NAME

transcode - LINUX video stream processing tool

SYNOPSIS

transcode [-i name] [-H n] [-p file] [-x vmod[,amod]] [-a a[,v]] [--dvd_access_delay N] [-e r[,b[,c]]] [-E r[,b[,c]]] [-n 0xnn] [-N 0xnn] [-b b[,v[,q[,m]]]] [--no_audio_adjust] [--no_bitreservoir] [--lame_preset name[,fast]] [-g wxh] [--import_asr C] [--export_asr C] [--export_par N,D] [--keep_asr] [-f rate[,frc]] [--export_fps f[,c]] [--export_frc F] [--hard_fps] [-o file] [-m file] [-y vmod[,amod]] [-F codec] [--avi_limit N] [--avi_comments F] [-d] [-s g[,c[,f[,r]]]] [-u m[,n]] [-A] [-V] [--uyvy] [-J f1[,f2[,...]]] [-P flag] [-D num] [--av_fine_ms t] [-M mode] [-O] [-r n[,m]] [-B n[,m[,M]]] [-X n[,m[,M]]] [-Z wxh[,fast]] [--zoom_filter str] [-C mode] [--antialias_para w,b] [-I mode] [-K] [-G val] [-z] [-l] [-k] [-j t[,l[,b[,r]]]] [-Y t[,l[,b[,r]]]] [--pre_clip t[,l[,b[,r]]]] [--post_clip t[,l[,b[,r]]]] [-w b[,k[,c]]] [--video_max_bitrate] [-R n[,f1[,f2]]] [-Q n[,m]] [--divx_quant min,max] [--divx_rc p,rp,rr] [--divx_vbv_prof N] [--divx_vbv br,sz,oc] [-c f1-f2[,f3-f4]] [-t n,base] [--dir_mode base] [--frame_interval N] [-U base] [-T t[,c[-d][,a]]] [-W n,m[,file]] [--cluster_percentage use] [--cluster_chunks a-b] [-S unit[,s1-s2]] [-L n] [--import_v4l n[,id]] [--pulldown] [--encode_fields] [--nav_seek file] [--psu_mode] [--psu_chunks a-b] [--no_split] [--ts_pid 0xnn] [--a52_drc_off] [--a52_demux] [--a52_dolby_off] [--print_status N[,r]] [--progress_off] [--color N] [--write_pid file] [--nice N] [--accel type] [--socket file] [--dv_yuy2_mode] [--config_dir dir] [--ext vid,aud] [--export_prof S] [-q level] [-h] [-v]

COPYRIGHT

transcode is Copyright (C) 2001-2003 by Thomas Östreich, 2003-2004 Tilmann Bitterberg.

DESCRIPTION

transcode is a linux text-console utility for video stream processing, running on a platform that supports shared libraries and threads.

Decoding and encoding is done by loading modules that are responsible for feeding transcode with raw video/audio streams (import modules) and encoding the frames (export modules).

It supports elementary video and audio frame transformations, including de-interlacing or fast resizing of video frames and loading of external filters.

A number of modules are included to enable import of DVDs on-the-fly, MPEG elementary (ES) or program streams (VOB), MPEG video, Digital Video (DV), YUV4MPEG streams, NuppelVideo file format, AVI based codecs and raw or compressed (pass-through) video frames and export modules for writing DivX;-), XviD, DivX 4.xx/5.xx or uncompressed AVI and raw files with MPEG, AC3 (pass-through) or PCM audio.

Additional export modules to write single frames (PPM) or YUV4MPEG streams are available, as well as an interface import module to the avifile library.

It's modular concept is intended to provide flexibility and easy user extensibility to include other video/audio codecs or filetypes. A set of tools is included to demux (tcdemux), extract (tcextract) and decode (tcdecode) the sources into raw video/audio streams for import, probing (tcprobe) and scanning (tcscan) your sources and to enable post-processing of AVI files, fixing AVI file header information (avifix), merging multiple files (avimerge), splitting large AVI files (avisplit) to fit on a CD and avisync to correct AV-offsyncs.

OPTIONS

-A

use AC3 as internal audio codec [off].
Only pass-through supported.

-B *n*[*,m*[*,M*]]

resize to height-*n***M* rows [,width-*m***M*] columns [off,32]. *M* must be one of 8, 16 or 32. It makes no difference which M you use. You might look at the *fast* flag of the **-Z** option if you don not want to calculate *n* and *m* yourself.

-C *mode*

enable anti-aliasing mode (1-3) [off].

1 de-interlace effects only

2 resize effects only

3 process full frame (slow)

-D *num*

sync video start with audio frame num [0].

-E *r*[*,b*[*,c*]]

audio output samplerate [Hz], bits per sample and channels [as input]. The option "-J resample" must be provided for export modules not capable of re-sampling.

-F *codec_string*

encoder parameter strings [module dependent].
The -F parameter has different meanings for different export modules.

-y *af6*:
The codec String for the codec you want to encode to. Example values are -F "Uncompressed I420" or -F "OpenDivX 3.11 compatible decoder". To get a list of valid codecs, use -F invalid.

-y *ffmpeg*:
The codec String for the codec you want to encode. Example values are -F mpeg4 or -F mpeg1video or -F huffyuv. To get a list of valid codecs, use -F list

-y *im*:
The name of the image format you want to export. Example values are '-F jpg' (default) or '-F png' or -F 'gif'. With -F jpg, -w gives quality in percent. With -F png, the first digit of -w gives compression level, the second one gives quality, so -w 95 selects best compression level (9) and 5 means PNG adaptive filtering.

-y *jpg*:
The quality of the JPEG encode in percent. Example values are '-F 75' (default) or '-F 100'

-y *mov*:
The name of the quicktime codec. Example values are '-F mjpa' (default) '-F cvid'

-y *mpeg2enc*:
Syntax: -F "<base-profile>[,<options_string>]"
<base-profile> can be one of
'mpeg1' = Generic MPEG1 (default)
'vcd' = Standard VCD
<options_string> will be passed down to mp1e untouched by transcode. Have a look at the manpage for mp1e.

-y *mpeg*:
Syntax: -F "<base-profile>[,<resizer-mode>[,user-profile]]"
<base-profile> can be one of
'1' = MPEG 1 (default)
'b' = big MPEG 1 (experimental)
'v' = VCD
's' = SVCD
'2' = MPEG2
'd' = DVD compliant
<resizer-mode> can be one of
0 = disable resizer (default)
1 = 352x288
2 = 480x480
3 = 480x576
4 = 352x240
<user-profile> is a filename of the profile file. You can either specify the absolute path to the file or if you transcode will look for it in the directory where you started transcode.

-y *mpeg2enc*:
Syntax: -F "<base-profile>[,<options_string>]"
<base-profile> can be one of
'0' = Generic MPEG1 (default)
'1' = Standard VCD
'2' = User VCD
'3' = Generic MPEG2
'4' = Standard SVCD
'5' = User SVCD
'6' = Manual parameter mode
'8' = DVD
<options_string> will be passed down to mpeg2enc untouched by transcode. Have a look at the manpage for mpeg2enc.

-G *val*

Gamma correction (0.0-10.0) [off].
A value of 1.0 does not change anything. A value lower than 1.0 will make the picture "brighter", a value above 1.0 will make it "darker".

-H *n*

auto-probe *n* MB of source (0=disable) default [1]
Use a higher value than the default to detect all subtitles in the VOB.

-I *mode*

enable de-interlacing mode (1-5) [off].

1
: "interpolate scanlines"
linear interpolation (takes the average of the surronding even rows to determine the odd rows), and copies the even rows as is.

2
: "handled by encoder"
tells the encoding code to handle the fact that the frames are interlaced. Most codecs do not handle this.

3
: "zoom to full frame"
drops to to half size, then zooms out. This can cause excessive blurring which is not always unwanted. On the other hand results are quite good.

4
: "drop field / half height"
drop every other field and keep half height.

5
: "interpolate scanlines / blend frames"
linear blend (similar to -vop pp=lb in mplayer) this, like linear blend calculates the odd rows as the average of the surrounding even rows, and also calculates the even rows as an average of the original even rows and also calculates the even rows as an average of the original odd rows and averages the calculated and original rows. Something like avg (avg (row1,row3), avg(row2, row4))

-J *filter1*[*,filter2*[*,...*]]

apply external filter plugins [off]. A full-blown transcode installation has the following filter modules:
See the section **FILTERS** for details.
To see what filters are available at your installation of transcode, execute

```
ls -1 `tcmodinfo -p`/filter*.so
```

Example:

```
transcode [...]
  -J 32detect=force_mode=3,normalize,cut="0-100 300-400"
```

Will load the 32detect filter plugin with parameter force_mode=3, the volume normalizer and the cut filter.

Note:

You can specify more than one -J argument. The order of filter arguments specify in which order the filters are applied. Note also, for transcode internally it makes no difference whether you do "-J filter1 -J filter2" or "-J filter1,filter2"

Use 'tcmodinfo -i *NAME*' to get more information about the filter_*NAME*. Not all filters support this but most of them do. Some of the filter plugins have additional documentation in the docs/ directory. The **FILTERS** section documents nearly every filter which might be of use.

-L *n*

seek to VOB stream offset *n*x2kB default [0]
This option is usually calculated automatically when giving --nav_seek and -c.
-K enable black/white by removing colors mode (grayscale) [off].
Please note this does not necessarily lead to a smaller image / better compression.
For YUV mode, this is done by emptying the chroma planes, for RGB mode a weightend grayscale value is computed.

-M *mode*

demuxer PES AV sync modes (0-4) [1].

Overview

The demuxer takes care that the right video frames go together with the right audio frame. This can sometimes be a complex task and transcode tries to aid you as much as possible.
WARNING: It does make a difference if you (the user) specifies a demuxer to use or if transcode resp. tcprobe(1) chooses the one which it thinks is right for your material.
This is done on purpose to avoid mystic side-effects. So think twice, wether you specify a demuxer or let transcode choose one or you might end up with an off-sync result.

