

Specify amplitude scale of audio samples.

Available values are:

lin Linear.

sqrt Square root.

cbt Cubic root.

log Logarithmic.

swap

Swap left channel axis with right channel axis.

mirror

Mirror axis.

none

No mirror.

x Mirror only x axis.

y Mirror only y axis.

xy Mirror both axis.

Examples

⊕ Complete example using **ffplay**:

```
ffplay -f lavfi 'amovie=input.mp3, asplit [a][out1];
```



```
[a] avectorscope=zoom=1.3:rc=2:gc=200:bc=10:rf=1:gf=8:bf=7 [out0]'
```

Commands

This filter supports the all above options as commands except options "size" and "rate".

bench, abench

Benchmark part of a filtergraph.

The filter accepts the following options:

action

Start or stop a timer.

Available values are:

start Get the current time, set it as frame metadata (using the key "lavfi.bench.start_time"), and forward the frame to the next filter.

stop Get the current time and fetch the "lavfi.bench.start_time" metadata from the input frame metadata to get the time difference. Time difference, average, maximum and minimum time (respectively "t", "avg", "max" and "min") are then printed. The timestamps are expressed in seconds.

Examples

⊕ Benchmark **selectivecolor** filter:

```
bench=start,selectivecolor=reds=-.2 .12 -.49,bench=stop
```

concat

Concatenate audio and video streams, joining them together one after the other.

The filter works on segments of synchronized video and audio streams. All segments must have the same number of streams of each type, and that will also be the number of streams at output.

The filter accepts the following options:

n Set the number of segments. Default is 2.

- v** Set the number of output video streams, that is also the number of video streams in each segment. Default is 1.
- a** Set the number of output audio streams, that is also the number of audio streams in each segment. Default is 0.

unsafe

Activate unsafe mode: do not fail if segments have a different format.

The filter has $v+a$ outputs: first v video outputs, then a audio outputs.

There are $nx(v+a)$ inputs: first the inputs for the first segment, in the same order as the outputs, then the inputs for the second segment, etc.

Related streams do not always have exactly the same duration, for various reasons including codec frame size or sloppy authoring. For that reason, related synchronized streams (e.g. a video and its audio track) should be concatenated at once. The concat filter will use the duration of the longest stream in each segment (except the last one), and if necessary pad shorter audio streams with silence.

For this filter to work correctly, all segments must start at timestamp 0.

All corresponding streams must have the same parameters in all segments; the filtering system will automatically select a common pixel format for video streams, and a common sample format, sample rate and channel layout for audio streams, but other settings, such as resolution, must be converted explicitly by the user.

Different frame rates are acceptable but will result in variable frame rate at output; be sure to configure the output file to handle it.

Examples

- ⊕ Concatenate an opening, an episode and an ending, all in bilingual version (video in stream 0, audio in streams 1 and 2):

```
ffmpeg -i opening.mkv -i episode.mkv -i ending.mkv -filter_complex \
'[0:0] [0:1] [0:2] [1:0] [1:1] [1:2] [2:0] [2:1] [2:2] \
concat=n=3:v=1:a=2 [v] [a1] [a2]' \
-map '[v]' -map '[a1]' -map '[a2]' output.mkv
```

- ⊕ Concatenate two parts, handling audio and video separately, using the (a)movie sources, and

adjusting the resolution:

```
movie=part1.mp4, scale=512:288 [v1] ; amovie=part1.mp4 [a1] ;
movie=part2.mp4, scale=512:288 [v2] ; amovie=part2.mp4 [a2] ;
[v1] [v2] concat [outv] ; [a1] [a2] concat=v=0:a=1 [outa]
```

Note that a desync will happen at the stitch if the audio and video streams do not have exactly the same duration in the first file.

Commands

This filter supports the following commands:

next Close the current segment and step to the next one

ebur128

EBU R128 scanner filter. This filter takes an audio stream and analyzes its loudness level. By default, it logs a message at a frequency of 10Hz with the Momentary loudness (identified by "M"), Short-term loudness ("S"), Integrated loudness ("I") and Loudness Range ("LRA").

The filter can only analyze streams which have sample format is double-precision floating point. The input stream will be converted to this specification, if needed. Users may need to insert `aformat` and/or `aresample` filters after this filter to obtain the original parameters.

The filter also has a video output (see the *video* option) with a real time graph to observe the loudness evolution. The graphic contains the logged message mentioned above, so it is not printed anymore when this option is set, unless the verbose logging is set. The main graphing area contains the short-term loudness (3 seconds of analysis), and the gauge on the right is for the momentary loudness (400 milliseconds), but can optionally be configured to instead display short-term loudness (see *gauge*).

The green area marks a +/- 1LU target range around the target loudness (-23LUFS by default, unless modified through *target*).

More information about the Loudness Recommendation EBU R128 on <http://tech.ebu.ch/loudness>.

The filter accepts the following options:

video

Activate the video output. The audio stream is passed unchanged whether this option is set or no. The video stream will be the first output stream if activated. Default is 0.

size Set the video size. This option is for video only. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default and minimum resolution is "640x480".

meter

Set the EBU scale meter. Default is 9. Common values are 9 and 18, respectively for EBU scale meter +9 and EBU scale meter +18. Any other integer value between this range is allowed.

metadata

Set metadata injection. If set to 1, the audio input will be segmented into 100ms output frames, each of them containing various loudness information in metadata. All the metadata keys are prefixed with "lavfi.r128."

Default is 0.

framelog

Force the frame logging level.

Available values are:

quiet

logging disabled

info information logging level

verbose

verbose logging level

By default, the logging level is set to *info*. If the **video** or the **metadata** options are set, it switches to *verbose*.

peak

Set peak mode(s).

Available modes can be cumulated (the option is a "flag" type). Possible values are:

none

Disable any peak mode (default).

sample

Enable sample-peak mode.

Simple peak mode looking for the higher sample value. It logs a message for sample-peak (identified by "SPK").

true Enable true-peak mode.

If enabled, the peak lookup is done on an over-sampled version of the input stream for better peak accuracy. It logs a message for true-peak. (identified by "TPK") and true-peak per frame (identified by "FTPK"). This mode requires a build with "libswresample".

dualmono

Treat mono input files as "dual mono". If a mono file is intended for playback on a stereo system, its EBU R128 measurement will be perceptually incorrect. If set to "true", this option will compensate for this effect. Multi-channel input files are not affected by this option.

panlaw

Set a specific pan law to be used for the measurement of dual mono files. This parameter is optional, and has a default value of -3.01dB.

target

Set a specific target level (in LUFS) used as relative zero in the visualization. This parameter is optional and has a default value of -23LUFS as specified by EBU R128. However, material published online may prefer a level of -16LUFS (e.g. for use with podcasts or video platforms).

gauge

Set the value displayed by the gauge. Valid values are "momentary" and "shortterm". By default the momentary value will be used, but in certain scenarios it may be more useful to observe the short term value instead (e.g. live mixing).

scale

Sets the display scale for the loudness. Valid parameters are "absolute" (in LUFS) or "relative" (LU) relative to the target. This only affects the video output, not the summary or continuous log output.

Examples

- ⊕ Real-time graph using **ffplay**, with a EBU scale meter +18:

```
ffplay -f lavfi -i "amovie=input.mp3,ebur128=video=1:meter=18 [out0][out1]"
```

- ⊕ Run an analysis with **ffmpeg**:

```
ffmpeg -nostats -i input.mp3 -filter_complex ebur128 -f null -
```

interleave, ainterleave

Temporally interleave frames from several inputs.

"interleave" works with video inputs, "ainterleave" with audio.

These filters read frames from several inputs and send the oldest queued frame to the output.

Input streams must have well defined, monotonically increasing frame timestamp values.

In order to submit one frame to output, these filters need to enqueue at least one frame for each input, so they cannot work in case one input is not yet terminated and will not receive incoming frames.

For example consider the case when one input is a "select" filter which always drops input frames. The "interleave" filter will keep reading from that input, but it will never be able to send new frames to output until the input sends an end-of-stream signal.

Also, depending on inputs synchronization, the filters will drop frames in case one input receives more frames than the other ones, and the queue is already filled.

These filters accept the following options:

nb_inputs, n

Set the number of different inputs, it is 2 by default.

duration

How to determine the end-of-stream.

longest

The duration of the longest input. (default)

shortest

The duration of the shortest input.

first The duration of the first input.

Examples

- ⊕ Interleave frames belonging to different streams using **ffmpeg**:

```
ffmpeg -i bambi.avi -i pr0n.mkv -filter_complex "[0:v][1:v] interleave" out.avi
```

- ⊕ Add flickering blur effect:

```
select='if(gt(random(0), 0.2), 1, 2)':n=2 [tmp], boxblur=2:2, [tmp] interleave
```

latency, alateny

Measure filtering latency.

Report previous filter filtering latency, delay in number of audio samples for audio filters or number of video frames for video filters.

On end of input stream, filter will report min and max measured latency for previous running filter in filtergraph.

metadata, ametadata

Manipulate frame metadata.

This filter accepts the following options:

mode

Set mode of operation of the filter.

Can be one of the following:

select

If both "value" and "key" is set, select frames which have such metadata. If only "key" is set, select every frame that has such key in metadata.

add Add new metadata "key" and "value". If key is already available do nothing.

modify

Modify value of already present key.

delete

If "value" is set, delete only keys that have such value. Otherwise, delete key. If "key" is not set, delete all metadata values in the frame.

print

Print key and its value if metadata was found. If "key" is not set print all metadata values

available in frame.

key Set key used with all modes. Must be set for all modes except "print" and "delete".

value

Set metadata value which will be used. This option is mandatory for "modify" and "add" mode.

function

Which function to use when comparing metadata value and "value".

Can be one of following:

same_str

Values are interpreted as strings, returns true if metadata value is same as "value".

starts_with

Values are interpreted as strings, returns true if metadata value starts with the "value" option string.

less Values are interpreted as floats, returns true if metadata value is less than "value".

equal

Values are interpreted as floats, returns true if "value" is equal with metadata value.

greater

Values are interpreted as floats, returns true if metadata value is greater than "value".

expr

Values are interpreted as floats, returns true if expression from option "expr" evaluates to true.

ends_with

Values are interpreted as strings, returns true if metadata value ends with the "value" option string.

expr

Set expression which is used when "function" is set to "expr". The expression is evaluated through the eval API and can contain the following constants:

VALUE1, FRAMEVAL

Float representation of "value" from metadata key.

VALUE2, USERVAL

Float representation of "value" as supplied by user in "value" option.

file If specified in "print" mode, output is written to the named file. Instead of plain filename any writable url can be specified. Filename “-” is a shorthand for standard output. If "file" option is not set, output is written to the log with AV_LOG_INFO loglevel.

direct

Reduces buffering in print mode when output is written to a URL set using *file*.

Examples

- ⊕ Print all metadata values for frames with key "lavfi.signalstats.YDIF" with values between 0 and 1.

```
signalstats,metadata=print:key=lavfi.signalstats.YDIF:value=0:function=expr:expr='between(VALUE1,0,1)'
```

- ⊕ Print silencedetect output to file *metadata.txt*.

```
silencedetect,ametadata=mode=print:file=metadata.txt
```

- ⊕ Direct all metadata to a pipe with file descriptor 4.

```
metadata=mode=print:file='pipe':4'
```

perms, aperms

Set read/write permissions for the output frames.

These filters are mainly aimed at developers to test direct path in the following filter in the filtergraph.

The filters accept the following options:

mode

Select the permissions mode.

It accepts the following values:

none

Do nothing. This is the default.

ro Set all the output frames read-only.

rw Set all the output frames directly writable.

toggle

Make the frame read-only if writable, and writable if read-only.

random

Set each output frame read-only or writable randomly.

seed

Set the seed for the *random* mode, must be an integer included between 0 and "UINT32_MAX". If not specified, or if explicitly set to "-1", the filter will try to use a good random seed on a best effort basis.

Note: in case of auto-inserted filter between the permission filter and the following one, the permission might not be received as expected in that following filter. Inserting a **format** or **aformat** filter before the perms/aperms filter can avoid this problem.

realtime, arealtime

Slow down filtering to match real time approximately.

These filters will pause the filtering for a variable amount of time to match the output rate with the input timestamps. They are similar to the **re** option to "ffmpeg".

They accept the following options:

limit

Time limit for the pauses. Any pause longer than that will be considered a timestamp discontinuity and reset the timer. Default is 2 seconds.

speed

Speed factor for processing. The value must be a float larger than zero. Values larger than 1.0 will result in faster than realtime processing, smaller will slow processing down. The *limit* is automatically adapted accordingly. Default is 1.0.

A processing speed faster than what is possible without these filters cannot be achieved.

Commands

Both filters supports the all above options as **commands**.

segment, asegment

Split single input stream into multiple streams.

This filter does opposite of concat filters.

"segment" works on video frames, "asegment" on audio samples.

This filter accepts the following options:

timestamps

Timestamps of output segments separated by '|'. The first segment will run from the beginning of the input stream. The last segment will run until the end of the input stream

frames, samples

Exact frame/sample count to split the segments.

In all cases, prefixing an each segment with '+' will make it relative to the previous segment.

Examples

- ⊕ Split input audio stream into three output audio streams, starting at start of input audio stream and storing that in 1st output audio stream, then following at 60th second and storing than in 2nd output audio stream, and last after 150th second of input audio stream store in 3rd output audio stream:

```
asegment=timestamps="60|150"
```

select, aselect

Select frames to pass in output.

This filter accepts the following options:

expr, e

Set expression, which is evaluated for each input frame.

If the expression is evaluated to zero, the frame is discarded.

If the evaluation result is negative or NaN, the frame is sent to the first output; otherwise it is sent to the output with index " $\text{ceil}(\text{val})-1$ ", assuming that the input index starts from 0.

For example a value of 1.2 corresponds to the output with index " $\text{ceil}(1.2)-1 = 2-1 = 1$ ", that is the second output.

outputs, n

Set the number of outputs. The output to which to send the selected frame is based on the result of the evaluation. Default value is 1.

The expression can contain the following constants:

n The (sequential) number of the filtered frame, starting from 0.

selected_n

The (sequential) number of the selected frame, starting from 0.

prev_selected_n

The sequential number of the last selected frame. It's NAN if undefined.