0

Pass-through. Do not mess with the stream, switch off any synchronization/demuxing process.

1

PTS only (default) Synchronize video and audio by inspecting PTS/DTS time stamps of audio and video. Preferred mode for PAL VOB streams and DVDs.

2

NTSC VOB stream synchronization feature. This mode generates synchronization information for transcode by analyzing the frame display time.

3

(like -M 1): sync AV at initial PTS, but invokes "-D/--av_fine_ms" options internally based on "tcprobe" PTS analysis. PTS stands for Presentation Time Stamp.

4

(like -M 2): initial PTS / enforce frame rate, with additional frame rate enforcement (for NTSC).

-N *0xNN*

export audio format id [0x55].

Available format IDs are:

0x1	PCM uncompressed audio
0x50	MPEG layer-2 aka MP2
0x55	MPEG layer-3 aka MP3. Also have a look at --lame_preset if you intend to do VBR audio.
0x2000	AC3 audio
0xfffe	OGG/Vorbis audio

-O

flush lame mp3 buffer on encoder stop [off].

-P *flag*

pass-through flag (0=off|1=V|2=A|3=A+V) [0].
Pass-through for *flag* != 1 is broken and not a trivial thing to fix.
You can pass-through DV video, AVI files and MPEG2 video. When doing MPEG2 pass-through (together with the -y raw module), you can give a requantization factor by using -w -- for example -w 1.5 -- this will make the MPEG2 stream smaller.
The pass-through mode is useful for reconstruction of a broken index of an AVI file. The -x ffmpeg import-module analyzes the compressed bitstream and can detect a keyframe for DIV3, MPEG4 (DivX, XviD, ..) and other formats. It then sets an internal flag which the export module will respect when writing the frame out.

-Q *n*[*,m*]

encoding[,decoding] quality (0=fastest-5=best) [5,5].

-R *n*[*,f1*[*,f2*]]

enable multi-pass encoding (0-3) [0,divx4.log,pcm.log].

0 Constant bitrate (CBR) encoding. [default]

The codec tries to achieve constant bitrate output. This means, each encoded frame is mostly the same size. This type of encoding can help in maintaining constant filling of hardware buffer on set top players or smooth streaming over networks. By the way, Constant bitrate is often obtained sacrifying quality during high motion scenes.

1 Variable bitrate encoding: First pass.

In this mode, the codec analyses the complete sequence in order to collect data that can improve the distribution of bits in a second VBR pass. The collected data is written to second sub argument (default: divx4.log). This data is codec dependant and cannot be used across codecs. It is strongly advised to use the same codec settings for the VBR analysis pass and the VBR encoding pass if you want predictable results.
The video output of the first pass is not of much use and can grow very large. It's a good idea to not save the video output to a file but directly to /dev/null. Usually the bitrate is ignored during first pass.
Disabling audio export makes sense too, so use -y codec,null. It is **not** recommended to disable the audio **import** because transcode might drop video frames to keep audio and video in sync. This is not possible when the audio import is disabled. It may lead to the fact that the codec will see a different sequence of frames which effectively renders the log file invalid.

2 Variable bitrate encoding: Second pass.

The first pass allowed the codec collecting data about the complete sequence. During the second pass, the codec will use that data in order to find an efficient bit distribution that respects both the desired bitrate and the natural bitrate curve shape. This ensures a good compromise between quality and desired bitrate.
Make sure you activate both sound and video encoding during this pass.

3 Constant quantizer encoding.

The quantizer is the "compression level" of the picture. The lower the quantizer is, the higher is the quality of the picture. This mode can help in making sure the sequence is encoded at constant quality, but no prediction can be made on the final bitrate. When using this mode, the **-w** option changes its meaning, it now takes the quantizer ranging from 1 to 31.

-S *unit*[*,s1-s2*]

process program stream unit[,s1-s2] sequences [0,all].
This option is a bit redundant to --psu*. This option lets you specify which units you want to have decoded or skipped. At a programm stream unit boundary, all (internal) mpeg timers are reset to 0. tcprobe will tell you how many units are in one file.

-T *t*[*,c*[*,a*]]

select DVD title[,chapter[,angle]] [1,1,1]. Only a single chapter is transcoded. Use -T 1,-1 to trancode all chapters in a row. You can even specify chapter ranges.

-U *base*

process DVD in chapter mode to base-ch%02d.avi [off].

-V

use YV12/I420 as internal video codec [off].
This is usually much faster than RGB processing but some import modules may not support this format. Always use this option when possible.

-W *n*,*m*[,*nav_file*]

autosplit and process part *n* of *m* (VOB only) [off]

-X *n*[,*m*,[*M*]]

resize to height+*n***M* rows [,width+*m***M*] columns [off,32]. M must be one of 8, 16 or 32. It makes no difference which M you use. You might look at the *fast* flag of the **-Z** option if you do not want to calculate *n* and *m* yourself.

-Y *top*[*,left*[*,bottom*[*,right*]]]

select (encoder) frame region by clipping border. Negative values add a border [off].

-Z *widthxheight*[*,fast*]

resize to *width* columns, *height* rows with filtering [off,notfast].
If *fast* is given, transcode will calculate the parameters for **-X** and/or **-B**. The option *fast* can only be used when the import and export geometry of an image is a multiple of 8.

In fast mode, a faster but less precise resizing algorithm will be used resulting in a slightly less good quality. Beside this (small) drawback, it is worth a try.

It is also possible to omit *width* OR *height*. In this case, transcode will calculate the missing value using the import aspect ratio. The new value will be aligned to be a multiple of 8. Using an additional *fast* is also possible.

Examples (assume input is a 16:9 coded file at 720x576):

```
-Z 576x328       uses filtered zoom.
-Z 576x328,fast  uses fast zoom.
-Z 576x,fast     guess 328 and do fast zoom.
-Z x328          guess 576 and do filtered zoom.
```

If you also set --export_prof, you can use just "fast" to indicate that fast resizing is wanted.

-a *ach*[*,vch*]
extract audio[,video] track for encoding.

-b *b*[,*v*,[*q*,[*m*]]]
audio encoder bitrate kBits/s[,vbr[,quality[,mode]]] [128,0,5,0]

The *mode* parameter specifies which modus lame should use for encoding. Available modes are:

0	Joint Stereo (default)
1	Full stereo
2	Mono

-c *f1-f2*[,*f3-f4*[, ...]]
encode only frames *f1-f2* [and *f3-f4*]. Default is to encode all available frames. Use this and you'll get statistics about remaining encoding time. The *f[N]* parameters may also be timecodes in the HH:MM:SS.FRAME format. Example:

```
-c 500-0:5:01,:10:20-1:18:02.1
```

Will encode only from frame 500 to 5 minutes and 1 second and from 10 min, 20 sec to 1 hour, 18 min, 2 sec and one frame.
Note that transcode starts counting frames at *0* and excludes the last frame specified. That means that "-c 0-100" will encoded 100 frames starting at frame *0* up to frame *99*

-d
swap bytes in audio stream [off].
In most cases, DVD PCM audio tracks require swapping of audio bytes

-e *r*[*,b*[*,c*]]
PCM audio stream parameter. Sample rate [Hz], bits per sample and number of channels [48000,16,2]. Normally this is autodetected.

-f *rate*[,*frc*]

import video frame rate[,frc] [25.000,0]. If *frc* (frame rate code) is specified, transcode will calculate the precise frames per second internally. Valid values for *frc* are

frc	rate
1	23.976 (24000/1001.0)
2	24
3	25
4	29.970 (30000/1001.0)
5	30
6	50
7	59.940 (2 * 29.970)
8	60
9	1
10	5
11	10
12	12
13	15

-g *Width*x*Height*

video stream frame size [720x576].

-h

print out usage information.

-i *name*

input file/directory/device/mountpoint/host name, default is [/dev/zero].

-j *top*[*,left*[*,bottom*[*,right*]]]

select frame region by clipping border. Negative values add a border [off].

-k

swap red/blue (Cb/Cr) in video frame [off]. Use if people have blue faces.

-l

mirror video frame [off].

-m *file*

write audio stream to separate file [off].

-n *0xnn*

import audio format id [0x2000]. Normally, this is autodetected.

-o *file*

output file name, default is [/dev/null].

-p *file*

read audio stream from separate file [off].

-q *debuglevel*

QUIET	0
INFO	1
DEBUG	2
STATS	4
WATCH	8
FLIST	16
VIDCORE	32
SYNC	64
COUNTER	128
PRIVATE	256

-r *n*[,*m*]

reduce video height/width by n[,m] [off]. Example: -r *2* will rescale the framesize of a 720x576 file to 360x288.

-s *gain*,[*center*,[*front*,[*rear*]]]

increase volume of audio stream by gain,[center,front,rear] default [off,1,1,1].

-t *n*,*base*

split output to *base*%03d.avi with *n* frames [off].

-u *m*[*,n*]

use *m* framebuffer[,*n* threads] for AV processing [10,1].

-v

print version.

-w *b*[*,k*[*,c*]]

encoder bitrate[,keyframes[,crispness]] [(6000 for MPEG 1/2, 1800 for others),250,100].

--video_max_bitrate *b*

Use *b* as maximal bitrate when encoding variable bitrate MPEG-2 streams

-x *vmod*[*,amod*]

video[,audio] import modules [auto,auto]. If omitted, transcode will probe for appropriate import modules. A full-blown transcode installation has the following import modules:
Module "ac3": (audio) AC3
Module "af6": (video) Win32 dll | (audio) PCM
Module "avi": (video) * | (audio) *
Module "divx": (video) DivX;-)/XviD/OpenDivX/DivX 4.xx/5.xx
Module "dv": (video) DV | (audio) PCM
Module "dvd": (video) DVD | (audio) MPEG/AC3/PCM
Module "ffmpeg": (video) FFMPEG API (build 4631) | MS MPEG4v1-3/MPEG4/MJPEG
Module "im": (video) RGB

Module "imlist": (video) RGB
Module "lav": (video) LAV | (audio) WAVE
Module "lzo": (video)
Module "mjpeg": (video) MJPEG
Module "mov": (video) * | (audio) *
Module "mp3": (audio) MPEG
Module "mpeg2": (video) MPEG2
Module "mpeg3": (video) MPEG2
Module "mplayer": (video) rendered by mplayer | (audio) rendered by mplayer
Module "null": (video) null | (audio) null
Module "nvrec": (video) nvrec - v4l | (audio) nvrec - dsp
Module "ogg": (video) * | (audio) Ogg Vorbis
Module "raw": (video) RGB/YUV | (audio) PCM
Module "rawlist": (video) YUV/RGB raw frames
Module "vdrac3": (audio) VDR-AC3
Module "vob": (video) MPEG-2 | (audio) MPEG/AC3/PCM | (subtitle)
Module "v4l": (video) YUV/RGB | (audio) PCM
Module "xml": (video) * | (audio) *
Module "xvid": (video) XviD/OpenDivX/DivX 4.xx/5.xx
Module "yuv4mpeg": (video) YUV4MPEGx | (audio) WAVE
To see what your transcode has, do a

```
ls -1 `tcmodinfo -p`/import*.so
```

It is possible to pass option strings to import modules like to filter modules. The average user does not this feature and not many modules support it. The syntax is **-x vmod=options,amod=options**
Example

```
  -x rawlist=uyvy,null
```

To tell the rawlist import module (which reads images from a list of files) the colour space of the images.

-y *vmod*[*,amod*]
video[,audio] export modules [null]. If omitted, transcode will encode to the *null* module. A full-blown transcode installation has the following export modules:

ac3 - (video) null | (audio) ac3

This module has no compile-time dependencies. At run-time ffmpeg must be present. Support for this module is good.
Lets you encode audio (raw PCM) to AC3 via the ffmpeg binary.

af6 - (video) Win32 dll | (audio) MPEG/AC3/PCM

At compile-time libavifile must be available. At run-time libavifile and win32codecs must be present. Support for this module is good.
Interface to the avifile library which allows the use of win32codec on the linux-x86 platform.

debugppm - (video) debugPPM/PGM | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. This module has no run-time dependencies. Support for this module is good.

Especially usefull when one want to try to find bugs in YUV mode. It encodes the three planes as seperate images. In RGB mode, the 3 color planes get encoded seperatly.

divx4 - (video) DivX 4.xx | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time divx4linux (old) and libdivxencore.so must be present. Support for this module is fair.

Encodes MPEG4 video using the closed-source binaries from divx.com to an AVI container.

divx4raw - (video) DivX 4.xx (ES) | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time divx4linux (old) and libdivxencore.so must be present. Support for this module is fair.

Encodes MPEG4 video using the closed-source binaries from divx.com into no file container at all. It writes out the raw bitstream.

divx5 - (video) DivX 5.xx | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time divx4linux (new) and libdivxencore.so must be present. Support for this module is good.

Encodes MPEG4 video using the closed-source binaries from divx.com to an AVI container.

divx5raw - (video) DivX 5.xx (ES) | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time divx4linux (new) and libdivxencore.so must be present. Support for this module is fair.

Encodes MPEG4 video using the closed-source binaries from divx.com into no file container at all. It writes out the raw bitstream.

dv - (video) Digital Video | (audio) MPEG/AC3/PCM

At compile-time libdv and libmp3lame must be available. At run-time libdv must be present. Support for this module is good.

Encodes DV into an AVI container. DV is a codec developed by Sony and is often used in digital camcorders.

dvraw - (video) Digital Video | (audio) PCM

At compile-time libdv must be available. At run-time libdv must be present. Support for this module is good.

Encodes DV into a DV file. DV is a codec developed by Sony and is often used in digital camcorders. A raw DV file can be played back into the camcorder.

fame - (video) MPEG-4 | (audio) MPEG/AC3/PCM

At compile-time libfame must be available. This module has no run-time dependencies. Support for this module is poor.

Fame is yet another MPEG4 encoder. It encodes to a raw file.

ffmpeg - (video) * | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. This module has no run-time dependencies. Support for this module is good.

Encodes many different formats to both AVI and raw. Supported are mpeg1video, mpeg2video, mpeg4, mjpeg, h263, h263p, wmv1, wmv2, rv10, msmpeg4, msmpeg4v2, huffyuv and dvvideo.

im - (video) * | (audio) MPEG/AC3/PCM

At compile-time libImageMagick must be available. This module has no run-time dependencies. Support for this module is good.

Encodes image sequences by using the ImageMagick library. ImageMagick is able to handle a lot of different image formats among are png, jpg, miff, tiff, etc. Use -F to select the desired format.

jpg - (video) * | (audio) MPEG/AC3/PCM

At compile-time libjpeg must be available. This module has no run-time dependencies. Support for this module is good.

Encodes jpg image sequences using libjpeg. Faster than ImageMagick. Use -F to select the compression quality.

lame - (audio) MPEG 1/2

At compile-time libmp3lame must be available. At run-time lame and sox must be present. Support for this module is good.

An audio-only encoder which drives the lame binary. The tool sox is used to do resampling if required. Encodes to a MP3 file.

lzo - (video) LZO real-time compression | (audio) MPEG/AC3/PCM

At compile-time liblzo and libmp3lame must be available. This module has no run-time dependencies. Support for this module is good.

Encodes video using a loss-less real-time LZO codec. This codec is a homegrown invention of transcode and is intended as an intermediate storage format. MPlayer can playback LZO-based AVI files as well.

mjpeg - (video) Motion JPEG | (audio) MPEG/AC3/PCM

At compile-time libmp3lame and libjpeg must be available. This module has no run-time dependencies. Support for this module is poor.

Encodes MJPEG based AVI files using a homegrown algorithm based on libjpeg. Using ffmpeg -F mjpeg for this task is a good idea.

mov - (video) * | (audio) *

At compile-time libquicktime must be available. At run-time libquicktime must be present. Support for this module is fair.

Interface to the quicktime library.

mp1e - (video) MPEG1 video | (audio) MPEG1-Layer

This module has no compile-time dependencies. At run-time mp1e must be present. Support for this module is good.

Drives the mp1e binary and writes an mpeg1 file to disc. It can also encode vcd compliant streams. Note: it writes an intermediate wav file for audio due to a limitation of mp1e.

mp2enc - (audio) MPEG 1/2

At compile-time mjpegtools must be available. At run-time mp2enc must be present. Support for this module is good.

Drives the mp2enc binary and writes an MP2 (MPEG1-Layer2) file. Useful for when encoding to SVCD to be multiplexed with mplex after encoding.

mpeg - (video) MPEG 1/2 | (audio) MPEG 1 Layer II

At compile-time nasm must be available. This module has no run-time dependencies. Support for this module is good.

Interface to the bbmpeg library (included in transcode). It can encode generic mpeg1, VCD, SVCD, MPEG2 and DVD type video. Encoded video goes into a elementary file to be multiplexed with the corresponding audio file after encoding.

mpeg2enc - (video) MPEG 1/2

At compile-time mjpegtools must be available. At run-time mpeg2enc must be present. Support for this module is good.

Drives the mpeg2enc binary. mpeg2enc is a very feature rich MPEG encoder, have a look at its manpage. Encodes generic mpeg1, VCD, SVCD, MPEG2 and DVD type video.

net - (video) RGB/YUV | (audio) PCM/AC3

At compile-time net-support must be available. This module has no run-time dependencies. Support for this module is good.

Transfers video data over a network between various transcodes. Useful if you have multiple systems connected via a fast network.

null - (video) null | (audio) null

This module has no compile-time dependencies. This module has no run-time dependencies. Support for this module is good.

Data sink. Does nothing else than discarding data.

ogg - (video) null | (audio) ogg

This module has no compile-time dependencies. At run-time oggenc must be present. Support for this module is good.

Drives the oggenc binary. Encodes an Ogg/Vorbis file. Resamples.

pcm - (audio) PCM (non-interleaved)

This module has no compile-time dependencies. This module has no run-time dependencies. Support for this module is good.

Writes each audio channel to a WAVE PCM file.

ppm - (video) PPM/PGM | (audio) MPEG/AC3/PCM

This module has no compile-time dependencies. This module has no run-time dependencies. Support for this module is good.

Writes an image sequence of PGM or PPM files. PPM is an old format and there are several tools around to manipulate such files.

pvm - (video) * | (audio) *

At compile-time libpvm3 must be available. At run-time pvm must be present. Support for this module is good.

Meta module. It allows transcode to be used in a PVM cluster. See docs/export_pvm.txt

raw - (video) * | (audio) MPEG/AC3/PCM

This module has no compile-time dependencies. This module has no run-time dependencies. Support for this module is good.

Can write uncompressed streams to an AVI file as well as raw mpeg2 files in pass-through mode.

toolame - (audio) MPEG 1/2

This module has no compile-time dependencies. At run-time toolame and sox must be present. Support for this module is good.

Drives the toolame binary to create MP2 audio tracks. Sox is used for resampling if requested.

wav - (audio) WAVE PCM

This module has no compile-time dependencies. This module has no run-time dependencies. Support for this module is good.

Creates WAVE PCM files with interleaved audio for stereo.

xvid2 - (video) XviD 0.9.x (aka API 2.1 series) | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time libxvidencore.so.2 must be present. Support for this module is good.

Encodes MPEG4 video using the library available form xvid.org. Check out the library from xvidcvs using cvs -d : pserver:anonymous@cvs.xvid.org:/xvid co -rrelease-0_9_2 xvidcore. The output can either be an AVI file or a MPEG4 elementary stream (with -F raw).

xvid3 - (video) XviD nonumber series (aka API 3.0) | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time libxvidcore.so.3 must be present. Support for this module is good.

Encodes MPEG4 video using the library available form xvid.org. Check out the library from xvidcvs using cvs -d : pserver:anonymous@cvs.xvid.org:/xvid co -rHEAD xvidcore. The output can either be an AVI file or a MPEG4 elementary stream (with -F raw).

xvid4 - (video) XviD 1.0.x series (aka API 4.0) | (audio) MPEG/AC3/PCM

At compile-time libmp3lame must be available. At run-time libxvidcore.so.4 must be present. Support for this module is good.