TB The timebase of the input timestamps.

pts The PTS (Presentation TimeStamp) of the filtered frame, expressed in *TB* units. It's NAN if undefined.

t The PTS of the filtered frame, expressed in seconds. It's NAN if undefined.

prev_pts

The PTS of the previously filtered frame. It's NAN if undefined.

prev_selected_pts

The PTS of the last previously filtered frame. It's NAN if undefined.

prev_selected_t

The PTS of the last previously selected frame, expressed in seconds. It's NAN if undefined.

start_pts

The first PTS in the stream which is not NAN. It remains NAN if not found.

start_t

The first PTS, in seconds, in the stream which is not NAN. It remains NAN if not found.

pict_type (*video only*)

The type of the filtered frame. It can assume one of the following values:

I

P

B

S

SI

SP

BI

interlace_type (*video only*)

The frame interlace type. It can assume one of the following values:

PROGRESSIVE

The frame is progressive (not interlaced).

TOPFIRST

The frame is top-field-first.

BOTTOMFIRST

The frame is bottom-field-first.

consumed_sample_n (*audio only*)

the number of selected samples before the current frame

samples_n (*audio only*)

the number of samples in the current frame

sample_rate (*audio only*)

the input sample rate

key This is 1 if the filtered frame is a key-frame, 0 otherwise.

pos the position in the file of the filtered frame, -1 if the information is not available (e.g. for synthetic video)

scene (*video only*)

value between 0 and 1 to indicate a new scene; a low value reflects a low probability for the

current frame to introduce a new scene, while a higher value means the current frame is more likely to be one (see the example below)

concatdec_select

The concat demuxer can select only part of a concat input file by setting an inpoint and an outpoint, but the output packets may not be entirely contained in the selected interval. By using this variable, it is possible to skip frames generated by the concat demuxer which are not exactly contained in the selected interval.

This works by comparing the frame pts against the *lavf.concat.start_time* and the *lavf.concat.duration* packet metadata values which are also present in the decoded frames.

The *concatdec_select* variable is -1 if the frame pts is at least start_time and either the duration metadata is missing or the frame pts is less than start_time + duration, 0 otherwise, and NaN if the start_time metadata is missing.

That basically means that an input frame is selected if its pts is within the interval set by the concat demuxer.

The default value of the select expression is "1".

Examples

- ⊕ Select all frames in input:

```
select
```

The example above is the same as:

```
select=1
```

- ⊕ Skip all frames:

```
select=0
```

- ⊕ Select only I-frames:

```
select='eq(pict_type,I)'
```

- ⊕ Select one frame every 100:

```
select='not(mod(n,100))'
```

- ⊕ Select only frames contained in the 10-20 time interval:

```
select=between(t,10,20)
```

- ⊕ Select only I-frames contained in the 10-20 time interval:

```
select=between(t,10,20)*eq(pict_type,I)
```

- ⊕ Select frames with a minimum distance of 10 seconds:

```
select='isnan(prev_selected_t)+gte(t-prev_selected_t,10)'
```

- ⊕ Use aselect to select only audio frames with samples number > 100:

```
aselect='gt(samples_n,100)'
```

- ⊕ Create a mosaic of the first scenes:

```
ffmpeg -i video.avi -vf select='gt(scene,0.4)',scale=160:120,tile -frames:v 1 preview.png
```

Comparing *scene* against a value between 0.3 and 0.5 is generally a sane choice.

- ⊕ Send even and odd frames to separate outputs, and compose them:

```
select=n=2:e='mod(n, 2)+1' [odd][even]; [odd] pad=h=2*ih [tmp]; [tmp][even] overlay=y=h
```

- ⊕ Select useful frames from an ffmpeg file which is using inpoints and outpoints but where the source files are not intra frame only.

```
ffmpeg -copyts -vsync 0 -segment_time_metadata 1 -i input.ffmpeg -vf select=concatdec_select -af aselect=
```

sendcmd, asendcmd

Send commands to filters in the filtergraph.

These filters read commands to be sent to other filters in the filtergraph.

"sendcmd" must be inserted between two video filters, "asendcmd" must be inserted between two audio filters, but apart from that they act the same way.

The specification of commands can be provided in the filter arguments with the *commands* option, or in a file specified by the *filename* option.

These filters accept the following options:

commands, c

Set the commands to be read and sent to the other filters.

filename, f

Set the filename of the commands to be read and sent to the other filters.

Commands syntax

A commands description consists of a sequence of interval specifications, comprising a list of commands to be executed when a particular event related to that interval occurs. The occurring event is typically the current frame time entering or leaving a given time interval.

An interval is specified by the following syntax:

<START>[-<END>] <COMMANDS>;

The time interval is specified by the *START* and *END* times. *END* is optional and defaults to the maximum time.

The current frame time is considered within the specified interval if it is included in the interval [*START*, *END*), that is when the time is greater or equal to *START* and is lesser than *END*.

COMMANDS consists of a sequence of one or more command specifications, separated by ",", relating to that interval. The syntax of a command specification is given by:

[<FLAGS>] <TARGET> <COMMAND> <ARG>

FLAGS is optional and specifies the type of events relating to the time interval which enable sending the specified command, and must be a non-null sequence of identifier flags separated by "+" or "|" and enclosed between "[" and "]".

The following flags are recognized:

enter

The command is sent when the current frame timestamp enters the specified interval. In other

words, the command is sent when the previous frame timestamp was not in the given interval, and the current is.

leave

The command is sent when the current frame timestamp leaves the specified interval. In other words, the command is sent when the previous frame timestamp was in the given interval, and the current is not.

expr

The command *ARG* is interpreted as expression and result of expression is passed as *ARG*.

The expression is evaluated through the eval API and can contain the following constants:

POS

Original position in the file of the frame, or undefined if undefined for the current frame.

PTS The presentation timestamp in input.

N The count of the input frame for video or audio, starting from 0.

T The time in seconds of the current frame.

TS The start time in seconds of the current command interval.

TE The end time in seconds of the current command interval.

TI The interpolated time of the current command interval, $TI = (T - TS) / (TE - TS)$.

W The video frame width.

H The video frame height.

If *FLAGS* is not specified, a default value of "[enter]" is assumed.

TARGET specifies the target of the command, usually the name of the filter class or a specific filter instance name.

COMMAND specifies the name of the command for the target filter.

ARG is optional and specifies the optional list of argument for the given *COMMAND*.

Between one interval specification and another, whitespaces, or sequences of characters starting with "#" until the end of line, are ignored and can be used to annotate comments.

A simplified BNF description of the commands specification syntax follows:

```

<COMMAND_FLAG> ::= "enter" | "leave"
<COMMAND_FLAGS> ::= <COMMAND_FLAG> [(+|")<COMMAND_FLAG>]
<COMMAND>      ::= ["[" <COMMAND_FLAGS> "]" ] <TARGET> <COMMAND> [<ARG>]
<COMMANDS>     ::= <COMMAND> [,<COMMANDS>]
<INTERVAL>     ::= <START>[-<END>] <COMMANDS>
<INTERVALS>    ::= <INTERVAL>[;<INTERVALS>]

```

Examples

- ⊕ Specify audio tempo change at second 4:

```
asendcmd=c='4.0 atempo tempo 1.5',atempo
```

- ⊕ Target a specific filter instance:

```
asendcmd=c='4.0 atempo@my tempo 1.5',atempo@my
```

- ⊕ Specify a list of drawtext and hue commands in a file.

```
# show text in the interval 5-10
5.0-10.0 [enter] drawtext reinit 'fontfile=FreeSerif.ttf:text=hello world',
[leave] drawtext reinit 'fontfile=FreeSerif.ttf:text=';
```

```
# desaturate the image in the interval 15-20
15.0-20.0 [enter] hue s 0,
[enter] drawtext reinit 'fontfile=FreeSerif.ttf:text=nocolor',
[leave] hue s 1,
[leave] drawtext reinit 'fontfile=FreeSerif.ttf:text=color';
```

```
# apply an exponential saturation fade-out effect, starting from time 25
25 [enter] hue s exp(25-t)
```

A filtergraph allowing to read and process the above command list stored in a file *test.cmd*, can be specified with:

```
sendcmd=f=test.cmd,drawtext=fontfile=FreeSerif.ttf:text='',hue
```

setpts, asetpts

Change the PTS (presentation timestamp) of the input frames.

"setpts" works on video frames, "asetpts" on audio frames.

This filter accepts the following options:

expr

The expression which is evaluated for each frame to construct its timestamp.

The expression is evaluated through the eval API and can contain the following constants:

FRAME_RATE, FR

frame rate, only defined for constant frame-rate video

PTS The presentation timestamp in input

N The count of the input frame for video or the number of consumed samples, not including the current frame for audio, starting from 0.

NB_CONSUMED_SAMPLES

The number of consumed samples, not including the current frame (only audio)

NB_SAMPLES, S

The number of samples in the current frame (only audio)

SAMPLE_RATE, SR

The audio sample rate.

STARTPTS

The PTS of the first frame.

STARTT

the time in seconds of the first frame

INTERLACED

State whether the current frame is interlaced.

T the time in seconds of the current frame

POS

original position in the file of the frame, or undefined if undefined for the current frame

PREV_INPTS

The previous input PTS.

PREV_INT

previous input time in seconds

PREV_OUTPTS

The previous output PTS.

PREV_OUTT

previous output time in seconds

RTCTIME

The wallclock (RTC) time in microseconds. This is deprecated, use **time(0)** instead.

RTCSTART

The wallclock (RTC) time at the start of the movie in microseconds.

TB The timebase of the input timestamps.

Examples

⊕ Start counting PTS from zero

setpts=PTS-STARTPTS

⊕ Apply fast motion effect:

setpts=0.5*PTS

⊕ Apply slow motion effect:

setpts=2.0*PTS

⊕ Set fixed rate of 25 frames per second:

```
setpts=N/(25*TB)
```

- ⊕ Set fixed rate 25 fps with some jitter:

```
setpts='1/(25*TB) * (N + 0.05 * sin(N*2*PI/25))'
```

- ⊕ Apply an offset of 10 seconds to the input PTS:

```
setpts=PTS+10/TB
```

- ⊕ Generate timestamps from a "live source" and rebase onto the current timebase:

```
setpts='(RTCTIME - RTCSTART) / (TB * 1000000)'
```

- ⊕ Generate timestamps by counting samples:

```
asetpts=N/SR/TB
```

setrange

Force color range for the output video frame.

The "setrange" filter marks the color range property for the output frames. It does not change the input frame, but only sets the corresponding property, which affects how the frame is treated by following filters.

The filter accepts the following options:

range

Available values are:

auto Keep the same color range property.

unspecified, unknown

Set the color range as unspecified.

limited, tv, mpeg

Set the color range as limited.

full, pc, jpeg

Set the color range as full.

settb, settb

Set the timebase to use for the output frames timestamps. It is mainly useful for testing timebase configuration.

It accepts the following parameters:

expr, tb

The expression which is evaluated into the output timebase.

The value for **tb** is an arithmetic expression representing a rational. The expression can contain the constants "AVTB" (the default timebase), "intb" (the input timebase) and "sr" (the sample rate, audio only). Default value is "intb".

Examples

- ⊕ Set the timebase to 1/25:

```
settb=expr=1/25
```

- ⊕ Set the timebase to 1/10:

```
settb=expr=0.1
```

- ⊕ Set the timebase to 1001/1000:

```
settb=1+0.001
```

- ⊕ Set the timebase to 2*intb:

```
settb=2*intb
```

- ⊕ Set the default timebase value:

```
settb=AVTB
```

showcqt

Convert input audio to a video output representing frequency spectrum logarithmically using Brown-Puckette constant Q transform algorithm with direct frequency domain coefficient calculation (but the transform itself is not really constant Q, instead the Q factor is actually variable/clamped), with musical tone scale, from E0 to D#10.

The filter accepts the following options:

size, s

Specify the video size for the output. It must be even. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "1920x1080".

fps, rate, r

Set the output frame rate. Default value is 25.

bar_h

Set the bargraph height. It must be even. Default value is "-1" which computes the bargraph height automatically.

axis_h

Set the axis height. It must be even. Default value is "-1" which computes the axis height automatically.

sono_h

Set the sonogram height. It must be even. Default value is "-1" which computes the sonogram height automatically.

fullhd

Set the fullhd resolution. This option is deprecated, use *size, s* instead. Default value is 1.

sono_v, volume

Specify the sonogram volume expression. It can contain variables:

bar_v

the *bar_v* evaluated expression

frequency, freq, f

the frequency where it is evaluated

timeclamp, tc

the value of *timeclamp* option

and functions:

a_weighting(f)

A-weighting of equal loudness

b_weighting(f)

B-weighting of equal loudness

c_weighting(f)

C-weighting of equal loudness.

Default value is 16.

bar_v, volume2

Specify the bargraph volume expression. It can contain variables:

sono_v

the *sono_v* evaluated expression

frequency, freq, f

the frequency where it is evaluated

timeclamp, tc

the value of *timeclamp* option

and functions:

a_weighting(f)

A-weighting of equal loudness

b_weighting(f)

B-weighting of equal loudness

c_weighting(f)

C-weighting of equal loudness.

Default value is "sono_v".

sono_g, gamma

Specify the sonogram gamma. Lower gamma makes the spectrum more contrast, higher gamma makes the spectrum having more range. Default value is 3. Acceptable range is "[1, 7]".

bar_g, gamma2

Specify the bargraph gamma. Default value is 1. Acceptable range is "[1, 7]".

bar_t

Specify the bargraph transparency level. Lower value makes the bargraph sharper. Default value is 1. Acceptable range is "[0, 1]".

timeclamp, tc

Specify the transform timeclamp. At low frequency, there is trade-off between accuracy in time domain and frequency domain. If timeclamp is lower, event in time domain is represented more accurately (such as fast bass drum), otherwise event in frequency domain is represented more accurately (such as bass guitar). Acceptable range is "[0.002, 1]". Default value is 0.17.

attack

Set attack time in seconds. The default is 0 (disabled). Otherwise, it limits future samples by applying asymmetric windowing in time domain, useful when low latency is required. Accepted range is "[0, 1]".

basefreq

Specify the transform base frequency. Default value is 20.01523126408007475, which is frequency 50 cents below E0. Acceptable range is "[10, 100000]".

endfreq

Specify the transform end frequency. Default value is 20495.59681441799654, which is frequency 50 cents above D#10. Acceptable range is "[10, 100000]".

coeffclamp

This option is deprecated and ignored.

tlength

Specify the transform length in time domain. Use this option to control accuracy trade-off between time domain and frequency domain at every frequency sample. It can contain variables:

frequency, freq, f

the frequency where it is evaluated

timeclamp, tc

the value of *timeclamp* option.

Default value is " $384*tc/(384+tc*f)$ ".

count

Specify the transform count for every video frame. Default value is 6. Acceptable range is "[1,

30]".

fcount

Specify the transform count for every single pixel. Default value is 0, which makes it computed automatically. Acceptable range is "[0, 10]".

fontfile

Specify font file for use with freetype to draw the axis. If not specified, use embedded font. Note that drawing with font file or embedded font is not implemented with custom *basefreq* and *endfreq*, use *axisfile* option instead.

font Specify fontconfig pattern. This has lower priority than *fontfile*. The ":" in the pattern may be replaced by "|" to avoid unnecessary escaping.

fontcolor

Specify font color expression. This is arithmetic expression that should return integer value 0xRRGGBB. It can contain variables:

frequency, freq, f

the frequency where it is evaluated

timeclamp, tc

the value of *timeclamp* option

and functions:

midi(f)

midi number of frequency f, some midi numbers: E0(16), C1(24), C2(36), A4(69)

r(x), g(x), b(x)

red, green, and blue value of intensity x.

Default value is "st(0, (midi(f)-59.5)/12); st(1, if(between(ld(0),0,1), 0.5-0.5*cos(2*PI*ld(0)), 0)); r(1-ld(1)) + b(ld(1))".

axisfile

Specify image file to draw the axis. This option override *fontfile* and *fontcolor* option.

axis, text

Enable/disable drawing text to the axis. If it is set to 0, drawing to the axis is disabled, ignoring

fontfile and *axisfile* option. Default value is 1.

csp Set colorspace. The accepted values are:

unspecified

Unspecified (default)

bt709

BT.709

fcc FCC

bt470bg

BT.470BG or BT.601-6 625

smpte170m

SMPTE-170M or BT.601-6 525

smpte240m

SMPTE-240M

bt2020ncl

BT.2020 with non-constant luminance

cscheme

Set spectrogram color scheme. This is list of floating point values with format "left_r|left_g|left_b|right_r|right_g|right_b". The default is "1|0.5|0|0|0.5|1".

Examples

- ⊕ Playing audio while showing the spectrum:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt [out0]'
```

- ⊕ Same as above, but with frame rate 30 fps:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt=fps=30:count=5 [out0]'
```

- ⊕ Playing at 1280x720:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt=s=1280x720:count=4 [out0]'
```

- ⊕ Disable sonogram display:

```
sono_h=0
```

- ⊕ A1 and its harmonics: A1, A2, (near)E3, A3:

```
ffplay -f lavfi 'aevalsrc=0.1*sin(2*PI*55*t)+0.1*sin(4*PI*55*t)+0.1*sin(6*PI*55*t)+0.1*sin(8*PI*55*t),
asplit[a][out1]; [a] showcqt [out0]'
```

- ⊕ Same as above, but with more accuracy in frequency domain:

```
ffplay -f lavfi 'aevalsrc=0.1*sin(2*PI*55*t)+0.1*sin(4*PI*55*t)+0.1*sin(6*PI*55*t)+0.1*sin(8*PI*55*t),
asplit[a][out1]; [a] showcqt=timeclamp=0.5 [out0]'
```

- ⊕ Custom volume:

```
bar_v=10:sono_v=bar_v*a_weighting(f)
```

- ⊕ Custom gamma, now spectrum is linear to the amplitude.

```
bar_g=2:sono_g=2
```

- ⊕ Custom tlength equation:

```
tc=0.33:tlength='st(0,0.17); 384*tc / (384 / ld(0) + tc*f / (1-ld(0))) + 384*tc / (tc*f / ld(0) + 384 / (1-ld(0)))'
```

- ⊕ Custom fontcolor and fontfile, C-note is colored green, others are colored blue:

```
fontcolor='if(mod(floor(midi(f)+0.5),12), 0x0000FF, g(1))':fontfile=myfont.ttf
```

- ⊕ Custom font using fontconfig:

```
font='Courier New,Monospace,mono|bold'
```

- ⊕ Custom frequency range with custom axis using image file:

```
axisfile=myaxis.png:basefreq=40:endfreq=10000
```

showcwt

Convert input audio to video output representing frequency spectrum using Continuous Wavelet Transform and Morlet wavelet.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "640x512".

rate, r

Set the output frame rate. Default value is 25.

scale

Set the frequency scale used. Allowed values are:

linear

log2

bark

mel

erbs

Default value is "linear".

min Set the minimum frequency that will be used in output. Default is 20 Hz.

max Set the maximum frequency that will be used in output. Default is 20000 Hz. The real frequency upper limit depends on input audio's sample rate and such will be enforced on this value when it is set to value greater than Nyquist frequency.

logb

Set the logarithmic basis for brightness strength when mapping calculated magnitude values to pixel values. Allowed range is from 0 to 1. Default value is 0.0001.

deviation

Set the frequency deviation. Lower values than 1 are more frequency oriented, while higher values than 1 are more time oriented. Allowed range is from 0 to 10. Default value is 1.

pps Set the number of pixel output per each second in one row. Allowed range is from 1 to 1024. Default value is 64.

mode

Set the output visual mode. Allowed values are:

magnitude

Show magnitude.

phase

Show only phase.

magphase

Show combination of magnitude and phase. Magnitude is mapped to brightness and phase to color.

channel

Show unique color per channel magnitude.

stereo

Show unique color per stereo difference.

Default value is "magnitude".

slide

Set the output slide method. Allowed values are:

replace**scroll****frame****direction**

Set the direction method for output slide method. Allowed values are:

lr Direction from left to right.

rl Direction from right to left.

ud Direction from up to down.

du Direction from down to up.

showfreqs

Convert input audio to video output representing the audio power spectrum. Audio amplitude is on

Y-axis while frequency is on X-axis.

The filter accepts the following options:

size, s

Specify size of video. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default is "1024x512".

rate, r

Set video rate. Default is 25.

mode

Set display mode. This set how each frequency bin will be represented.

It accepts the following values:

line

bar

dot

Default is "bar".

ascale

Set amplitude scale.

It accepts the following values:

lin Linear scale.

sqrt Square root scale.

cbrt Cubic root scale.

log Logarithmic scale.

Default is "log".

fscale

Set frequency scale.

It accepts the following values:

lin Linear scale.

log Logarithmic scale.

rlog Reverse logarithmic scale.

Default is "lin".

win_size

Set window size. Allowed range is from 16 to 65536.

Default is 2048

win_func

Set windowing function.

It accepts the following values:

rect

bartlett

hanning

hamming

blackman

welch

flattop

bharris

bnuttall

bhann

sine

nuttall

lanczos

gauss

tukey

dolph

cauchy

parzen

poisson

bohman

kaiser

Default is "hanning".

overlap

Set window overlap. In range "[0, 1]". Default is 1, which means optimal overlap for selected window function will be picked.

averaging

Set time averaging. Setting this to 0 will display current maximal peaks. Default is 1, which means time averaging is disabled.

colors

Specify list of colors separated by space or by '|' which will be used to draw channel frequencies. Unrecognized or missing colors will be replaced by white color.

cmode

Set channel display mode.

It accepts the following values:

combined
separate

Default is "combined".

minamp

Set minimum amplitude used in "log" amplitude scaler.

data Set data display mode.

It accepts the following values:

magnitude
phase
delay

Default is "magnitude".

channels

Set channels to use when processing audio. By default all are processed.

showspatial

Convert stereo input audio to a video output, representing the spatial relationship between two channels.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "512x512".

win_size

Set window size. Allowed range is from *1024* to *65536*. Default size is *4096*.

win_func

Set window function.

It accepts the following values:

- rect**
- bartlett**
- hann**
- hanning**
- hamming**
- blackman**
- welch**
- flattop**
- bharris**
- bnttall**
- bhann**
- sine**
- nutall**
- lanczos**
- gauss**
- tukey**
- dolph**
- cauchy**
- parzen**
- poisson**

bohman
kaiser

Default value is "hann".

rate, r

Set output framerate.

showspectrum

Convert input audio to a video output, representing the audio frequency spectrum.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "640x512".

slide

Specify how the spectrum should slide along the window.

It accepts the following values:

replace

the samples start again on the left when they reach the right

scroll

the samples scroll from right to left

fullframe

frames are only produced when the samples reach the right

rscroll

the samples scroll from left to right

lreplace

the samples start again on the right when they reach the left

Default value is "replace".

mode

Specify display mode.

It accepts the following values:

combined

all channels are displayed in the same row

separate

all channels are displayed in separate rows

Default value is **combined**.

color

Specify display color mode.

It accepts the following values:

channel

each channel is displayed in a separate color

intensity

each channel is displayed using the same color scheme

rainbow

each channel is displayed using the rainbow color scheme

moreland

each channel is displayed using the moreland color scheme

nebulae

each channel is displayed using the nebulae color scheme

fire each channel is displayed using the fire color scheme

fiery

each channel is displayed using the fiery color scheme

fruit each channel is displayed using the fruit color scheme

cool each channel is displayed using the cool color scheme

magma

each channel is displayed using the magma color scheme

green

each channel is displayed using the green color scheme

viridis

each channel is displayed using the viridis color scheme

plasma

each channel is displayed using the plasma color scheme

cividis

each channel is displayed using the cividis color scheme

terrain

each channel is displayed using the terrain color scheme

Default value is **channel**.

scale

Specify scale used for calculating intensity color values.

It accepts the following values:

lin linear

sqrt square root, default

cbrt cubic root

log logarithmic

4thrt

4th root

5thrt

5th root

Default value is **sqrt**.

fscale

Specify frequency scale.

It accepts the following values:

lin linear

log logarithmic

Default value is **lin**.

saturation

Set saturation modifier for displayed colors. Negative values provide alternative color scheme. 0 is no saturation at all. Saturation must be in [-10.0, 10.0] range. Default value is 1.

win_func

Set window function.

It accepts the following values:

rect

bartlett

hann

hanning

hamming

blackman

welch

flattop

bharris

bnuttall

bhann

sine

nutall

lanczos

gauss

tukey

dolph

cauchy

parzen

poisson

bohman**kaiser**

Default value is "hann".

orientation

Set orientation of time vs frequency axis. Can be "vertical" or "horizontal". Default is "vertical".

overlap

Set ratio of overlap window. Default value is 0. When value is 1 overlap is set to recommended size for specific window function currently used.

gain Set scale gain for calculating intensity color values. Default value is 1.

data Set which data to display. Can be "magnitude", default or "phase", or unwrapped phase: "uphase".

rotation

Set color rotation, must be in [-1.0, 1.0] range. Default value is 0.

start Set start frequency from which to display spectrogram. Default is 0.

stop Set stop frequency to which to display spectrogram. Default is 0.

fps Set upper frame rate limit. Default is "auto", unlimited.

legend

Draw time and frequency axes and legends. Default is disabled.

drange

Set dynamic range used to calculate intensity color values. Default is 120 dBFS. Allowed range is from 10 to 200.

limit

Set upper limit of input audio samples volume in dBFS. Default is 0 dBFS. Allowed range is from -100 to 100.

opacity

Set opacity strength when using pixel format output with alpha component.

The usage is very similar to the showwaves filter; see the examples in that section.

Examples

- ⊕ Large window with logarithmic color scaling:

```
showspectrum=s=1280x480:scale=log
```

- ⊕ Complete example for a colored and sliding spectrum per channel using **ffplay**:

```
ffplay -f lavfi 'amovie=input.mp3, asplit [a][out1];  
[a] showspectrum=mode=separate:color=intensity:slide=1:scale=cbrt [out0]'
```

showspectrumpic

Convert input audio to a single video frame, representing the audio frequency spectrum.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "4096x2048".

mode

Specify display mode.

It accepts the following values:

combined

all channels are displayed in the same row

separate

all channels are displayed in separate rows

Default value is **combined**.

color

Specify display color mode.

It accepts the following values:

channel

each channel is displayed in a separate color

intensity

each channel is displayed using the same color scheme

rainbow

each channel is displayed using the rainbow color scheme

moreland

each channel is displayed using the moreland color scheme

nebulae

each channel is displayed using the nebulae color scheme

fire each channel is displayed using the fire color scheme

fiery

each channel is displayed using the fiery color scheme

fruit each channel is displayed using the fruit color scheme

cool each channel is displayed using the cool color scheme

magma

each channel is displayed using the magma color scheme

green

each channel is displayed using the green color scheme

viridis

each channel is displayed using the viridis color scheme

plasma

each channel is displayed using the plasma color scheme

cividis

each channel is displayed using the cividis color scheme

terrain

each channel is displayed using the terrain color scheme

Default value is **intensity**.

scale

Specify scale used for calculating intensity color values.

It accepts the following values:

lin linear

sqrt square root, default

cbrt cubic root

log logarithmic

4thrt
4th root

5thrt
5th root

Default value is **log**.

fscale

Specify frequency scale.

It accepts the following values:

lin linear

log logarithmic

Default value is **lin**.

saturation

Set saturation modifier for displayed colors. Negative values provide alternative color scheme. 0 is no saturation at all. Saturation must be in [-10.0, 10.0] range. Default value is 1.

win_func

Set window function.

It accepts the following values:

rect
bartlett
hann
hanning
hamming
blackman
welch
flattop
bharris
bnuttall
bhann
sine
nutall
lanczos
gauss
tukey
dolph
cauchy
parzen
poisson
bohman
kaiser

Default value is "hann".

orientation

Set orientation of time vs frequency axis. Can be "vertical" or "horizontal". Default is "vertical".

gain Set scale gain for calculating intensity color values. Default value is 1.

legend

Draw time and frequency axes and legends. Default is enabled.

rotation

Set color rotation, must be in [-1.0, 1.0] range. Default value is 0.

start Set start frequency from which to display spectrogram. Default is 0.

stop Set stop frequency to which to display spectrogram. Default is 0.

drange

Set dynamic range used to calculate intensity color values. Default is 120 dBFS. Allowed range is from 10 to 200.

limit

Set upper limit of input audio samples volume in dBFS. Default is 0 dBFS. Allowed range is from -100 to 100.

opacity

Set opacity strength when using pixel format output with alpha component.

Examples

- ⊕ Extract an audio spectrogram of a whole audio track in a 1024x1024 picture using **ffmpeg**:

```
ffmpeg -i audio.flac -lavfi showspectrumpic=s=1024x1024 spectrogram.png
```

showvolume

Convert input audio volume to a video output.

The filter accepts the following options:

rate, r

Set video rate.

- b** Set border width, allowed range is [0, 5]. Default is 1.

- w** Set channel width, allowed range is [80, 8192]. Default is 400.

- h** Set channel height, allowed range is [1, 900]. Default is 20.

- f** Set fade, allowed range is [0, 1]. Default is 0.95.

- c** Set volume color expression.

The expression can use the following variables:

VOLUME

Current max volume of channel in dB.

PEAK

Current peak.

CHANNEL

Current channel number, starting from 0.

- t** If set, displays channel names. Default is enabled.
- v** If set, displays volume values. Default is enabled.
- o** Set orientation, can be horizontal: "h" or vertical: "v", default is "h".
- s** Set step size, allowed range is [0, 5]. Default is 0, which means step is disabled.
- p** Set background opacity, allowed range is [0, 1]. Default is 0.
- m** Set metering mode, can be peak: "p" or rms: "r", default is "p".
- ds** Set display scale, can be linear: "lin" or log: "log", default is "lin".
- dm** In second. If set to > 0., display a line for the max level in the previous seconds. default is disabled: 0.
- dmc** The color of the max line. Use when "dm" option is set to > 0. default is: "orange"

showwaves

Convert input audio to a video output, representing the samples waves.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "600x240".

mode

Set display mode.

Available values are:

point

Draw a point for each sample.

line Draw a vertical line for each sample.

p2p Draw a point for each sample and a line between them.

cline

Draw a centered vertical line for each sample.

Default value is "point".

n Set the number of samples which are printed on the same column. A larger value will decrease the frame rate. Must be a positive integer. This option can be set only if the value for *rate* is not explicitly specified.

rate, r

Set the (approximate) output frame rate. This is done by setting the option *n*. Default value is "25".

split_channels

Set if channels should be drawn separately or overlap. Default value is 0.

colors

Set colors separated by '|' which are going to be used for drawing of each channel.

scale

Set amplitude scale.

Available values are:

lin Linear.

log Logarithmic.

sqrt Square root.

cbt Cubic root.

Default is linear.

draw

Set the draw mode. This is mostly useful to set for high n .

Available values are:

scale

Scale pixel values for each drawn sample.

full Draw every sample directly.

Default value is "scale".

Examples

- ⊕ Output the input file audio and the corresponding video representation at the same time:

```
amovie=a.mp3,asplit[out0],showwaves[out1]
```

- ⊕ Create a synthetic signal and show it with showwaves, forcing a frame rate of 30 frames per second:

```
aevalsrc=sin(1*2*PI*t)*sin(880*2*PI*t):cos(2*PI*200*t),asplit[out0],showwaves=r=30[out1]
```

showwavespic

Convert input audio to a single video frame, representing the samples waves.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the "**Video size**" section in the **ffmpeg-utils manual**. Default value is "600x240".

split_channels

Set if channels should be drawn separately or overlap. Default value is 0.

colors

Set colors separated by '|' which are going to be used for drawing of each channel.

scale

Set amplitude scale.

Available values are:

lin Linear.

log Logarithmic.

sqrt Square root.

cbrt Cubic root.

Default is linear.

draw

Set the draw mode.

Available values are:

scale

Scale pixel values for each drawn sample.

full Draw every sample directly.

Default value is "scale".

filter

Set the filter mode.

Available values are:

average

Use average samples values for each drawn sample.

peak

Use peak samples values for each drawn sample.

Default value is "average".

Examples

- ⊕ Extract a channel split representation of the wave form of a whole audio track in a 1024x800

picture using **ffmpeg**:

```
ffmpeg -i audio.flac -lavfi showwavespic=split_channels=1:s=1024x800 waveform.png
```

sidedata, asidedata

Delete frame side data, or select frames based on it.

This filter accepts the following options:

mode

Set mode of operation of the filter.

Can be one of the following:

select

Select every frame with side data of "type".

delete

Delete side data of "type". If "type" is not set, delete all side data in the frame.

type Set side data type used with all modes. Must be set for "select" mode. For the list of frame side data types, refer to the "AVFrameSideDataType" enum in *libavutil/frame.h*. For example, to choose "AV_FRAME_DATA_PANSCAN" side data, you must specify "PANSCAN".

spectrumsynth

Synthesize audio from 2 input video spectrums, first input stream represents magnitude across time and second represents phase across time. The filter will transform from frequency domain as displayed in videos back to time domain as presented in audio output.

This filter is primarily created for reversing processed **showspectrum** filter outputs, but can synthesize sound from other spectrograms too. But in such case results are going to be poor if the phase data is not available, because in such cases phase data need to be recreated, usually it's just recreated from random noise. For best results use gray only output ("channel" color mode in **showspectrum** filter) and "log" scale for magnitude video and "lin" scale for phase video. To produce phase, for 2nd video, use "data" option. Inputs videos should generally use "fullframe" slide mode as that saves resources needed for decoding video.

The filter accepts the following options:

sample_rate

Specify sample rate of output audio, the sample rate of audio from which spectrum was generated may differ.

channels

Set number of channels represented in input video spectrums.

scale

Set scale which was used when generating magnitude input spectrum. Can be "lin" or "log". Default is "log".

slide

Set slide which was used when generating inputs spectrums. Can be "replace", "scroll", "fullframe" or "rscroll". Default is "fullframe".

win_func

Set window function used for resynthesis.

overlap

Set window overlap. In range "[0, 1]". Default is 1, which means optimal overlap for selected window function will be picked.

orientation

Set orientation of input videos. Can be "vertical" or "horizontal". Default is "vertical".

Examples

- ⊕ First create magnitude and phase videos from audio, assuming audio is stereo with 44100 sample rate, then resynthesize videos back to audio with spectrumsynth:

```
ffmpeg -i input.flac -lavfi showspectrum=mode=separate:scale=log:overlap=0.875:color=channel:slide=fullfr
```

```
ffmpeg -i input.flac -lavfi showspectrum=mode=separate:scale=lin:overlap=0.875:color=channel:slide=fullfr
```

```
ffmpeg -i magnitude.nut -i phase.nut -lavfi spectrumsynth=channels=2:sample_rate=44100:win_func=hann:o
```

split, asplit

Split input into several identical outputs.

"asplit" works with audio input, "split" with video.

The filter accepts a single parameter which specifies the number of outputs. If unspecified, it defaults to 2.

Examples

- ⊕ Create two separate outputs from the same input:

```
[in] split [out0][out1]
```

- ⊕ To create 3 or more outputs, you need to specify the number of outputs, like in:

```
[in] asplit=3 [out0][out1][out2]
```

- ⊕ Create two separate outputs from the same input, one cropped and one padded:

```
[in] split [splitout1][splitout2];
[splitout1] crop=100:100:0:0 [cropout];
[splitout2] pad=200:200:100:100 [padout];
```

- ⊕ Create 5 copies of the input audio with **ffmpeg**:

```
ffmpeg -i INPUT -filter_complex asplit=5 OUTPUT
```

zmq, azmq

Receive commands sent through a libzmq client, and forward them to filters in the filtergraph.

"zmq" and "azmq" work as a pass-through filters. "zmq" must be inserted between two video filters, "azmq" between two audio filters. Both are capable to send messages to any filter type.

To enable these filters you need to install the libzmq library and headers and configure FFmpeg with "--enable-libzmq".

For more information about libzmq see: <<http://www.zeromq.org/>>

The "zmq" and "azmq" filters work as a libzmq server, which receives messages sent through a network interface defined by the **bind_address** (or the abbreviation "**b**") option. Default value of this option is *tcp://localhost:5555*. You may want to alter this value to your needs, but do not forget to escape any ':' signs (see **filtergraph escaping**).

The received message must be in the form:

```
<TARGET> <COMMAND> [<ARG>]
```

TARGET specifies the target of the command, usually the name of the filter class or a specific filter instance name. The default filter instance name uses the pattern **Parsed_<filter_name>_<index>**, but you can override this by using the **filter_name@id** syntax (see **Filtergraph syntax**).

COMMAND specifies the name of the command for the target filter.

ARG is optional and specifies the optional argument list for the given *COMMAND*.

Upon reception, the message is processed and the corresponding command is injected into the filtergraph. Depending on the result, the filter will send a reply to the client, adopting the format:

```
<ERROR_CODE> <ERROR_REASON>
<MESSAGE>
```

MESSAGE is optional.

Examples

Look at *tools/zmqsend* for an example of a zmq client which can be used to send commands processed by these filters.

Consider the following filtergraph generated by **ffplay**. In this example the last overlay filter has an instance name. All other filters will have default instance names.

```
ffplay -dumpgraph 1 -f lavfi "
color=s=100x100:c=red [l];
color=s=100x100:c=blue [r];
nullsrc=s=200x100, zmq [bg];
[bg][l] overlay [bg+l];
[bg+l][r] overlay@my=x=100 "
```

To change the color of the left side of the video, the following command can be used:

```
echo Parsed_color_0 c yellow | tools/zmqsend
```

To change the right side:

```
echo Parsed_color_1 c pink | tools/zmqsend
```

To change the position of the right side:

echo overlay@my x 150 | tools/zmqsend

MULTIMEDIA SOURCES

Below is a description of the currently available multimedia sources.

amovie

This is the same as **movie** source, except it selects an audio stream by default.

avsyncctest

Generate an Audio/Video Sync Test.

Generated stream periodically shows flash video frame and emits beep in audio. Useful to inspect A/V sync issues.

It accepts the following options:

size, s

Set output video size. Default value is "hd720".

framerate, fr

Set output video frame rate. Default value is 30.

samplerate, sr

Set output audio sample rate. Default value is 44100.

amplitude, a

Set output audio beep amplitude. Default value is 0.7.

period, p

Set output audio beep period in seconds. Default value is 3.

delay, dl

Set output video flash delay in number of frames. Default value is 0.

cycle, c

Enable cycling of video delays, by default is disabled.

duration, d

Set stream output duration. By default duration is unlimited.

fg, bg, ag

Set foreground/background/additional color.

movie

Read audio and/or video stream(s) from a movie container.

It accepts the following parameters:

filename

The name of the resource to read (not necessarily a file; it can also be a device or a stream accessed through some protocol).

format_name, f

Specifies the format assumed for the movie to read, and can be either the name of a container or an input device. If not specified, the format is guessed from *movie_name* or by probing.

seek_point, sp

Specifies the seek point in seconds. The frames will be output starting from this seek point. The parameter is evaluated with "av_strtod", so the numerical value may be suffixed by an IS postfix. The default value is "0".

streams, s

Specifies the streams to read. Several streams can be specified, separated by "+". The source will then have as many outputs, in the same order. The syntax is explained in the "**Stream specifiers**" section in the **ffmpeg manual**. Two special names, "dv" and "da" specify respectively the default (best suited) video and audio stream. Default is "dv", or "da" if the filter is called as "amovie".

stream_index, si

Specifies the index of the video stream to read. If the value is -1, the most suitable video stream will be automatically selected. The default value is "-1". Deprecated. If the filter is called "amovie", it will select audio instead of video.

loop

Specifies how many times to read the stream in sequence. If the value is 0, the stream will be looped infinitely. Default value is "1".

Note that when the movie is looped the source timestamps are not changed, so it will generate non monotonically increasing timestamps.

discontinuity

Specifies the time difference between frames above which the point is considered a timestamp discontinuity which is removed by adjusting the later timestamps.

dec_threads

Specifies the number of threads for decoding

format_opts

Specify format options for the opened file. Format options can be specified as a list of *key=value* pairs separated by ':'. The following example shows how to add `protocol_whitelist` and `protocol_blacklist` options:

```
ffplay -f lavfi
"movie=filename='1.sdp':format_opts='protocol_whitelist=file,rtp,udp\':protocol_blacklist=http"
```

It allows overlaying a second video on top of the main input of a filtergraph, as shown in this graph:

```
input -----> deltapts0 --> overlay --> output
                    ^
                    |
movie --> scale--> deltapts1 -----+
```

Examples

- ⊕ Skip 3.2 seconds from the start of the AVI file `in.avi`, and overlay it on top of the input labelled "in":

```
movie=in.avi:seek_point=3.2, scale=180:-1, setpts=PTS-STARTPTS [over];
[in] setpts=PTS-STARTPTS [main];
[main][over] overlay=16:16 [out]
```

- ⊕ Read from a `video4linux2` device, and overlay it on top of the input labelled "in":

```
movie=/dev/video0:f=video4linux2, scale=180:-1, setpts=PTS-STARTPTS [over];
[in] setpts=PTS-STARTPTS [main];
[main][over] overlay=16:16 [out]
```

- ⊕ Read the first video stream and the audio stream with id `0x81` from `dvd.vob`; the video is connected to the pad named "video" and the audio is connected to the pad named "audio":