Encodes MPEG4 video using the library available form xvid.org. Check out the library from xvidcvs using cvs -d : pserver:anonymous@cvs.xvid.org:/xvid co -rdev-api-4 xvidcore. There is also a tool available to create configuration files for this xvid version at http://zebra.fh-weingarten.de/transcode/xvid4conf. The output can either be an AVI file or a MPEG4 elementary stream (with -F raw).

yuv4mpeg - (video) YUV4MPEG2 | (audio) MPEG/AC3/PCM

At compile-time mjpegtools must be available. This module has no run-time dependencies. Support for this module is good.

Writes the uncompressed raw YUV data in a YUV4MPEG format as used by the lav* and mjpeg* tools.

To see what your transcode has, do a

```
ls -1 `tcmodinfo -p`/export*.so
```

It is possible to pass option strings to export modules like to filter modules. The syntax is

```
-y vmod=options,amod=options
```

-z

flip video frame upside down [off].

--accel *type*

enforce experimental IA32 acceleration for type [autodetect]. *type* may be one of

C	No acceleration
ia32asm	plain x86 assembly
mmx *3dnow* *sse* *sse2*	Acceleration for a specific SIMD extension

--avi_limit *N*

split/rotate output AVI file after N MB [2048].

--avi_comments *F*

Read AVI header comments from file *F* [off].
The AVI file format supports so-called tomb-stone data. It can be used to write annotations into the AVI file.
See the file **docs/avi_comments.txt** for a sample input file with all tags. When the file is read, empty lines and lines starting with '#' are ignored. The syntax is: "TAG<space>STRING". The order of the tags does not matter. If a tag has no string following it, it is ignored. That means, you can use the file docs/avi_comments.txt as input and only fill out the fields you want.

A very simple example is:

```
----------------snip----------------
INAM My 1st Birthday
ISBJ My first steps!
IART My proud family
----------------snip----------------
```

Keep in mind that there is no endless space in the AVI header, most likely its around 1000 bytes.

--zoom_filter *string*

use filter string for video resampling -Z [Lanczos3]
The following filters are available:
- Bell
- Box
- Lanczos3 (default)
- Mitchell
- Hermite
- B_spline
- Triangle

--cluster_percentage
use percentage mode for cluster encoding -W [off]

--cluster_chunks *a-b*
process chunk range instead of selected chunk [off]

--export_asr *C*
set export aspect ratio code *C* [as input]

Valid codes for *C* are

1	1:1
2	4:3
3	16:9
4	2.21:1

--export_prof *S*
Select an export profile {vcd, svcd, dvd} [none]
If you set this meta option to one of the values below, transcode will adjust some internal paramaters as well as geometry and clipping. If no export modules are specified, mpeg2enc for video and mp2enc for audio are used when compiled with mjpegtools support.

Valid values for *S* are vcd, vcd-pal, vcd-ntsc, svcd, svcd-pal, svcd-ntsc, dvd, dvd-pal and dvd-ntsc.

When one of the above is used, transcode will calculate the needed clipping and resizing values for you based on the import and export aspect ratio. This is especially handy if you want to encode a 16:9 DVD into a 4:3 SVCD for example. Transcode internally then sets --pre_clip to add the black bars ("letterboxing").

If you use "vcd" instead of "vcd-pal" or "vcd-ntsc", transcode will make an educated guess if PAL or NTSC vcd is wanted. The same is true for "svcd" and "dvd". When the input file has no aspect ratio information at all, transcode guesses it based on the import frame sizes. You can set the import aspect ratio by giving --import_asr CODE.

Examples (assume input is a 16:9 coded file at 720x576 (PAL)):

```
--export_prof svcd      clip top/bot -96; resizes to 480x576
--export_prof vcd-ntsc  clip top/bot -96; resizes to 352x240
```

This enables simpilified commandlines where transcode tries to set sensible values for mpeg export.

```
transcode -i vob/ -V --export_prof svcd -Z fast -o test
```

--export_par *C*[,*N*]

set export pixel aspect ratio to *C*[,*N*]
To encode anamorphic material, transcode can encode the target pixel aspect ratio into the file. This is NOT the actual aspect ratio of the image, but only the amount by which every single pixel is stretched when played with an aspect ratio aware application, like mplayer.

To encode at non standard aspect ratios, set both *C* and *N* E.g. to make every pixel twice as high as it's wide (e.g. to scale back to normal size after deinterlacing by dropping every second line).

If *C* and *N* are specified, the value set for *C* does give the pixel aspect ratio of the width and *N* the one for the height If only *C* is specified, the table below applies.

Valid codes for *C* are

1	1:1	No stretching
2	12:11	5:4 image to 4:3 (ex: 720x576 -> 768x576)
3	10:11	3:2 image to 4:3 (ex: 720x480 -> 640x480)
4	16:11	5:4 image to 16:9 (ex: 720x576 -> 1024x576)
5	40:33	3:2 image to 16:9 (ex: 720x480 -> 872x480)

--import_asr *C*

set import aspect ratio code *C* [autoprobed]

Valid codes for *C* are

1	1:1
2	4:3
3	16:9
4	2.21:1

--ext *vid*,*aud*

Use these file extensions [.avi,.mp3]
When this option is not given, transcode will use a file extension dependend on the export module. For the mpeg export modules this is ".m2v" resp ".m1v" and ".mpa" for audio. To clean up this mess, the option --ext was introduced without breaking exising behaviour. Use **--ext** *none*,*none* to disable filename extension.

--keep_asr

try to keep aspect ratio (only with -Z) [off]

The **--keep_asr** options changes the meaning of **-Z**. It tries to fit the video into a framesize specified by **-Z** by keeping the *original* aspect ratio.

```
+----------------+                            +---480-----+
|                |                            | black     |
|720x306 = 2.35:1| -> -Z 480x480 --keep_asr ->|-----------4
|                |                            | 480x204   8
+----------------+                            |-----------0
                                              | black     |
                                              +-----------+
```

Consider **--keep_asr** a wrapper for calculating **-Y** options and **-Z** options

--divx_quant *min,max*

divx encoder min/max quantizer [2,31]

--divx_rc *p*,*rp*,*rr*

divx encoder rate control parameter [2000,10,20]

--divx_vbv_prof *N*

divx5 encoder VBV profile (0=free-5=hiqhq) [3]

Sets a predefined profile for the Video Bitrate Verifier. If *N* is set to zero, no profile is applied and the user specified values from **--divx_vbv** are used.

Valid profiles

0 Free/No profile	(			Use supplied values	)
1 Handheld	(	320,	16,	3072	)
2 Portable	(	1920,	64,	12288	)
3 Home Theatre	(	10000,	192,	36864	)
4 High Definition	(	20000,	384,	73728	)

--divx_vbv *br*,*sz*,*oc*

divx5 encoder VBV params (bitrate,size,occup.) [10000,192,36864]

These parameters are normally set through the profile parameter **--divx_vbv_prof**. Do not mess with theses value unless you are absolutely sure of what you are doing. Transcode internally multiplicates vbv_bitrate with 400, vbv_size with 16384 and vbv_occupancy with 64 to ensure the supplied values are multiples of what the encoder library wants.

--export_fps *rate*[,*frc*]

set export frame rate (and code) [as input].Valid values for *frc* are

frc	rate	
1	23.976	(24000/1001.0)
2	24	
3	25	
4	29.970	(30000/1001.0)
5	30	
6	50	
7	59.940	(2 * 29.970)
8	60	
9	1	
10	5	
11	10	
12	12	
13	15	

--export_frc *F*

set export frame rate code *F* [as input]
obsolete, use --export_fps 0,F

--hard_fps

disable smooth dropping (for variable fps clips) [off]
see /docs/framerate.txt for more information.

--uyvy

use UYVY (4:2:2) as internal video codec [off]
This is an experimental feature and a developers playground. Not many import, export and filter modules support this colorspace. A 4:2:2 colorspace offers much more quality than the consumer oriented 4:2:0 colorspaces like YV12/I420. Most equipment in film and video post-production uses UYVY. UYVY doubles the precision for chroma (color difference) information in the image.
All internal transformations are supported in UYVY mode (clipping, flipping, zooming, etc).

--import_v4l *n*[,*id*]

channel number and station number or name [0]

--record_v4l *a-b*

recording time interval in seconds [off]
obsolete, use -c a-b.

--duration *hh:mm:ss*

limit v4l recording to this duration [off]
obsolete, use -c 0-hh:mm:ss.

--pulldown

set MPEG 3:2 pulldown flags on export [off]

--antialias_para *w,b*

center pixel weight, xy-bias [0.333,0.500]

--no_audio_adjust

disable internal audio frame sample adjustment [off]

--no_bitreservoir

disable lame bitreservoir for MP3 encoding [off]

--config_dir *dir*

Assume config files are in this *dir*
This only affects the -y ffmpeg and all -y xvid234 modules. It gives the path where the modules search for their configuration.

--lame_preset *name*[*,fast*]

use lame preset with *name*. [off]
Lame features some built-in presets. Those presets are designed to provide the highest possible quality. They have for the most part been subject to and tuned via rigorous listening tests to verify and achieve this objective. These are continually updated to coincide with the latest developments that occur and as a result should provide you with nearly the best quality currently possible from LAME. Any of those VBR presets can also be used in fast mode, using the new vbr algorithm. This mode is faster, but its quality could be a little lower. To enable the fast mode, append "*,fast*"

<N kbps>
Using this preset will usually give you good quality at a specified bitrate. Depending on the bitrate entered, this preset will determine the optimal settings for that particular situation. While this approach works, it is not nearly as flexible as VBR, and usually will not reach the same quality level as VBR at higher bitrates. ABR.

medium
This preset should provide near transparency to most people on most music. The resulting bitrate should be in the 150-180kbps range, according to music complexity. VBR.

standard
This preset should generally be transparent to most people on most music and is already quite high in quality. The resulting bitrate should be in the 170-210kbps range, according to music complexity. VBR.

extreme
If you have extremely good hearing and similar equipment, this preset will provide slightly higher quality than the "standard" mode. The resulting bitrate should be in the 200-240kbps range, according to music complexity. VBR.

insane
This preset will usually be overkill for most people and most situations, but if you must have the absolute highest quality with no regard to filesize, this is the way to go. This preset is the highest preset quality available. 320kbps CBR.

(taken from http://www.mp3dev.org/mp3/doc/html/presets.html)

--av_fine_ms *t*
AV fine-tuning shift *t* in millisecs [autodetect]
also see -D.

--nav_seek *file*
use VOB or AVI navigation file [off].
Generate a nav file with tcdemux -W >nav_log for VOB files or with aviindex(1) for AVI files.

--psu_mode
process VOB in PSU, -o is a filemask incl. %d [off]. PSU means Program Stream Unit and this mode is useful for (mostly) NTSC DVDs which have several program stream units.

--psu_chunks *a-b*
process only selected units *a-b* for PSU mode [all]

--no_split
encode to single file in chapter/psu/directory mode [off]
If you don't give this option, you'll end up with several files in one of the above mentioned modes. It is still possible to merge them with avimerge(1).

--pre_clip t[,l[,b[,r]]]
select initial frame region by clipping border [off]

--post_clip t[,l[,b[,r]]]
select final frame region by clipping border [off]

--a52_drc_off
disable liba52 dynamic range compression [enabled]
If you dont specify this option, liba52 will provide the default behaviour, which is to apply the full dynamic range compression as specified in the A/52 stream. This basically makes the loud sounds softer, and the soft sounds louder, so you can more easily listen to the stream in a noisy environment without disturbing anyone.

If you let it enabled this this will totally disable the dynamic range compression and provide a playback more adapted to a movie theater or a listening room.

--a52_demux
demux AC3/A52 to separate channels [off]

--a52_dolby_off
disable liba52 dolby surround [enabled]
selects whether the output is plain stereo (if the option is set) or if it is Dolby Pro Logic - also called Dolby surround or 3:1 - downmix (if the option is not used).

--dir_mode *base*
process directory contents to base-%03d.avi [off]

--frame_interval *N*

select only every *N*th frame to be exported [1]

--encode_fields *C*

enable field based encoding (if supported) [off]
This option takes an argument if given to denote the order of fields. If the option is not given, it defaults to progressive (do not assume the picture is interlaced)

Valid codes for *C* are:

p
progressive (default)

t
top-field first

b
bottom-field first

--dv_yuy2_mode

decoded Digital Video (raw) YUV frame is in YUY2 (packet) format using libdv. Downsample frame to YV12. PAL users should compile libdv with --with-pal-yuv=YV12 to avoid this option [off]

--write_pid *file*

write pid of signal thread to *file* [off] Enables you to terminate transcode cleanly by sending a SIGINT (2) to the pid in *file*. Please note *file* will be overwritten. Usage example

```
$ transcode ... --write_pid /tmp/transcode.pid &
$ kill -2 `cat /tmp/transcode.pid`
```

--nice *N*

set niceness to *N* [off]
The option --nice which renices transcode to the given positive or negative value. -10 sets a high priority; +10 a low priority. This might be useful for cluster mode.

--progress_off

disable progress meter status line [off]

--color *N*

level of color in transcodes output [1]
Colorful output can be disabled by setting *N* to *0*. It will be automatically disabled if the output of transcode is a file or a pipe.

--print_status *N*[,*usecr*]

print status every *N* frames / use CR or NL [1,1]
The first parameter controls how frequently the status message is printed (every *N* frames), the second parameter (if provided) controls whether transcode ends the line with a CR ('\r') or NL ('\n') character. Transcode defaults to ending with a CR if its output is going to a terminal, or a LF if its output is going to somewhere else (such as a logfile), so most people shouldn't have any need to specify the second parameter since it should do the right thing most of the time.

--socket *FILE*

Open a socket to accept commands while running. See **tcmodinfo(1)** and /docs/filter-socket.txt for more information about the protocol.

--more_help *param*

more help on named parameter (if supported)

FILTERS

The syntax for filter options is simple. A filter is specified with

```
-J filter=optionstring
```

The optionstring can contain multiple options which are separated by colons `:'

```
-J filter=option1:option2:option3
```

An option can have an argument or non (bool). For options with an argument, the format in which the argument has to be given to the option is specified in a printf(1) like string. The most common case is `%d' which simply means a number. The argument has to be seperated from the option by a `='.

```
-J filter=bool1:option1=15:option2=20x30
```

Most filters try to do the right thing with the default options. You should play with various parameters if you are not satisfied with the default behaviour. If you have no idea what a filter does, its very likely that you don't need it.

If a filter takes (for eg.) a filename as an argument, make sure that the filename does not contain a `:' or a `='. Its a limitation of the parser. A comma `,' is possible but must be extra quoted. For the text filter that is

```
-J text=string="Hello\, World"
```

29to23 - **frame rate conversion filter (interpolating 29 to 23)**

29to23 was written by Max Alekseyev, Tilmann Bitterberg. The version documented here is v0.3 (2003-07-18). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

32detect - 3:2 pulldown / interlace detection plugin

32detect was written by Thomas. The version documented here is v0.2.4 (2003-07-22). This is a video filter. It can handle RGB and YUV mode. It supports multiple instances and can run as a pre-processing and/or as a post-processing filter.

- *threshold* = *%d* [default *9*]
  Interlace detection threshold
- *chromathres* = *%d* [default *4*]
  Interlace detection chroma threshold
- *equal* = *%d* [default *10*]
  threshold for equal colors
- *chromaeq* = *%d* [default *5*]
  threshold for equal chroma
- *diff* = *%d* [default *30*]
  threshold for different colors

- *chromadi = %d* [default *15*]
  threshold for different chroma
- *force_mode = %d* [default *0*]
  set internal force de-interlace flag with mode -I N
- *pre = %d* [default *1*]
  run as pre filter
- *verbose* (bool)
  show results
  This filter checks for interlaced video frames. Subsequent de-interlacing with transcode can be enforced with 'force_mode' option

32drop - 3:2 inverse telecine removal plugin

32drop was written by Thomas Oestreich. The version documented here is v0.4 (2003-02-01). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

aclip - generate audio clips from source

aclip was written by Thomas Oestreich. The version documented here is v0.1.1 (2003-09-04). This is a audio filter. It is a pre-processing only filter.

- *level = %d* [default *10*]
  The audio must be under this level to be skipped
- *range = %d* [default *25*]
  Number of samples over level will be keyframes

astat - audio statistics filter plugin

astat was written by Thomas Oestreich. The version documented here is v0.1.3 (2003-09-04). This is a audio filter. It is a pre-processing only filter.

- *file = %s*
  File to save the calculated volume rescale number to

compare - compare with other image to find a pattern

compare was written by Antonio Beamud. The version documented here is v0.1.2 (2003-08-29). This is a video filter. It can handle RGB and YUV mode. It supports multiple instances. It is a post-processing only filter.

- *pattern = %s*
  Pattern image file path
- *results = %s*
  Results file path
- *delta = %f* [default *45.000000*]
  Delta error
  Generate a file in with information about the times, frame, etc the pattern defined in the image parameter is observed.

control - apply a filter control list

control was written by Tilmann Bitterberg. The version documented here is v0.0.1 (2003-11-29). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

* *file* = *%s*
read commands to apply from file.

* *ofile* = *%s*
write output of commands to output file
The format of the command file is framenumber followed by at least one whitespace followed by the command followed by at least one whitespace followed by arguments for the command. Empty lines and lines starting with a `#' are ignored. The frame numbers must be sorted ascending.

```
# Example file
# At frame 10 load the smooth filter
10 load smooth
# reconfigure at 20
20 configure smooth=strength=0.9
99 disable smooth
```

cpaudio - **copy one audio channel to the other channel filter plugin**

cpaudio was written by William H Wittig. The version documented here is v0.1 (2003-04-30). This is a audio filter. It is a post-processing only filter.

* *source* = *%c* [default *l*]
Source channel (l=left, r=right)
Copies audio from one channel to another

cshift - **chroma-lag shifter**

cshift was written by Thomas Östreich, Chad Page. The version documented here is v0.2.1 (2003-01-21). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

* *shift* = *%d* [default *1*]
Shift chroma(color) to the left

cut - **encode only listed frames**

cut was written by Thomas Oestreich. The version documented here is v0.1.0 (2003-05-03). This is a video and audio filter. It is a pre-processing only filter.

* *HH:MM:SS.f-HH:MM:SS.f/step* = *%s*
apply filter [start-end] frames [0-oo/1]

decimate - **NTSC decimation plugin**

decimate was written by Thanassis Tsiodras. The version documented here is v0.4 (2003-04-22). This is a video filter. It can handle YUV mode only. It is a post-processing only filter.

* *verbose* (bool)
print verbose information
see /docs/README.Inverse.Telecine.txt

denoise3d - **High speed 3D Denoiser**

denoise3d was written by Daniel Moreno & A'rpi. The version documented here is v1.0.3 (2003-11-08). This is a video filter. It can handle YUV mode only. It supports multiple instances. It can be used as a pre-processing or as a post-processing filter.

- *luma* = *%f* [default *4.000000*]
  spatial luma strength
- *chroma* = *%f* [default *3.000000*]
  spatial chroma strength
- *luma_strength* = *%f* [default *6.000000*]
  temporal luma strength
- *chroma_strength* = *%f* [default *8.000000*]
  temporal chroma strength
- *pre* = *%d* [default *0*]
  run as a pre filter
  What: The denoise3d filter from mplayer (sibling of hqdn3d). Works very crude and simple but also very fast. In fact it is even faster than the original from mplayer as I managed to tweak some things (a.o. zero frame copying).