```
movie=dvd.vob:s=v:0+#0x81 [video] [audio]
```

Commands

Both movie and amovie support the following commands:

seek

Perform seek using "av_seek_frame". The syntax is: seek *stream_index|timestamp|flags*

- ⊕ *stream_index*: If stream_index is -1, a default stream is selected, and *timestamp* is automatically converted from AV_TIME_BASE units to the stream specific time_base.
- ⊕ *timestamp*: Timestamp in AVStream.time_base units or, if no stream is specified, in AV_TIME_BASE units.
- ⊕ *flags*: Flags which select direction and seeking mode.

get_duration

Get movie duration in AV_TIME_BASE units.

EXTERNAL LIBRARIES

FFmpeg can be hooked up with a number of external libraries to add support for more formats. None of them are used by default, their use has to be explicitly requested by passing the appropriate flags to **./configure**.

Alliance for Open Media (AOM)

FFmpeg can make use of the AOM library for AV1 decoding and encoding.

Go to <<http://aomedia.org/>> and follow the instructions for installing the library. Then pass "--enable-libaom" to configure to enable it.

AMD AMF/VCE

FFmpeg can use the AMD Advanced Media Framework library for accelerated H.264 and HEVC(only windows) encoding on hardware with Video Coding Engine (VCE).

To enable support you must obtain the AMF framework header files(version 1.4.9+) from <<https://github.com/GPUOpen-LibrariesAndSDKs/AMF.git>>.

Create an "AMF/" directory in the system include path. Copy the contents of "AMF/amf/public/include/" into that directory. Then configure FFmpeg with "--enable-amf".

Initialization of amf encoder occurs in this order: 1) trying to initialize through dx11(only windows) 2)

trying to initialize through dx9(only windows) 3) trying to initialize through vulkan

To use h.264(AMD VCE) encoder on linux amdgru-pro version 19.20+ and amf-amdgpu-pro package(amdgru-pro contains, but does not install automatically) are required.

This driver can be installed using amdgpu-pro-install script in official amd driver archive.

AviSynth

FFmpeg can read AviSynth scripts as input. To enable support, pass "--enable-avisynth" to configure after installing the headers provided by <<https://github.com/AviSynth/AviSynthPlus>>. AviSynth+ can be configured to install only the headers by either passing "-DHEADERS_ONLY:bool=on" to the normal CMake-based build system, or by using the supplied "GNUmakefile".

For Windows, supported AviSynth variants are <<http://avisynth.nl>> for 32-bit builds and <<http://avisynth.nl/index.php/AviSynth+>> for 32-bit and 64-bit builds.

For Linux, macOS, and BSD, the only supported AviSynth variant is <<https://github.com/AviSynth/AviSynthPlus>>, starting with version 3.5.

In 2016, AviSynth+ added support for building with GCC. However, due to the eccentricities of Windows' calling conventions, 32-bit GCC builds of AviSynth+ are not compatible with typical 32-bit builds of FFmpeg.

By default, FFmpeg assumes compatibility with 32-bit MSVC builds of AviSynth+ since that is the most widely-used and entrenched build configuration. Users can override this and enable support for 32-bit GCC builds of AviSynth+ by passing "-DAVSC_WIN32_GCC32" to "--extra-cflags" when configuring FFmpeg.

64-bit builds of FFmpeg are not affected, and can use either MSVC or GCC builds of AviSynth+ without any special flags.

AviSynth(+) is loaded dynamically. Distributors can build FFmpeg with "--enable-avisynth", and the binaries will work regardless of the end user having AviSynth installed. If/when an end user would like to use AviSynth scripts, then they can install AviSynth(+) and FFmpeg will be able to find and use it to open scripts.

Chromaprint

FFmpeg can make use of the Chromaprint library for generating audio fingerprints. Pass "--enable-chromaprint" to configure to enable it. See <<https://acoustid.org/chromaprint>>.

codec2

FFmpeg can make use of the codec2 library for codec2 decoding and encoding. There is currently no native decoder, so libcodec2 must be used for decoding.

Go to <<http://freedv.org/>>, download "Codec 2 source archive". Build and install using CMake. Debian users can install the libcodec2-dev package instead. Once libcodec2 is installed you can pass "--enable-libcodec2" to configure to enable it.

The easiest way to use codec2 is with .c2 files, since they contain the mode information required for decoding. To encode such a file, use a .c2 file extension and give the libcodec2 encoder the -mode option: "ffmpeg -i input.wav -mode 700C output.c2". Playback is as simple as "ffplay output.c2". For a list of supported modes, run "ffmpeg -h encoder=libcodec2". Raw codec2 files are also supported. To make sense of them the mode in use needs to be specified as a format option: "ffmpeg -f codec2raw -mode 1300 -i input.raw output.wav".

dav1d

FFmpeg can make use of the dav1d library for AV1 video decoding.

Go to <<https://code.videolan.org/videolan/dav1d>> and follow the instructions for installing the library. Then pass "--enable-libdav1d" to configure to enable it.

davs2

FFmpeg can make use of the davs2 library for AVS2-P2/IEEE1857.4 video decoding.

Go to <<https://github.com/pkuvcl/davs2>> and follow the instructions for installing the library. Then pass "--enable-libdavs2" to configure to enable it.

libdavs2 is under the GNU Public License Version 2 or later (see <<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>> for details), you must upgrade FFmpeg's license to GPL in order to use it.

uavs3d

FFmpeg can make use of the uavs3d library for AVS3-P2/IEEE1857.10 video decoding.

Go to <<https://github.com/uavs3/uavs3d>> and follow the instructions for installing the library. Then pass "--enable-libuavs3d" to configure to enable it.

Game Music Emu

FFmpeg can make use of the Game Music Emu library to read audio from supported video game music file formats. Pass "--enable-libgme" to configure to enable it. See

<<https://bitbucket.org/mpyne/game-music-emu/overview>>.

Intel QuickSync Video

FFmpeg can use Intel QuickSync Video (QSV) for accelerated decoding and encoding of multiple codecs. To use QSV, FFmpeg must be linked against the "libmfx" dispatcher, which loads the actual decoding libraries.

The dispatcher is open source and can be downloaded from

<https://github.com/lu-zero/mfx_dispatch.git>. FFmpeg needs to be configured with the "--enable-libmfx" option and "pkg-config" needs to be able to locate the dispatcher's ".pc" files.

Kvazaar

FFmpeg can make use of the Kvazaar library for HEVC encoding.

Go to <<https://github.com/ultravideo/kvazaar>> and follow the instructions for installing the library. Then pass "--enable-libkvazaar" to configure to enable it.

LAME

FFmpeg can make use of the LAME library for MP3 encoding.

Go to <<http://lame.sourceforge.net/>> and follow the instructions for installing the library. Then pass "--enable-libmp3lame" to configure to enable it.

libilbc

iLBC is a narrowband speech codec that has been made freely available by Google as part of the WebRTC project. libilbc is a packaging friendly copy of the iLBC codec. FFmpeg can make use of the libilbc library for iLBC decoding and encoding.

Go to <<https://github.com/TimothyGu/libilbc>> and follow the instructions for installing the library. Then pass "--enable-libilbc" to configure to enable it.

libjxl

JPEG XL is an image format intended to fully replace legacy JPEG for an extended period of life. See <<https://jpegxl.info/>> for more information, and see <<https://github.com/libjxl/libjxl>> for the library source. You can pass "--enable-libjxl" to configure in order enable the libjxl wrapper.

libvpx

FFmpeg can make use of the libvpx library for VP8/VP9 decoding and encoding.

Go to <<http://www.webmproject.org/>> and follow the instructions for installing the library. Then pass

"--enable-libvpx" to configure to enable it.

ModPlug

FFmpeg can make use of this library, originating in Modplug-XMMS, to read from MOD-like music files. See <<https://github.com/Konstanty/libmodplug>>. Pass "--enable-libmodplug" to configure to enable it.

OpenCORE, VisualOn, and Fraunhofer libraries

Spun off Google Android sources, OpenCore, VisualOn and Fraunhofer libraries provide encoders for a number of audio codecs.

OpenCORE and VisualOn libraries are under the Apache License 2.0 (see <<http://www.apache.org/licenses/LICENSE-2.0>> for details), which is incompatible to the LGPL version 2.1 and GPL version 2. You have to upgrade FFmpeg's license to LGPL version 3 (or if you have enabled GPL components, GPL version 3) by passing "--enable-version3" to configure in order to use it.

The license of the Fraunhofer AAC library is incompatible with the GPL. Therefore, for GPL builds, you have to pass "--enable-nonfree" to configure in order to use it. To the best of our knowledge, it is compatible with the LGPL.

OpenCORE AMR

FFmpeg can make use of the OpenCORE libraries for AMR-NB decoding/encoding and AMR-WB decoding.

Go to <<http://sourceforge.net/projects/opencore-amr/>> and follow the instructions for installing the libraries. Then pass "--enable-libopencore-amrnb" and/or "--enable-libopencore-amrwb" to configure to enable them.

VisualOn AMR-WB encoder library

FFmpeg can make use of the VisualOn AMR-WBenc library for AMR-WB encoding.

Go to <<http://sourceforge.net/projects/opencore-amr/>> and follow the instructions for installing the library. Then pass "--enable-libvo-amrwbenc" to configure to enable it.

Fraunhofer AAC library

FFmpeg can make use of the Fraunhofer AAC library for AAC decoding & encoding.

Go to <<http://sourceforge.net/projects/opencore-amr/>> and follow the instructions for installing the library. Then pass "--enable-libfdk-aac" to configure to enable it.

OpenH264

FFmpeg can make use of the OpenH264 library for H.264 decoding and encoding.

Go to <<http://www.openh264.org/>> and follow the instructions for installing the library. Then pass "--enable-libopenh264" to configure to enable it.

For decoding, this library is much more limited than the built-in decoder in libavcodec; currently, this library lacks support for decoding B-frames and some other main/high profile features. (It currently only supports constrained baseline profile and CABAC.) Using it is mostly useful for testing and for taking advantage of Cisco's patent portfolio license (<http://www.openh264.org/BINARY_LICENSE.txt>).

OpenJPEG

FFmpeg can use the OpenJPEG libraries for decoding/encoding J2K videos. Go to <<http://www.openjpeg.org/>> to get the libraries and follow the installation instructions. To enable using OpenJPEG in FFmpeg, pass "--enable-libopenjpeg" to *./configure*.

rav1e

FFmpeg can make use of rav1e (Rust AV1 Encoder) via its C bindings to encode videos. Go to <<https://github.com/xiph/rav1e>> and follow the instructions to build the C library. To enable using rav1e in FFmpeg, pass "--enable-librav1e" to *./configure*.

SVT-AV1

FFmpeg can make use of the Scalable Video Technology for AV1 library for AV1 encoding.

Go to <<https://gitlab.com/AOMediaCodec/SVT-AV1>> and follow the instructions for installing the library. Then pass "--enable-libsvtav1" to configure to enable it.

TwoLAME

FFmpeg can make use of the TwoLAME library for MP2 encoding.

Go to <<http://www.twolame.org/>> and follow the instructions for installing the library. Then pass "--enable-libtwolame" to configure to enable it.

VapourSynth

FFmpeg can read VapourSynth scripts as input. To enable support, pass "--enable-vapoursynth" to configure. Vapoursynth is detected via "pkg-config". Versions 42 or greater supported. See

<<http://www.vapoursynth.com/>>.

Due to security concerns, Vapoursynth scripts will not be autodetected so the input format has to be forced. For ff* CLI tools, add "-f vapoursynth" before the input "-i yoursript.vpy".

x264

FFmpeg can make use of the x264 library for H.264 encoding.

Go to <<http://www.videolan.org/developers/x264.html>> and follow the instructions for installing the library. Then pass "--enable-libx264" to configure to enable it.

x264 is under the GNU Public License Version 2 or later (see <<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>> for details), you must upgrade FFmpeg's license to GPL in order to use it.

x265

FFmpeg can make use of the x265 library for HEVC encoding.

Go to <<http://x265.org/developers.html>> and follow the instructions for installing the library. Then pass "--enable-libx265" to configure to enable it.

x265 is under the GNU Public License Version 2 or later (see <<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>> for details), you must upgrade FFmpeg's license to GPL in order to use it.

xavs

FFmpeg can make use of the xavs library for AVS encoding.

Go to <<http://xavs.sf.net/>> and follow the instructions for installing the library. Then pass "--enable-libxavs" to configure to enable it.

xavs2

FFmpeg can make use of the xavs2 library for AVS2-P2/IEEE1857.4 video encoding.

Go to <<https://github.com/pkuvcl/xavs2>> and follow the instructions for installing the library. Then pass "--enable-libxavs2" to configure to enable it.

libxavs2 is under the GNU Public License Version 2 or later (see <<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>> for details), you must upgrade FFmpeg's license to GPL in order to use it.

ZVBI

ZVBI is a VBI decoding library which can be used by FFmpeg to decode DVB teletext pages and DVB teletext subtitles.

Go to <<http://sourceforge.net/projects/zapping/>> and follow the instructions for installing the library. Then pass "--enable-libzvbi" to configure to enable it.

SUPPORTED FILE FORMATS

You can use the "-formats" and "-codecs" options to have an exhaustive list.

File Formats

FFmpeg supports the following file formats through the "libavformat" library:

Name : Encoding @tab Decoding @tab Comments

3dostr : @tab X

4xm : @tab X

@tab 4X Technologies format, used in some games.

8088flex TMV : @tab X

AAX : @tab X

@tab Audible Enhanced Audio format, used in audiobooks.

AA : @tab X

@tab Audible Format 2, 3, and 4, used in audiobooks.

ACT Voice : @tab X

@tab contains G.729 audio

Adobe Filmstrip : X @tab X

Audio IFF (AIFF) : X @tab X

American Laser Games MM : @tab X

@tab Multimedia format used in games like Mad Dog McCree.

3GPP AMR : X @tab X

Amazing Studio Packed Animation File : @tab X

@tab Multimedia format used in game Heart Of Darkness.

Apple HTTP Live Streaming : @tab X

Artworx Data Format : @tab X

Interplay ACM : @tab X

@tab Audio only format used in some Interplay games.

ADP : @tab X

@tab Audio format used on the Nintendo Gamecube.

AFC : @tab X

@tab Audio format used on the Nintendo Gamecube.

ADS/SS2 : @tab X

@tab Audio format used on the PS2.

APNG : X @tab X

ASF : X @tab X

@tab Advanced / Active Streaming Format.

AST : X @tab X

@tab Audio format used on the Nintendo Wii.

AVI : X @tab X

AviSynth : @tab X

AVR : @tab X

@tab Audio format used on Mac.

AVS : @tab X

@tab Multimedia format used by the Creature Shock game.

Beam Software SIFF : @tab X

@tab Audio and video format used in some games by Beam Software.

Bethesda Softworks VID : @tab X

@tab Used in some games from Bethesda Softworks.

Binary text : @tab X

Bink : @tab X

@tab Multimedia format used by many games.

Bink Audio : @tab X

@tab Audio only multimedia format used by some games.

Bitmap Brothers JV : @tab X

@tab Used in Z and Z95 games.

BRP : @tab X

@tab Argonaut Games format.

Brute Force & Ignorance : @tab X

@tab Used in the game Flash Traffic: City of Angels.

BFSTM : @tab X

@tab Audio format used on the Nintendo WiiU (based on BRSTM).

BRSTM : @tab X

@tab Audio format used on the Nintendo Wii.

BW64 : @tab X

@tab Broadcast Wave 64bit.

BWF : X @tab X

codec2 (raw) : X @tab X

@tab Must be given -mode format option to decode correctly.

codec2 (.c2 files) : X @tab X

@tab Contains header with version and mode info, simplifying playback.

CRI ADX : X @tab X

@tab Audio-only format used in console video games.

CRI AIX : @tab X

CRI HCA : @tab X

@tab Audio-only format used in console video games.

Discworld II BMV : @tab X

Interplay C93 : @tab X

@tab Used in the game Cyberia from Interplay.

Delphine Software International CIN : @tab X

@tab Multimedia format used by Delphine Software games.

Digital Speech Standard (DSS) : @tab X

CD+G : @tab X

@tab Video format used by CD+G karaoke disks

Phantom Cine : @tab X

Commodore CDXL : @tab X

@tab Amiga CD video format

Core Audio Format : X @tab X

@tab Apple Core Audio Format

CRC testing format : X @tab

Creative Voice : X @tab X

@tab Created for the Sound Blaster Pro.

CRYO APC : @tab X

@tab Audio format used in some games by CRYO Interactive Entertainment.

D-Cinema audio : X @tab X

Deluxe Paint Animation : @tab X

DCSTR : @tab X

DFA : @tab X

@tab This format is used in Chronomaster game

DirectDraw Surface : @tab X

DSD Stream File (DSF) : @tab X

DV video : X @tab X

DXA : @tab X

@tab This format is used in the non-Windows version of the Feeble Files game and different game cutscenes repacked for use with ScummVM.

Electronic Arts cdata : @tab X

Electronic Arts Multimedia : @tab X

@tab Used in various EA games; files have extensions like WVE and UV2.

Ensoniq Paris Audio File : @tab X

FFM (FFserver live feed) : X @tab X

Flash (SWF) : X @tab X

Flash 9 (AVM2) : X @tab X

@tab Only embedded audio is decoded.

FLI/FLC/FLX animation : @tab X

@tab .fli/.flc files

Flash Video (FLV) : X @tab X

@tab Macromedia Flash video files

framecrc testing format : X @tab

FunCom ISS : @tab X

@tab Audio format used in various games from FunCom like The Longest Journey.

G.723.1 : X @tab X

G.726 : @tab X @tab Both left- and right-justified.

G.729 BIT : X @tab X

G.729 raw : @tab X

GENH : @tab X

@tab Audio format for various games.

GIF Animation : X @tab X

GXF : X @tab X

@tab General eXchange Format SMPTE 360M, used by Thomson Grass Valley
playout servers.

HNM : @tab X

@tab Only version 4 supported, used in some games from Cryo Interactive

iCEDraw File : @tab X

ICO : X @tab X

@tab Microsoft Windows ICO

id Quake II CIN video : @tab X

id RoQ : X @tab X

@tab Used in Quake III, Jedi Knight 2 and other computer games.

IEC61937 encapsulation : X @tab X

IFF : @tab X

@tab Interchange File Format

IFV : @tab X

@tab A format used by some old CCTV DVRs.

iLBC : X @tab X

Interplay MVE : @tab X

@tab Format used in various Interplay computer games.

Iterated Systems ClearVideo : @tab X

@tab I-frames only

IV8 : @tab X

@tab A format generated by IndigoVision 8000 video server.

IVF (On2) : X @tab X

@tab A format used by libvpx

Internet Video Recording : @tab X

IRCAM : X @tab X

LAF : @tab X

@tab Limitless Audio Format

LATM : X @tab X

LMLM4 : @tab X

@tab Used by Linux Media Labs MPEG-4 PCI boards

LOAS : @tab X

@tab contains LATM multiplexed AAC audio

LRC : X @tab X

LVF : @tab X

LXF : @tab X

@tab VR native stream format, used by Leitch/Harris' video servers.

Magic Lantern Video (MLV) : @tab X

Matroska : X @tab X

Matroska audio : X @tab

FFmpeg metadata : X @tab X

@tab Metadata in text format.

MAXIS XA : @tab X

@tab Used in Sim City 3000; file extension .xa.

MCA : @tab X

@tab Used in some games from Capcom; file extension .mca.

MD Studio : @tab X
Metal Gear Solid: The Twin Snakes : @tab X
Megalux Frame : @tab X
 @tab Used by Megalux Ultimate Paint

MobiClip MODS : @tab X
MobiClip MOFLEX : @tab X
Mobotix .mxg : @tab X
Monkey's Audio : @tab X
Motion Pixels MVI : @tab X
MOV/QuickTime/MP4 : X @tab X
 @tab 3GP, 3GP2, PSP, iPod variants supported

MP2 : X @tab X
MP3 : X @tab X
MPEG-1 System : X @tab X
 @tab muxed audio and video, VCD format supported

MPEG-PS (program stream) : X @tab X
 @tab also known as C<VOB> file, SVCD and DVD format supported

MPEG-TS (transport stream) : X @tab X
 @tab also known as DVB Transport Stream

MPEG-4 : X @tab X
 @tab MPEG-4 is a variant of QuickTime.

MSF : @tab X
 @tab Audio format used on the PS3.

Mirillis FIC video : @tab X
 @tab No cursor rendering.

MIDI Sample Dump Standard : @tab X
MIME multipart JPEG : X @tab
MSN TCP webcam : @tab X
 @tab Used by MSN Messenger webcam streams.

MTV : @tab X
Musepack : @tab X

Musepack SV8 : @tab X
Material eXchange Format (MXF) : X @tab X
 @tab SMPTE 377M, used by D-Cinema, broadcast industry.

Material eXchange Format (MXF), D-10 Mapping : X @tab X
 @tab SMPTE 386M, D-10/IMX Mapping.

NC camera feed : @tab X
 @tab NC (AVIP NC4600) camera streams

NIST SPeech HEader REsources : @tab X
Computerized Speech Lab NSP : @tab X
NTT TwinVQ (VQF) : @tab X
 @tab Nippon Telegraph and Telephone Corporation TwinVQ.

Nullsoft Streaming Video : @tab X
NuppelVideo : @tab X
NUT : X @tab X
 @tab NUT Open Container Format

Ogg : X @tab X
Playstation Portable PMP : @tab X
Portable Voice Format : @tab X
RK Audio (RKA) : @tab X
TechnoTrend PVA : @tab X
 @tab Used by TechnoTrend DVB PCI boards.

QCP : @tab X
raw ADTS (AAC) : X @tab X
raw AC-3 : X @tab X
raw AMR-NB : @tab X
raw AMR-WB : @tab X
raw APAC : @tab X
raw aptX : X @tab X
raw aptX HD : X @tab X
raw Bonk : @tab X
raw Chinese AVS video : X @tab X
raw DFPWM : X @tab X
raw Dirac : X @tab X
raw DNxHD : X @tab X