  Who: Everyone who wants to have their captured frames thoroughly denoised (i.e. who want to encode to mpeg or mjpeg) but do not have enough processing power to real-time encode AND use hqdn3d (better quality but a lot slower) or dnr (yet slower), not to mention the other denoisers that are even slower. Quality is really good for static scenes (if fed with the right parameters), moving objects may show a little ghost-image (also depends on parameters) though. Your milage may vary.

  How: Parameters are the same as the hqdn3d module, although in practice you'll not end up with exactly the same values. Just experiment. Particular for this version of the filter is that if you supply -1 to either component's parameters (luma/chroma), that component will not have the filter applied to. If you're still short on CPU cycles, try disabling the luma filter, this will not make much difference in the effectiveness of the filter!

detectsilence - **audio silence detection with tcmp3cut commandline generation**

detectsilence was written by Tilmann Bitterberg. The version documented here is v0.0.1 (2003-07-26). This is a audio filter. It is a pre-processing only filter.

detectclipping - **detect clipping parameters (-j or -Y)**

detectclipping was written by Tilmann Bitterberg, A'rpi. The version documented here is v0.1.0 (2003-11-01). This is a video filter. It can handle RGB and YUV mode. It can be used as a pre-processing or as a post-processing filter.

- *range* = *%u-%u/%d* [default *0-4294967295/1*]
  apply filter to [start-end]/step frames
- *limit* = *%d* [default *24*]
  the sum of a line must be below this limit to be considered as black
- *post* (bool)
  run as a POST filter (calc -Y instead of the default -j)
  Detect black regions on top, bottom, left and right of an image. It is suggested that the filter is run for around 100 frames. It will print its detected parameters every frame. If you don't notice any change in the printout for a while, the filter probably won't find any other values. The filter converges, meaning it will learn.

dilyuvmmx - **yuv de-interlace filter plugin**

dilyuvmmx was written by Thomas Oestreich. The version documented here is v0.1.1 (2002-02-21). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter.

divxkey - **check for DivX 4.xx / OpenDivX / DivX;-) keyframe**

divxkey was written by Thomas Oestreich. The version documented here is v0.1 (2002-01-15). This is a video filter. It is a pre-processing only filter.

dnr - **dynamic noise reduction**

dnr was written by Gerhard Monzel. The version documented here is v0.2 (2003-01-21). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

* *lt* = *%d* [default *10*]
Threshold to blend luma/red
* *ll* = *%d* [default *4*]
Threshold to lock luma/red
* *ct* = *%d* [default *16*]
Threshold to blend croma/green+blue
* *cl* = *%d* [default *8*]
Threshold to lock croma/green+blue
* *sc* = *%d* [default *30*]
Percentage of picture difference (scene change)
see /docs/filter_dnr.txt (german only)

doublefps - **double frame rate by creating frames from fields**

doublefps was written by Tilmann Bitterberg. The version documented here is v0.2 (2003-06-23). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

* *shiftEven* = *%d* [default *0*]
Assume even field dominance

extsub - **DVD subtitle overlay plugin**

extsub was written by Thomas Oestreich. The version documented here is 0.3.5 (2003-10-15). This is a video filter. It can handle RGB and YUV mode. It can be used as a pre-processing or as a post-processing filter.

* *track* = *%d* [default *0*]
Subtitle track to render
* *vertshift* = *%d* [default *0*]
offset of subtitle with respect to bottom of frame in rows
* *timeshift* = *%d* [default *0*]
global display start time correction in msec
* *antialias* = *%d* [default *1*]
anti-aliasing the rendered text (0=off,1=on)
* *pre* = *%d* [default *1*]
Run as a pre filter
* *color1* = *%d* [default *0*]
Make a subtitle color visible with given intensity
* *color2* = *%d* [default *0*]
Make a subtitle color visible with given intensity

- *ca* = *%d* [default *0*]
  Shuffle the color assignment by choosing another subtitle color
- *cb* = *%d* [default *0*]
  Shuffle the color assignment by choosing another subtitle color

fields - Field adjustment plugin

fields was written by Alex Stewart. The version documented here is v0.1.1 (2003-01-21). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

- *flip* (bool)
  Exchange the top field and bottom field of each frame
- *shift* (bool)
  Shift the video by one field
- *flip_first* (bool)
  Normally shifting is performed before flipping, this option reverses that
  The 'fields' filter is designed to shift, reorder, and generally rearrange independent fields of an interlaced video input. Input retrieved from broadcast (PAL, NTSC, etc) video sources generally comes in an interlaced form where each pass from top to bottom of the screen displays every other scanline, and then the next pass displays the lines between the lines from the first pass. Each pass is known as a "field" (there are generally two fields per frame). When this form of video is captured and manipulated digitally, the two fields of each frame are usually merged together into one flat (planar) image per frame. This usually produces reasonable results, however there are conditions which can cause this merging to be performed incorrectly or less-than-optimally, which is where this filter can help.

  The following options are supported for this filter (they can be separated by colons):

  shift - Shift the video by one field (half a frame), changing frame boundaries appropriately. This is useful if a video capture started grabbing video half a frame (one field) off from where frame boundaries were actually intended to be.

  flip - Exchange the top field and bottom field of each frame. This can be useful if the video signal was sent "bottom field first" (which can happen sometimes with PAL video sources) or other oddities occurred which caused the frame boundaries to be at the right place, but the scanlines to be swapped.

  flip_first
  - Normally shifting is performed before flipping if both are specified. This option reverses that behavior. You should not normally need to use this unless you have some extremely odd input material, it is here mainly for completeness.

help - Print this text.

Note: the 'shift' function may produce slight color discrepancies if YV12 is used as the internal transcode video format (-V flag). This is because YV12 does not contain enough information to do field shifting cleanly. For best (but slower) results, use RGB mode for field shifting.

fps - **convert video frame rate, gets defaults from -f and --export_fps**

fps was written by Christopher Cramer. The version documented here is v0.2 (2003-08-10). This is a video filter. It can handle RGB and YUV mode. It can be used as a pre-processing or as a post-processing filter.

options: <input fps>:<output fps> example: -J fps=25:29.97 will convert from PAL to NTSC If no options are given, defaults or -f/--export_fps/--export_frc will be used.

hqdn3d - **High Quality 3D Denoiser**

hqdn3d was written by Daniel Moreno & A'rpi. The version documented here is v1.0.2 (2003-08-15). This is a video filter. It can handle YUV mode only. It supports multiple instances. It can be used as a pre-processing or as a post-processing filter.

- *luma* = *%f* [default *4.000000*]
  spatial luma strength
- *chroma* = *%f* [default *3.000000*]
  spatial chroma strength
- *luma_strength* = *%f* [default *6.000000*]
  temporal luma strength
- *chroma_strength* = *%f* [default *4.500000*]
  temporal chroma strength
- *pre* = *%d* [default *0*]
  run as a pre filter
  This filter aims to reduce image noise producing smooth images and making still images really still (This should enhance compressibility).

invert - **invert the image**

invert was written by Tilmann Bitterberg. The version documented here is v0.1.4 (2003-10-12). This is a video filter. It can handle RGB,YUV and YUV422 mode. It is a post-processing only filter.

- *range* = *%u-%u/%d* [default *0-4294967295/1*]
  apply filter to [start-end]/step frames

ivtc - **NTSC inverse telecine plugin**

ivtc was written by Thanassis Tsiodras. The version documented here is v0.4.1 (2004-06-01). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter.

- *verbose* (bool)
  print verbose information
- *field* = *%d* [default *0*]
  which field to replace (0=top 1=bottom)
- *magic* = *%d* [default *0*]
  perform magic? (0=no 1=yes)
  see /docs/README.Inverse.Telecine.txt

logo - **render image in videostream**

logo was written by Tilmann Bitterberg. The version documented here is v0.9 (2003-04-09). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

- *file = %s*
  Image filename
- *posdef = %d* [default *0*]
  Position (0=None, 1=TopL, 2=TopR, 3=BotL, 4=BotR, 5=Center)
- *pos = %dx%d* [default *0x0*]
  Position (0-width x 0-height)
- *range = %u-%u* [default *0-0*]
  Restrict rendering to framerange
- *ignoredelay* (bool)
  Ignore delay specified in animations
- *rgbswap* (bool)
  Swap red/blue colors
- *grayout* (bool)
  YUV only: don't write Cb and Cr, makes a nice effect
- *flip* (bool)
  Mirror image
  This filter renders an user specified image into the video. Any image format ImageMagick can read is accepted. Transparent images are also supported. Image origin is at the very top left.

  see /docs/filter_logo.txt

logoaway - **remove an image from the video**

logoaway was written by Thomas Wehrspann <thomas@wehrspann.de>. The version documented here is v0.4 (2003-09-03). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter. It supports multiple instances.

- *range = %d-%d* [default *0-4294967295*]
  Frame Range
- *pos = %dx%d* [default *0x0*]
  Position of logo
- *size = %dx%d* [default *10x10*]
  Size of logo
- *mode = %d* [default *0*]
  Filter Mode (0=none,1=solid,2=xy,3=shape)
- *border* (bool)
  Visible Border
- *xweight = %d* [default *50*]
  X-Y Weight(0%-100%)
- *fill = %2x%2x%2x* [default *000*]
  Solid Fill Color(RGB)
- *file = %s*
  Image with alpha/shape information
  This filter removes an image in a user specified area from the video. You can choose from different methods.

  see /docs/filter_logoaway.txt

lowpass - **High and low pass filter**

lowpass was written by Tilmann Bitterberg. The version documented here is v0.1.0 (2002-02-26). This is a audio filter. It is a pre-processing only filter.

* *taps* = *%d* [default *30*]
strength (may be negative)

mask - **Filter through a rectangular Mask**

mask was written by Thomas Östreich, Chad Page. The version documented here is v0.2.3 (2003-10-12). This is a video filter. It can handle RGB,YUV and YUV422 mode. It is a pre-processing only filter.

* *lefttop* = *%dx%d* [default *0x0*]
Upper left corner of the box
* *rightbot* = *%dx%d* [default *32x32*]
Lower right corner of the box
This filter applies an rectangular mask to the video. Everything outside the mask is set to black.

modfps - **plugin to modify framerate**

modfps was written by Marrq. The version documented here is v0.10 (2003-08-18). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

* *mode* = *%d* [default *1*]
mode of operation
* *infps* = *%f* [default *25.000000*]
Original fps
* *infrc* = *%d* [default *0*]
Original frc
* *examine* = *%d* [default *5*]
How many frames to buffer
* *subsample* = *%d* [default *32*]
How many pixels to subsample
* *clonetype* = *%d* [default *0*]
How to clone frames
* *verbose* = *%d* [default *1*]
run in verbose mode
This filter aims to allow transcode to alter the fps of video. While one can reduce the fps to any amount, one can only increase the fps to at most twice the original fps.

There are two modes of operation, buffered and unbuffered, unbuffered is quick, but buffered, especially when dropping frames should look better.

For most users, modfps will need either no options, or just mode=1

see /docs/README.filter.modfps

msharpen - **VirtualDub's MSharpen Filter**

msharpen was written by Donald Graft, William Hawkins. The version documented here is (1.0) (2003-07-17). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

* *strength* = *%d* [default *100*]

How much of the effect
- *threshold = %d* [default *10*]
  How close a pixel must be to the brightest or dimmest pixel to be mapped
- *highq = %d* [default *1*]
  Tradeoff speed for quality of detail detection
- *mask = %d* [default *0*]
  Areas to be sharpened are shown in white
  This plugin implements an unusual concept in spatial sharpening. Although designed specifically for anime, it also works well with normal video. The filter is very effective at sharpening important edges without amplifying noise.

  * Strength 'strength' (0-255) [100]
  This is the strength of the sharpening to be applied to the edge detail areas. It is applied only to the edge detail areas as determined by the 'threshold' parameter. Strength 255 is the strongest sharpening.
  * Threshold 'threshold' (0-255) [10]
  This parameter determines what is detected as edge detail and thus sharpened. To see what edge detail areas will be sharpened, use the 'mask' parameter.
  * Mask 'mask' (0-1) [0]
  When set to true, the areas to be sharpened are shown in white against a black background. Use this to set the level of detail to be sharpened. This function also makes a basic edge detection filter.
  * HighQ 'highq' (0-1) [1]
  This parameter lets you tradeoff speed for quality of detail detection. Set it to true for the best detail detection. Set it to false for maximum speed.

nored - **nored the image**

nored was written by Tilmann Bitterberg. The version documented here is v0.1.3 (2003-01-26). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter.

- *range = %u-%u/%d* [default *0x4294967295/1*]
  apply filter to [start-end]/step frames
- *subst = %d* [default *2*]
  substract N red from Cr

normalize - **Volume normalizer**

normalize was written by pl, Tilmann Bitterberg. The version documented here is v0.1.1 (2002-06-18). This is a audio filter. It is a pre-processing only filter.

- *smooth = %f* [default *0.06*]
  Value for smoothing]0.0 1.0[
- *smoothlast = %f* [default *0.06*]
  Value for smoothing last sample]0.0, 1.0[
- *algo = %d* [default *1*]
  Algorithm to use (1 or 2). 1=uses a 1 value memory and coefficients new=a*old+b*cur (with a+b=1). 2=uses several samples to smooth the variations (standard weighted mean on past samples)

null - **demo filter plugin; does nothing**

null was written by Thomas Oestreich. The version documented here is v0.2 (2003-09-04). This is a video and audio filter. It can be used as a pre-processing or as a post-processing filter.

pp - **Mplayers postprocess filters**

pp was written by Michael Niedermayer et al, Gerhard Monzel. The version documented here is v1.2.4 (2003-01-24). This is a video filter. It can handle YUV mode only. It supports multiple instances. It can be used as a pre-processing or as a post-processing filter.

- * *hb = %d:%d* [default *64:40*]
  Horizontal deblocking filter
- * *vb = %d:%d* [default *64:40*]
  Vertical deblocking filter
- * *h1* (bool)
  Experimental h deblock filter 1
- * *v1* (bool)
  Experimental v deblock filter 1
- * *dr* (bool)
  Deringing filter
- * *al* (bool)
  Automatic brightness / contrast
- * *f* (bool)
  Stretch luminance to (0..255)
- * *lb* (bool)
  Linear blend deinterlacer
- * *li* (bool)
  Linear interpolating deinterlace
- * *ci* (bool)
  Cubic interpolating deinterlacer
- * *md* (bool)
  Median deinterlacer
- * *de* (bool)
  Default preset (hb:a/vb:a/dr:a/al)
- * *fa* (bool)
  Fast preset (h1:a/v1:a/dr:a/al)
- * *tn = %d:%d:%d* [default *64:128:256*]
  Temporal Noise Reducer (1<=2<=3)
- * *fq = %d* [default *15*]
  Force quantizer
- * *pre* (bool)
  Run as a PRE filter

preview - **xv/sdl/gtk preview plugin**

preview was written by Thomas Oestreich. The version documented here is v0.1.4 (2002-10-08). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

XXX: Write me

pv - **xv only preview plugin**

pv was written by Thomas Oestreich, Tilmann Bitterberg. The version documented here is v0.2.3 (2004-06-01). This is a video filter. It can handle YUV and YUV422 mode. It is a post-processing only filter.

- *cache* = *%d* [default *15*]
  Number of raw frames to cache for seeking
- *skip* = *%d* [default *0*]
  display only every Nth frame
  The filter listens to mouse and key strokes. If you click into the preview window, the first time say near the upper left corner and the second time near the lower right corner, transcode will draw a rectangle and will print out the coordinates of this rectangle on stdout and the socket. See the table below for available keys.

  When you start transcode with the --socket option and the pv filter with (for example) cache=20 you can talk to transcode and the pv filter at runtime using the socket.

```
transcode -i file.avi -V -J pv=cache=30 --socket /tmp/sock
```

Available Commands

Key	Socket*	Effect
RET	draw	Redraws the image, applying filters.
u	undo	goes to image before draw.
SPACE	pause	pause the preview (and transcode).
UP	fastfw	In pause mode, step forward 5 frames.
RIGHT	slowfw	In pause mode, step forward 1 frame.
DOWN	fastbw	In pause mode, step back 5 frames.
LEFT	slowbw	In pause mode, step back 1 frame.
q	display	Toggle display of frames
s	slower	slow down
f	faster	speed up
y	toggle	toggle displaying only every 5 frames
j	grab	Save a JPEG
r	rotate	Rotate AVI file after next keyframe

(*) all commands must be prefixed with "preview ".

resample - **audio resampling filter plugin**

resample was written by Thomas Oestreich. The version documented here is v0.1.4 (2003-08-22). This is a audio filter. It is a pre-processing only filter.

skip - **skip all listed frames**

skip was written by Thomas Oestreich. The version documented here is v0.0.1 (2001-11-27). This is a video and audio filter. It is a pre-processing only filter.

* *fstart1-fend1 [fstart2-fend2 [..]] = %s*

apply filter [start-end] frames

slowmo - **slow-motion effect**

slowmo was written by Tilmann Bitterberg. The version documented here is v0.2 (2003-06-23). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

This filter produces a simple slow-motion effect by duplicating certain frames. I have seen this effect on TV and despite its the simple algorithm it works quite well. The filter has no options.

smartbob - **Motion-adaptive deinterlacing for double-frame-rate output.**

smartbob was written by Donald Graft, Tilmann Bitterberg. The version documented here is v1.1beta2 (2003-06-23). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

* *motionOnly* = *%d* [default *0*]

Show motion areas only

* *shiftEven* = *%d* [default *0*]

Blend instead of interpolate in motion areas

* *threshold* = *%d* [default *12*]

Motion Threshold

* *denoise* = *%d* [default *1*]

Phase shift

This filter only makes sence when fed by -J doublefps. It will take the field-frames which filter_doublefps produces and generates full-sized motion adaptive deinterlaced output at the double import framerate. If you force reading the imput file twice its actual frames per second, A/V will stay in sync (for PAL): -f 50 -J doublefps=shiftEven=1,smartbob=denoise=1:threshold=12

smartdeinter - **VirtualDub's smart deinterlacer**

smartdeinter was written by Donald Graft. The version documented here is v2.7b (2003-02-01). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

* *motionOnly* = *%d* [default *0*]

Show motion areas only

* *Blend* = *%d* [default *0*]

Blend instead of interpolate in motion areas

* *threshold* = *%d* [default *15*]

Motion Threshold

* *scenethreshold* = *%d* [default *100*]

Scene Change Threshold

* *fieldShift* = *%d* [default *0*]

Phase shift

* *inswap* = *%d* [default *0*]

Field swap before phase shift

* *outswap* = *%d* [default *0*]

Field swap after phase shift

- *noMotion* = *%d* [default *0*]
  Disable motion processing
- *highq* = *%d* [default *0*]
  Motion map denoising for field-only
- *diffmode* = *%d* [default *0*]
  Motion Detection (0=frame, 1=field, 2=both)
- *colordiff* = *%d* [default *1*]
  Compare color channels instead of luma
- *cubic* = *%d* [default *0*]
  Use cubic for interpolation
  This filter provides a smart, motion-based deinterlacing capability. In static picture areas, interlacing artifacts do not appear, so data from both fields is used to provide full detail. In moving areas, deinterlacing is performed

smartyuv - **Motion-adaptive deinterlacing**

smartyuv was written by Tilmann Bitterberg. The version documented here is 0.1.4 (2003-10-13). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter.