```

raw DTS           : X @tab X
raw DTS-HD       : @tab X
raw E-AC-3       : X @tab X
raw FLAC         : X @tab X
raw GSM          : @tab X
raw H.261        : X @tab X
raw H.263        : X @tab X
raw H.264        : X @tab X
raw HEVC         : X @tab X
raw Ingenient MJPEG : @tab X
raw MJPEG        : X @tab X
raw MLP          : @tab X
raw MPEG         : @tab X
raw MPEG-1       : @tab X
raw MPEG-2       : @tab X
raw MPEG-4       : X @tab X
raw NULL         : X @tab
raw video        : X @tab X
raw id RoQ       : X @tab
raw OBU          : X @tab X
raw SBC          : X @tab X
raw Shorten      : @tab X
raw TAK          : @tab X
raw TrueHD       : X @tab X
raw VC-1         : X @tab X
raw PCM A-law    : X @tab X
raw PCM mu-law   : X @tab X
raw PCM Archimedes VIDC : X @tab X
raw PCM signed 8 bit : X @tab X
raw PCM signed 16 bit big-endian : X @tab X
raw PCM signed 16 bit little-endian : X @tab X
raw PCM signed 24 bit big-endian : X @tab X
raw PCM signed 24 bit little-endian : X @tab X
raw PCM signed 32 bit big-endian : X @tab X
raw PCM signed 32 bit little-endian : X @tab X
raw PCM signed 64 bit big-endian : X @tab X
raw PCM signed 64 bit little-endian : X @tab X
raw PCM unsigned 8 bit : X @tab X
raw PCM unsigned 16 bit big-endian : X @tab X
raw PCM unsigned 16 bit little-endian : X @tab X

```

raw PCM unsigned 24 bit big-endian : X @tab X
raw PCM unsigned 24 bit little-endian : X @tab X
raw PCM unsigned 32 bit big-endian : X @tab X
raw PCM unsigned 32 bit little-endian : X @tab X
raw PCM 16.8 floating point little-endian : @tab X
raw PCM 24.0 floating point little-endian : @tab X
raw PCM floating-point 32 bit big-endian : X @tab X
raw PCM floating-point 32 bit little-endian : X @tab X
raw PCM floating-point 64 bit big-endian : X @tab X
raw PCM floating-point 64 bit little-endian : X @tab X
RDT : @tab X
REDCODE R3D : @tab X

@tab File format used by RED Digital cameras, contains JPEG 2000 frames and PCM audio.

RealMedia : X @tab X
Redirector : @tab X
RedSpark : @tab X
Renderware TeXture Dictionary : @tab X
Resolume DXV : @tab X
RF64 : @tab X
RL2 : @tab X

@tab Audio and video format used in some games by Entertainment Software Partners.

RPL/ARMovie : @tab X
Lego Mindstorms RSO : X @tab X
RSD : @tab X
RTMP : X @tab X

@tab Output is performed by publishing stream to RTMP server

RTP : X @tab X
RTSP : X @tab X
Sample Dump eXchange : @tab X
SAP : X @tab X
SBG : @tab X
SDNS : @tab X
SDP : @tab X
SER : @tab X
Digital Pictures SGA : @tab X
Sega FILM/CPK : X @tab X

@tab Used in many Sega Saturn console games.

Silicon Graphics Movie : @tab X

Sierra SOL : @tab X

@tab .sol files used in Sierra Online games.

Sierra VMD : @tab X

@tab Used in Sierra CD-ROM games.

Smacker : @tab X

@tab Multimedia format used by many games.

SMJPEG : X @tab X

@tab Used in certain Loki game ports.

SMPTE 337M encapsulation : @tab X

Smush : @tab X

@tab Multimedia format used in some LucasArts games.

Sony OpenMG (OMA) : X @tab X

@tab Audio format used in Sony Sonic Stage and Sony Vegas.

Sony PlayStation STR : @tab X

Sony Wave64 (W64) : X @tab X

SoX native format : X @tab X

SUN AU format : X @tab X

SUP raw PGS subtitles : X @tab X

SVAG : @tab X

@tab Audio format used in Konami PS2 games.

TDSC : @tab X

Text files : @tab X

THP : @tab X

@tab Used on the Nintendo GameCube.

Tiertex Limited SEQ : @tab X

@tab Tiertex .seq files used in the DOS CD-ROM version of the game Flashback.

True Audio : X @tab X

VAG : @tab X

@tab Audio format used in many Sony PS2 games.

VC-1 test bitstream : X @tab X

Vidvox Hap : X @tab X

Vivo : @tab X

VPK : @tab X

@tab Audio format used in Sony PS games.

Marble WADY : @tab X

WAV : X @tab X

Waveform Archiver : @tab X

WavPack : X @tab X

WebM : X @tab X

Windows Televison (WTV) : X @tab X

Wing Commander III movie : @tab X

@tab Multimedia format used in Origin's Wing Commander III computer game.

Westwood Studios audio : X @tab X

@tab Multimedia format used in Westwood Studios games.

Westwood Studios VQA : @tab X

@tab Multimedia format used in Westwood Studios games.

Wideband Single-bit Data (WSD) : @tab X

WVE : @tab X

Konami XMD : @tab X

XMV : @tab X

@tab Microsoft video container used in Xbox games.

XVAG : @tab X

@tab Audio format used on the PS3.

xWMA : @tab X

@tab Microsoft audio container used by XAudio 2.

eXtended BINary text (XBIN) : @tab X

YUV4MPEG pipe : X @tab X

Psygnosis YOP : @tab X

"X" means that the feature in that column (encoding / decoding) is supported.

Image Formats

Ffmpeg can read and write images for each frame of a video sequence. The following image formats are supported:

Name : Encoding @tab Decoding @tab Comments

.Y.U.V : X @tab X

@tab one raw file per component

Alias PIX : X @tab X

@tab Alias/Wavefront PIX image format

animated GIF : X @tab X

APNG : X @tab X

@tab Animated Portable Network Graphics

BMP : X @tab X

@tab Microsoft BMP image

BRender PIX : @tab X

@tab Argonaut BRender 3D engine image format.

CRI : @tab X

@tab Cintel RAW

DPX : X @tab X

@tab Digital Picture Exchange

EXR : @tab X

@tab OpenEXR

FITS : X @tab X

@tab Flexible Image Transport System

HDR : X @tab X

@tab Radiance HDR RGBE Image format

IMG : @tab X

@tab GEM Raster image

JPEG : X @tab X

@tab Progressive JPEG is not supported.

JPEG 2000 : X @tab X

JPEG-LS : X @tab X

LJPEG : X @tab
@tab Lossless JPEG

Media 100 : @tab X

MSP : @tab X
@tab Microsoft Paint image

PAM : X @tab X
@tab PAM is a PNM extension with alpha support.

PBM : X @tab X
@tab Portable BitMap image

PCD : @tab X
@tab PhotoCD

PCX : X @tab X
@tab PC Paintbrush

PFM : X @tab X
@tab Portable FloatMap image

PGM : X @tab X
@tab Portable GrayMap image

PGMYUV : X @tab X
@tab PGM with U and V components in YUV 4:2:0

PGX : @tab X
@tab PGX file decoder

PHM : X @tab X
@tab Portable HalfFloatMap image

PIC : @tab X
@tab Pictor/PC Paint

PNG : X @tab X

@tab Portable Network Graphics image

PPM : X @tab X

@tab Portable PixelMap image

PSD : @tab X

@tab Photoshop

PTX : @tab X

@tab V.Flash PTX format

QOI : X @tab X

@tab Quite OK Image format

SGI : X @tab X

@tab SGI RGB image format

Sun Rasterfile : X @tab X

@tab Sun RAS image format

TIFF : X @tab X

@tab YUV, JPEG and some extension is not supported yet.

Truevision Targa : X @tab X

@tab Targa (.TGA) image format

VBN : X @tab X

@tab Vizrt Binary Image format

WBMP : X @tab X

@tab Wireless Application Protocol Bitmap image format

WebP : E @tab X

@tab WebP image format, encoding supported through external library libwebp

XBM : X @tab X

@tab X BitMap image format

XFace : X @tab X

@tab X-Face image format

XPM : @tab X
 @tab X Pixmap image format

XWD : X @tab X
 @tab X Window Dump image format

"X" means that the feature in that column (encoding / decoding) is supported.

"E" means that support is provided through an external library.

Video Codecs

Name : **Encoding** @tab **Decoding** @tab **Comments**

4X Movie : @tab X
 @tab Used in certain computer games.

8088flex TMV : @tab X
A64 multicolor : X @tab
 @tab Creates video suitable to be played on a commodore 64 (multicolor mode).

Amazing Studio PAF Video : @tab X
American Laser Games MM : @tab X
 @tab Used in games like Mad Dog McCree.

Amuse Graphics Movie : @tab X
AMV Video : X @tab X
 @tab Used in Chinese MP3 players.

ANSI/ASCII art : @tab X
Apple Intermediate Codec : @tab X
Apple MJPEG-B : @tab X
Apple Pixlet : @tab X
Apple ProRes : X @tab X
 @tab fourcc: apch,apcn,apcs,apco,ap4h,ap4x

Apple QuickDraw : @tab X
 @tab fourcc: qdrw

Argonaut Video : @tab X
 @tab Used in some Argonaut games.

Asus v1 : **X @tab X**
@tab fourcc: ASV1

Asus v2 : **X @tab X**
@tab fourcc: ASV2

ATI VCR1 : **@tab X**
@tab fourcc: VCR1

ATI VCR2 : **@tab X**
@tab fourcc: VCR2

Auravision Aura : **@tab X**
Auravision Aura 2 : **@tab X**
Autodesk Animator Flic video : **@tab X**
Autodesk RLE : **@tab X**
@tab fourcc: AASC

AV1 : **E @tab E**
@tab Supported through external libraries libaom, libdav1d, librav1e and libsvtav1

Avid 1:1 10-bit RGB Packer : **X @tab X**
@tab fourcc: AVrp

AVS (Audio Video Standard) video : **@tab X**
@tab Video encoding used by the Creature Shock game.

AVS2-P2/IEEE1857.4 : **E @tab E**
@tab Supported through external libraries libxavs2 and libdavs2

AVS3-P2/IEEE1857.10 : **@tab E**
@tab Supported through external library libuavs3d

AYUV : **X @tab X**
@tab Microsoft uncompressed packed 4:4:4:4

Beam Software VB : **@tab X**
Bethesda VID video : **@tab X**
@tab Used in some games from Bethesda Softworks.

Bink Video : @tab X
BitJazz SheerVideo : @tab X
Bitmap Brothers JV video : @tab X
y41p Brooktree uncompressed 4:1:1 12-bit : X @tab X
Brooktree Prosumer Video : @tab X
@tab fourcc: BT20

Brute Force & Ignorance : @tab X
@tab Used in the game Flash Traffic: City of Angels.

C93 video : @tab X
@tab Codec used in Cyberia game.

CamStudio : @tab X
@tab fourcc: CSCD

CD+G : @tab X
@tab Video codec for CD+G karaoke disks

CDXL : @tab X
@tab Amiga CD video codec

Chinese AVS video : E @tab X
@tab AVS1-P2, JiZhun profile, encoding through external library libxavs

Delphine Software International CIN video : @tab X
@tab Codec used in Delphine Software International games.

Discworld II BMV Video : @tab X
CineForm HD : X @tab X
Canopus HQ : @tab X
Canopus HQA : @tab X
Canopus HQX : @tab X
Canopus Lossless Codec : @tab X
CDToons : @tab X
@tab Codec used in various Broderbund games.

Cinepak : @tab X
Cirrus Logic AccuPak : X @tab X
@tab fourcc: CLJR

CPiA Video Format : @tab X
Creative YUV (CYUV) : @tab X
DFA : @tab X
 @tab Codec used in Chronomaster game.

Dirac : E @tab X
 @tab supported though the native vc2 (Dirac Pro) encoder

Deluxe Paint Animation : @tab X
DNxHD : X @tab X
 @tab aka SMPTE VC3

Duck TrueMotion 1.0 : @tab X
 @tab fourcc: DUCK

Duck TrueMotion 2.0 : @tab X
 @tab fourcc: TM20

Duck TrueMotion 2.0 RT : @tab X
 @tab fourcc: TR20

DV (Digital Video) : X @tab X
Dxtory capture format : @tab X
Feeble Files/ScummVM DXA : @tab X
 @tab Codec originally used in Feeble Files game.

Electronic Arts CMV video : @tab X
 @tab Used in NHL 95 game.

Electronic Arts Madcow video : @tab X
Electronic Arts TGV video : @tab X
Electronic Arts TGQ video : @tab X
Electronic Arts TQI video : @tab X
Escape 124 : @tab X
Escape 130 : @tab X
FFmpeg video codec #1 : X @tab X
 @tab lossless codec (fourcc: FFV1)

Flash Screen Video v1 : X @tab X
 @tab fourcc: FSV1

Flash Screen Video v2 : X @tab X

Flash Video (FLV) : X @tab X

@tab Sorenson H.263 used in Flash

FM Screen Capture Codec : @tab X

Forward Uncompressed : @tab X

Fraps : @tab X

Go2Meeting : @tab X

@tab fourcc: G2M2, G2M3

Go2Webinar : @tab X

@tab fourcc: G2M4

Gremlin Digital Video : @tab X

H.261 : X @tab X

H.263 / H.263-1996 : X @tab X

H.263+ / H.263-1998 / H.263 version 2 : X @tab X

H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 : E @tab X

@tab encoding supported through external library libx264 and OpenH264

HEVC : X @tab X

@tab encoding supported through external library libx265 and libkvazaar

HNM version 4 : @tab X

HuffYUV : X @tab X

HuffYUV FFmpeg variant : X @tab X

IBM Ultimotion : @tab X

@tab fourcc: ULTI

id Cinematic video : @tab X

@tab Used in Quake II.

id RoQ video : X @tab X

@tab Used in Quake III, Jedi Knight 2, other computer games.

IFF ILBM : @tab X

@tab IFF interleaved bitmap

IFF ByteRun1 : @tab X

@tab IFF run length encoded bitmap

Infinity IMM4 : @tab X
Intel H.263 : @tab X
Intel Indeo 2 : @tab X
Intel Indeo 3 : @tab X
Intel Indeo 4 : @tab X
Intel Indeo 5 : @tab X
Interplay C93 : @tab X

@tab Used in the game Cyberia from Interplay.

Interplay MVE video : @tab X

@tab Used in Interplay .MVE files.

J2K : X @tab X

Karl Morton's video codec : @tab X

@tab Codec used in Worms games.

Kega Game Video (KGV1) : @tab X

@tab Kega emulator screen capture codec.

Lagarith : @tab X

LCL (LossLess Codec Library) MSZH : @tab X

LCL (LossLess Codec Library) ZLIB : E @tab E

LOCO : @tab X

LucasArts SANM/Smush : @tab X

@tab Used in LucasArts games / SMUSH animations.

lossless MJPEG : X @tab X

MagicYUV Video : X @tab X

Mandssoft Screen Capture Codec : @tab X

Microsoft ATC Screen : @tab X

@tab Also known as Microsoft Screen 3.

Microsoft Expression Encoder Screen : @tab X

@tab Also known as Microsoft Titanium Screen 2.

Microsoft RLE : @tab X

Microsoft Screen 1 : @tab X

@tab Also known as Windows Media Video V7 Screen.

Microsoft Screen 2 : @tab X

@tab Also known as Windows Media Video V9 Screen.

Microsoft Video 1 : @tab X

Mimic : @tab X

@tab Used in MSN Messenger Webcam streams.

Miro VideoXL : @tab X

@tab fourcc: VIXL

MJPEG (Motion JPEG) : X @tab X

Robotix MxPEG video : @tab X

Motion Pixels video : @tab X

MPEG-1 video : X @tab X

MPEG-2 video : X @tab X

MPEG-4 part 2 : X @tab X

@tab libxvidcore can be used alternatively for encoding.

MPEG-4 part 2 Microsoft variant version 1 : @tab X

MPEG-4 part 2 Microsoft variant version 2 : X @tab X

MPEG-4 part 2 Microsoft variant version 3 : X @tab X

Newtek SpeedHQ : X @tab X

Nintendo Gamecube THP video : @tab X

NotchLC : @tab X

NuppelVideo/RTjpeg : @tab X

@tab Video encoding used in NuppelVideo files.

On2 VP3 : @tab X

@tab still experimental

On2 VP4 : @tab X

@tab fourcc: VP40

On2 VP5 : @tab X

@tab fourcc: VP50

On2 VP6 : @tab X

@tab fourcc: VP60,VP61,VP62

On2 VP7 : @tab X

@tab fourcc: VP70,VP71

VP8 : E @tab X

@tab fourcc: VP80, encoding supported through external library libvpx

VP9 : E @tab X

@tab encoding supported through external library libvpx

Pinnacle TARGA CineWave YUV16 : @tab X

@tab fourcc: Y216

Q-team QPEG : @tab X

@tab fourccs: QPEG, Q1.0, Q1.1

QuickTime 8BPS video : @tab X

QuickTime Animation (RLE) video : X @tab X

@tab fourcc: 'rle '

QuickTime Graphics (SMC) : X @tab X

@tab fourcc: 'smc '

QuickTime video (RPZA) : X @tab X

@tab fourcc: rpza

R10K AJA Kona 10-bit RGB Codec : X @tab X

R210 Quicktime Uncompressed RGB 10-bit : X @tab X

Raw Video : X @tab X

RealVideo 1.0 : X @tab X

RealVideo 2.0 : X @tab X

RealVideo 3.0 : @tab X

@tab still far from ideal

RealVideo 4.0 : @tab X

Renderware TXD (TeXture Dictionary) : @tab X

@tab Texture dictionaries used by the Renderware Engine.

RL2 video : @tab X

@tab used in some games by Entertainment Software Partners

ScreenPressor : @tab X

Screenpresso : @tab X

Screen Recorder Gold Codec : @tab X

Sierra VMD video : @tab X

@tab Used in Sierra VMD files.

Silicon Graphics Motion Video Compressor 1 (MVC1) : @tab X

Silicon Graphics Motion Video Compressor 2 (MVC2) : @tab X

Silicon Graphics RLE 8-bit video : @tab X

Smacker video : @tab X

@tab Video encoding used in Smacker.

SMPTE VC-1 : @tab X

Snow : X @tab X

@tab experimental wavelet codec (fourcc: SNOW)

Sony PlayStation MDEC (Motion DECoder) : @tab X

Sorenson Vector Quantizer 1 : X @tab X

@tab fourcc: SVQ1

Sorenson Vector Quantizer 3 : @tab X

@tab fourcc: SVQ3

Sunplus JPEG (SP5X) : @tab X

@tab fourcc: SP5X

TechSmith Screen Capture Codec : @tab X

@tab fourcc: TSCC

TechSmith Screen Capture Codec 2 : @tab X

@tab fourcc: TSC2

Theora : E @tab X

@tab encoding supported through external library libtheora

Tiertex Limited SEQ video : @tab X

@tab Codec used in DOS CD-ROM FlashBack game.

Ut Video : X @tab X

v210 QuickTime uncompressed 4:2:2 10-bit : X @tab X

v308 QuickTime uncompressed 4:4:4 : X @tab X

v408 QuickTime uncompressed 4:4:4:4 : X @tab X

v410 QuickTime uncompressed 4:4:4 10-bit : X @tab X

VBLE Lossless Codec : @tab X
VMware Screen Codec / VMware Video : @tab X
 @tab Codec used in videos captured by VMware.

Westwood Studios VQA (Vector Quantized Animation) video : @tab X
Windows Media Image : @tab X
Windows Media Video 7 : X @tab X
Windows Media Video 8 : X @tab X
Windows Media Video 9 : @tab X
 @tab not completely working

Wing Commander III / Xan : @tab X
 @tab Used in Wing Commander III .MVE files.

Wing Commander IV / Xan : @tab X
 @tab Used in Wing Commander IV.

Winnov WNV1 : @tab X
WMV7 : X @tab X
YAMAHA SMAF : X @tab X
Psygnosis YOP Video : @tab X
yuv4 : X @tab X
 @tab libquicktime uncompressed packed 4:2:0

ZeroCodec Lossless Video : @tab X
ZLIB : X @tab X
 @tab part of LCL, encoder experimental

Zip Motion Blocks Video : X @tab X
 @tab Encoder works only in PAL8.

"X" means that the feature in that column (encoding / decoding) is supported.

"E" means that support is provided through an external library.

Audio Codecs

Name : **Encoding** @tab **Decoding** @tab **Comments**
8SVX exponential : @tab X
8SVX fibonacci : @tab X
AAC : EX @tab X

@tab encoding supported through internal encoder and external library libfdk-aac

AAC+ : **E @tab IX**

@tab encoding supported through external library libfdk-aac

AC-3 : **IX @tab IX**

ACELP.KELVIN : **@tab X**

ADPCM 4X Movie : **@tab X**

ADPCM Yamaha AICA : **@tab X**

ADPCM AmuseGraphics Movie : **@tab X**

ADPCM Argonaut Games : **X @tab X**

ADPCM CDROM XA : **@tab X**

ADPCM Creative Technology : **@tab X**

@tab 16 -E<gt> 4, 8 -E<gt> 4, 8 -E<gt> 3, 8 -E<gt> 2

ADPCM Electronic Arts : **@tab X**

@tab Used in various EA titles.

ADPCM Electronic Arts Maxis CDROM XS : **@tab X**

@tab Used in Sim City 3000.

ADPCM Electronic Arts R1 : **@tab X**

ADPCM Electronic Arts R2 : **@tab X**

ADPCM Electronic Arts R3 : **@tab X**

ADPCM Electronic Arts XAS : **@tab X**

ADPCM G.722 : **X @tab X**

ADPCM G.726 : **X @tab X**

ADPCM IMA Acorn Replay : **@tab X**

ADPCM IMA AMV : **X @tab X**

@tab Used in AMV files

ADPCM IMA Cunning Developments : **@tab X**

ADPCM IMA Electronic Arts EACS : **@tab X**

ADPCM IMA Electronic Arts SEAD : **@tab X**

ADPCM IMA Funcom : **@tab X**

ADPCM IMA High Voltage Software ALP : **X @tab X**

ADPCM IMA Mobiclip MOFLEX : **@tab X**

ADPCM IMA QuickTime : **X @tab X**

ADPCM IMA Simon & Schuster Interactive : **X @tab X**

ADPCM IMA Ubisoft APM : **X @tab X**

ADPCM IMA Loki SDL MJPEG : @tab X

ADPCM IMA WAV : X @tab X

ADPCM IMA Westwood : @tab X

ADPCM ISS IMA : @tab X

@tab Used in FunCom games.

ADPCM IMA Dialogic : @tab X

ADPCM IMA Duck DK3 : @tab X

@tab Used in some Sega Saturn console games.

ADPCM IMA Duck DK4 : @tab X

@tab Used in some Sega Saturn console games.

ADPCM IMA Radical : @tab X

ADPCM Microsoft : X @tab X

ADPCM MS IMA : X @tab X

ADPCM Nintendo Gamecube AFC : @tab X

ADPCM Nintendo Gamecube DTK : @tab X

ADPCM Nintendo THP : @tab X

ADPCM Playstation : @tab X

ADPCM QT IMA : X @tab X

ADPCM SEGA CRI ADX : X @tab X

@tab Used in Sega Dreamcast games.

ADPCM Shockwave Flash : X @tab X

ADPCM Sound Blaster Pro 2-bit : @tab X

ADPCM Sound Blaster Pro 2.6-bit : @tab X

ADPCM Sound Blaster Pro 4-bit : @tab X

ADPCM VIMA : @tab X

@tab Used in LucasArts SMUSH animations.

ADPCM Konami XMD : @tab X

ADPCM Westwood Studios IMA : X @tab X

@tab Used in Westwood Studios games like Command and Conquer.

ADPCM Yamaha : X @tab X

ADPCM Zork : @tab X

AMR-NB : E @tab X

@tab encoding supported through external library libopencore-amrnb

AMR-WB : **E @tab X**

@tab encoding supported through external library libvo-amrwbenc

Amazing Studio PAF Audio : **@tab X**

Apple lossless audio : **X @tab X**

@tab QuickTime fourcc 'alac'

aptX : **X @tab X**

@tab Used in Bluetooth A2DP

aptX HD : **X @tab X**

@tab Used in Bluetooth A2DP

ATRAC1 : **@tab X**

ATRAC3 : **@tab X**

ATRAC3+ : **@tab X**

ATRAC9 : **@tab X**

Bink Audio : **@tab X**

@tab Used in Bink and Smacker files in many games.

Bonk audio : **@tab X**

CELT : **@tab E**

@tab decoding supported through external library libcelt

codec2 : **E @tab E**

@tab en/decoding supported through external library libcodec2

CRI HCA : **@tab X**

Delphine Software International CIN audio : **@tab X**

@tab Codec used in Delphine Software International games.

DFPWM : **X @tab X**

Digital Speech Standard - Standard Play mode (DSS SP) : **@tab X**

Discworld II BMV Audio : **@tab X**

COOK : **@tab X**

@tab All versions except 5.1 are supported.

DCA (DTS Coherent Acoustics) : **X @tab X**

@tab supported extensions: XCh, XXCH, X96, XBR, XLL, LBR (partially)

Dolby E : @tab X

DPCM Cuberoot-Delta-Exact : @tab X

@tab Used in few games.

DPCM Gremlin : @tab X

DPCM id RoQ : X @tab X

@tab Used in Quake III, Jedi Knight 2 and other computer games.

DPCM Marble WADY : @tab X

DPCM Interplay : @tab X

@tab Used in various Interplay computer games.

DPCM Squareroot-Delta-Exact : @tab X

@tab Used in various games.

DPCM Sierra Online : @tab X

@tab Used in Sierra Online game audio files.

DPCM Sol : @tab X

DPCM Xan : @tab X

@tab Used in Origin's Wing Commander IV AVI files.

DPCM Xilam DERF : @tab X

DSD (Direct Stream Digital), least significant bit first : @tab X

DSD (Direct Stream Digital), most significant bit first : @tab X

DSD (Direct Stream Digital), least significant bit first, planar : @tab X

DSD (Direct Stream Digital), most significant bit first, planar : @tab X

DSP Group TrueSpeech : @tab X

DST (Direct Stream Transfer) : @tab X

DV audio : @tab X

Enhanced AC-3 : X @tab X

EVRC (Enhanced Variable Rate Codec) : @tab X

FLAC (Free Lossless Audio Codec) : X @tab IX

FTR Voice : @tab X

G.723.1 : X @tab X

G.729 : @tab X

GSM : E @tab X

@tab encoding supported through external library libgsm

GSM Microsoft variant : E @tab X

@tab encoding supported through external library libgsm

IAC (Indeo Audio Coder) : @tab X

iLBC (Internet Low Bitrate Codec) : E @tab EX

@tab encoding and decoding supported through external library libilbc

IMC (Intel Music Coder) : @tab X

Interplay ACM : @tab X

MACE (Macintosh Audio Compression/Expansion) 3:1 : @tab X

MACE (Macintosh Audio Compression/Expansion) 6:1 : @tab X

Marian's A-pac audio : @tab X

MI-SC4 (Micronas SC-4 Audio) : @tab X

MLP (Meridian Lossless Packing) : X @tab X

@tab Used in DVD-Audio discs.

Monkey's Audio : @tab X

MP1 (MPEG audio layer 1) : @tab IX

MP2 (MPEG audio layer 2) : IX @tab IX

@tab encoding supported also through external library TwoLAME

MP3 (MPEG audio layer 3) : E @tab IX

@tab encoding supported through external library LAME, ADU MP3 and MP3onMP4 also supported

MPEG-4 Audio Lossless Coding (ALS) : @tab X

MobiClip FastAudio : @tab X

Musepack SV7 : @tab X

Musepack SV8 : @tab X

Nellymoser Asao : X @tab X

On2 AVC (Audio for Video Codec) : @tab X

Opus : E @tab X

@tab encoding supported through external library libopus

PCM A-law : X @tab X

PCM mu-law : X @tab X

PCM Archimedes VIDC : X @tab X

PCM signed 8-bit planar : X @tab X

PCM signed 16-bit big-endian planar : X @tab X

PCM signed 16-bit little-endian planar : X @tab X

PCM signed 24-bit little-endian planar : X @tab X

PCM signed 32-bit little-endian planar : X @tab X

PCM 32-bit floating point big-endian : X @tab X
PCM 32-bit floating point little-endian : X @tab X
PCM 64-bit floating point big-endian : X @tab X
PCM 64-bit floating point little-endian : X @tab X
PCM D-Cinema audio signed 24-bit : X @tab X
PCM signed 8-bit : X @tab X
PCM signed 16-bit big-endian : X @tab X
PCM signed 16-bit little-endian : X @tab X
PCM signed 24-bit big-endian : X @tab X
PCM signed 24-bit little-endian : X @tab X
PCM signed 32-bit big-endian : X @tab X
PCM signed 32-bit little-endian : X @tab X
PCM signed 16/20/24-bit big-endian in MPEG-TS : @tab X
PCM unsigned 8-bit : X @tab X
PCM unsigned 16-bit big-endian : X @tab X
PCM unsigned 16-bit little-endian : X @tab X
PCM unsigned 24-bit big-endian : X @tab X
PCM unsigned 24-bit little-endian : X @tab X
PCM unsigned 32-bit big-endian : X @tab X
PCM unsigned 32-bit little-endian : X @tab X
PCM SGA : @tab X
QCELP / PureVoice : @tab X
QDesign Music Codec 1 : @tab X
QDesign Music Codec 2 : @tab X

@tab There are still some distortions.

RealAudio 1.0 (14.4K) : X @tab X

@tab Real 14400 bit/s codec

RealAudio 2.0 (28.8K) : @tab X

@tab Real 28800 bit/s codec

RealAudio 3.0 (dnet) : IX @tab X

@tab Real low bitrate AC-3 codec

RealAudio Lossless : @tab X

RealAudio SIPR / ACELP.NET : @tab X

RK Audio (RKA) : @tab X

SBC (low-complexity subband codec) : X @tab X

@tab Used in Bluetooth A2DP

Shorten : @tab X
Sierra VMD audio : @tab X
 @tab Used in Sierra VMD files.

Smacker audio : @tab X
SMPTE 302M AES3 audio : X @tab X
Sonic : X @tab X
 @tab experimental codec

Sonic lossless : X @tab X
 @tab experimental codec

Speex : E @tab EX
 @tab supported through external library libspeex

TAK (Tom's lossless Audio Kompressor) : @tab X
True Audio (TTA) : X @tab X
TrueHD : X @tab X
 @tab Used in HD-DVD and Blu-Ray discs.

TwinVQ (VQF flavor) : @tab X
VIMA : @tab X
 @tab Used in LucasArts SMUSH animations.

ViewQuest VQC : @tab X
Vorbis : E @tab X
 @tab A native but very primitive encoder exists.

Voxware MetaSound : @tab X
Waveform Archiver : @tab X
WavPack : X @tab X
Westwood Audio (SND1) : @tab X
Windows Media Audio 1 : X @tab X
Windows Media Audio 2 : X @tab X
Windows Media Audio Lossless : @tab X
Windows Media Audio Pro : @tab X
Windows Media Audio Voice : @tab X
Xbox Media Audio 1 : @tab X
Xbox Media Audio 2 : @tab X

"X" means that the feature in that column (encoding / decoding) is supported.

"E" means that support is provided through an external library.

"I" means that an integer-only version is available, too (ensures high performance on systems without hardware floating point support).

Subtitle Formats

Name : Muxing @tab Demuxing @tab Encoding @tab Decoding

3GPP Timed Text : @tab @tab X @tab X

AQTitle : @tab X @tab @tab X

DVB : X @tab X @tab X @tab X

DVB teletext : @tab X @tab @tab E

DVD : X @tab X @tab X @tab X

JACOsub : X @tab X @tab @tab X

MicroDVD : X @tab X @tab @tab X

MPL2 : @tab X @tab @tab X

MPSub (MPlayer) : @tab X @tab @tab X

PGS : @tab @tab @tab X

PJS (Phoenix) : @tab X @tab @tab X

RealText : @tab X @tab @tab X

SAMI : @tab X @tab @tab X

Spruce format (STL) : @tab X @tab @tab X

SSA/ASS : X @tab X @tab X @tab X

SubRip (SRT) : X @tab X @tab X @tab X

SubViewer v1 : @tab X @tab @tab X

SubViewer : @tab X @tab @tab X

TED Talks captions : @tab X @tab @tab X

TTML : X @tab @tab X @tab

VobSub (IDX+SUB) : @tab X @tab @tab X

VPlayer : @tab X @tab @tab X

WebVTT : X @tab X @tab X @tab X

XSUB : @tab @tab X @tab X

"X" means that the feature is supported.

"E" means that support is provided through an external library.

Network Protocols

Name : Support

```

AMQP      : E
file     : X
FTP      : X
Gopher   : X
Gophers  : X
HLS      : X
HTTP     : X
HTTPS    : X
Icecast  : X
MMSH    : X
MMST    : X
pipe     : X
Pro-MPEG FEC : X
RTMP     : X
RTMPE    : X
RTMPS    : X
RTMPT    : X
RTMPTE   : X
RTMPTS   : X
RTP      : X
SAMBA    : E
SCTP     : X
SFTP     : E
TCP      : X
TLS      : X
UDP      : X
ZMQ      : E

```

"X" means that the protocol is supported.

"E" means that support is provided through an external library.

Input/Output Devices

```

Name      : Input @tab Output
ALSA     : X @tab X
BKTR     : X @tab
caca     : @tab X
DV1394   : X @tab
Lavfi virtual device : X @tab
Linux framebuffer : X @tab X

```



```

JACK      : X  @tab
LIBCDIO   : X
LIBDC1394 : X  @tab
OpenAL    : X
OpenGL    :  @tab X
OSS       : X  @tab X
PulseAudio : X  @tab X
SDL       :  @tab X
Video4Linux2 : X  @tab X
VfW capture : X  @tab
X11 grabbing : X  @tab
Win32 grabbing : X  @tab

```

"X" means that input/output is supported.

Timecode

```

Codec/format : Read @tab Write
AVI          : X  @tab X
DV           : X  @tab X
GXF          : X  @tab X
MOV          : X  @tab X
MPEG1/2      : X  @tab X
MXF          : X  @tab X

```

SEE ALSO

ffmpeg(1), **ffplay(1)**, **ffprobe(1)**, **ffmpeg-utils(1)**, **ffmpeg-scaler(1)**, **ffmpeg-resampler(1)**, **ffmpeg-codecs(1)**, **ffmpeg-bitstream-filters(1)**, **ffmpeg-formats(1)**, **ffmpeg-devices(1)**, **ffmpeg-protocols(1)**, **ffmpeg-filters(1)**

AUTHORS

The FFmpeg developers.

For details about the authorship, see the Git history of the project (<https://git.ffmpeg.org/ffmpeg>), e.g. by typing the command **git log** in the FFmpeg source directory, or browsing the online repository at <https://git.ffmpeg.org/ffmpeg>.

Maintainers for the specific components are listed in the file *MAINTAINERS* in the source code tree.