- *motionOnly* = *%d* [default *0*]
  Show motion areas only, blacking out static areas
- *diffmode* = *%d* [default *0*]
  Motion Detection (0=frame, 1=field, 2=both)
- *threshold* = *%d* [default *14*]
  Motion Threshold (luma)
- *chromathres* = *%d* [default *7*]
  Motion Threshold (chroma)
- *scenethres* = *%d* [default *31*]
  Threshold for detecting scenechanges
- *highq* = *%d* [default *1*]
  High-Quality processing (motion Map denoising)
- *cubic* = *%d* [default *1*]
  Do cubic interpolation
- *Blend* = *%d* [default *1*]
  Blend the frames for deinterlacing
- *doChroma* = *%d* [default *1*]
  Enable chroma processing (slower but more accurate)
- *verbose* = *%d* [default *0*]
  Verbose mode
  This filter is basically a rewrite of the smartdeinter filter by Donald Graft (without advanced processing options) for YUV mode only. Its faster than using the smartdeinter in YUV mode and is also tuned with its threshold settings for YUV mode. The filter detects motion and static areas in an image and only deinterlaces (either by blending or by cubic interpolation) the moving areas. The result is an image with high detail in static areas, no information is lost there.

  The threshold settings should be sufficent for most users. As a rule of thumb, I recommend setting the chroma threshold to about the half of the luma threshold. If you want more deinterlacing, lower the thresholds. The scene threshold can be easily found by turning on verbose mode and the preview filter. In verbose mode, the filter will print out, when it detects a scene change. If scenechanges go by unnoticed, lower the scene threshold. You can completly disable chroma processing with the doChroma=0 option. Here is a sample commandline

```
-J
smartyuv=highq=1:diffmode=2:cubic=1:Blend=1:chromathres=4:threshold=8:
doChroma=1
```

smooth - **(single-frame) smoothing plugin**

smooth was written by Chad Page. The version documented here is v0.2.3 (2003-03-27). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter. It supports multiple instances.

- *strength = %f* [default *0.25*]
  Blending factor
- *cdiff = %d* [default *6*]
  Max difference in chroma values
- *ldiff = %d* [default *8*]
  Max difference in luma value
- *range = %d* [default *4*]
  Search Range

  "single-frame" means it only works with the current frame, it does not need the next or the previous frame for operation. Usually smoothing is done by talking the data of previous frames into account to see which parts of the picture can be "safely" smoothed, this filter only needs one frame.

subtitler - **subtitle filter**

subtitler was written by Panteltje. The version documented here is v0.4 (2002-02-19). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

```
Usage -J subtitler="[no_objects] [subtitle_file=s] [color_depth=n] [font_dir=s]
[font=n] [font_factor=f [frame_offset=n] [debug] [help]" f is float, h is hex, n is
integer, s is string.

no_objects disables subtitles and other objects (off).
color_depth= 32 or 24 (overrides X auto) (32).
font= 0 or 1, 1 gives strange symbols... (0).
font_dir= place where font.desc is (~/.subtitles/font).
font_factor= .1 to 100 outline characters (10.75).
frame_offset= positive (text later) or negative (earlier) integer (0).
subtitle_file= pathfilename.ppml location of ppml file (~/.subtitles/demo.ppml).

debug prints debug messages (off).
help prints this list and exit.
```

tc_video - **video 23.9 -> 29.9 telecide filter**

tc_video was written by Tilmann Bitterberg. The version documented here is v0.2 (2003-06-10). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter.

testframe - **generate stream of testframes**

testframe was written by Thomas Oestreich. The version documented here is v0.1.3 (2003-09-04). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

- *mode = %d* [default *0*]
  Choose the test pattern (0-4 interlaced, 5 colorfull)

text - **write text in the image**

text was written by Tilmann Bitterberg. The version documented here is v0.1.2 (2003-06-27). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

- *range = %u-%u/%d* [default *0-4294967295/1*]
  apply filter to [start-end]/step frames
- *string = %s*
  text to display (no ':') [defaults to `date`]
- *font = %s*
  full path to font file [defaults to arial.ttf]
- *points = %d* [default *25*]
  size of font (in points)
- *dpi = %d* [default *96*]
  resolution of font (in dpi)
- *fade = %d* [default *0*]
  fade in/out (0=off, 1=slow, 10=fast)
- *antialias = %d* [default *1*]
  Anti-Alias text (0=off 1=on)
- *pos = %dx%d* [default *0x0*]
  Position (0-width x 0-height)
- *posdef = %d* [default *0*]
  Position (0=None 1=TopL 2=TopR 3=BotL 4=BotR 5=Cent 6=BotCent)
- *notransparent* (bool)
  disable transparency (enables block box)
  see /docs/filter_text.txt

unsharp - **unsharp mask & gaussian blur**

unsharp was written by Rémi Guyomarch. The version documented here is v1.0.1 (2003-10-27). This is a video filter. It can handle YUV mode only. It is a post-processing only filter.

- *amount = %f* [default *0.0*]
  Luma and chroma (un)sharpness amount
- *matrix = %dx%d* [default *0x0*]
  Luma and chroma search matrix size
- *luma = %f* [default *0.0*]
  Luma (un)sharpness amount
- *chroma = %f* [default *0.0*]
  Chroma (un)sharpness amount
- *luma_matrix = %dx%d* [default *0x0*]
  Luma search matrix size
- *chroma_matrix = %dx%d* [default *0x0*]
  Chroma search matrix size
- *pre = %d* [default *0*]

run as a pre filter
This filter blurs or sharpens an image depending on the sign of "amount". You can either set amount for both luma and chroma or you can set it individually (recommended). A positive value for amount will sharpen the image, a negative value will blur it. A sane range for amount is -1.5 to 1.5.

The matrix sizes must be odd and define the range/strength of the effect. Sensible ranges are 3x3 to 7x7.

It sometimes makes sense to sharpen the sharpen the luma and to blur the chroma. Sample string is:

luma=0.8:luma_matrix=7x5:chroma=-0.2:chroma_matrix=3x3

videocore - **Core video transformations**

videocore was written by Thomas, Tilmann. The version documented here is v0.0.4 (2003-02-01). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

* *deinterlace = %d* [default *0*]
same as -I
* *flip* (bool)
same as -z
* *mirror* (bool)
same as -l
* *rgbswap* (bool)
same as -k
* *decolor* (bool)
same as -K
* *dgamma = %f* [default *0.000000*]
same as -G
* *antialias = %d/%f/%f* [default *0/0.33/0.50*]
same as -C/weight/bias

whitebalance - **White Balance Filter - correct images with a broken white balance**

whitebalance was written by Guillaume Cottenceau. The version documented here is v0.1 (2003-10-01). This is a video filter. It can handle RGB and YUV mode. It is a pre-processing only filter.

* *level = %d* [default *40*]
Strength of filter (may also be negative for opposite effect)
* *limit = %s* [default disabled]
A string containing a series of frame numbers prepended by '-' to switch off or '+' to switch on (for example, -50+80-120 will disable from frame number 50 to number 80, then disable from 120 to the end)
This filter allows correcting movies with a broken white balance (e.g. bluish movie, for example).

xharpen - **VirtualDub's XSharpen Filter**

xharpen was written by Donald Graft, Tilmann Bitterberg. The version documented here is (1.0b2) (2003-02-12). This is a video filter. It can handle RGB and YUV mode. It is a post-processing only filter.

- *strength* = *%d* [default *200*]
  How much of the effect
- *threshold* = *%d* [default *255*]
  How close a pixel must be to the brightest or dimmest pixel to be mapped
  This filter performs a subtle but useful sharpening effect. The result is a sharpening effect that not only avoids amplifying noise, but also tends to reduce it. A welcome side effect is that files processed with this filter tend to compress to smaller files.

  Strength 'strength' (0-255) [200]
  When this value is 255, mapped pixels are not blended with the original pixel values, so a full-strength effect is obtained. As the value is reduced, each mapped pixel is blended with more of the original pixel. At a value of 0, the original pixels are passed through and there is no sharpening effect.

  Threshold 'threshold' (0-255) [255]
  This value determines how close a pixel must be to the brightest or dimmest pixel to be mapped. If a pixel is more than threshold away from the brightest or dimmest pixel, it is not mapped. Thus, as the threshold is reduced, pixels in the mid range start to be spared.

yuvdenoise - **mjpegs YUV denoiser**

yuvdenoise was written by Stefan Fendt, Tilmann Bitterberg. The version documented here is v0.2.1 (2003-11-26). This is a video filter. It can handle YUV mode only. It can be used as a pre-processing or as a post-processing filter.

- *radius* = *%d* [default *8*]
  Search radius
- *threshold* = *%d* [default *5*]
  Denoiser threshold
- *pp_threshold* = *%d* [default *4*]
  Pass II threshold
- *delay* = *%d* [default *3*]
  Average 'n' frames for a time-lowpassed pixel
- *postprocess* = *%d* [default *1*]
  Filter internal postprocessing
- *luma_contrast* = *%d* [default *100*]
  Luminance contrast in percent
- *chroma_contrast* = *%d* [default *100*]
  Chrominance contrast in percent.
- *sharpen* = *%d* [default *125*]
  Sharpness in percent
- *deinterlace* = *%d* [default *0*]
  Force deinterlacing
- *mode* = *%d* [default *0*]
  [0]: Progressive [1]: Interlaced [2]: Fast
- *scene_thres* = *%d%%* [default *50*]
  Blocks where motion estimation should fail before scenechange
- *block_thres* = *%d* [default *1024*]
  Every SAD value greater than this will be considered bad
- *do_reset* = *%d* [default *2*]

Reset the filter for `n' frames after a scene

- *increment_cr* = *%d* [default *2*]
  Increment Cr with constant
- *increment_cb* = *%d* [default *2*]
  Increment Cb with constant
- *border* = *%dx%d-%dx%d* [default *0x0-32x32*]
  Active image area
- *pre* = *%d* [default *0*]
  run this filter as a pre-processing filter
  see /docs/filter_yuvdenoise.txt

yuvmedian - mjpegs YUV median filter

yuvmedian was written by Mike Bernson, Tilmann Bitterberg. The version documented here is v0.1.0 (2003-01-24). This is a video filter. It can handle YUV mode only. It can be used as a pre-processing or as a post-processing filter.

- *radius_luma* = *%d* [default *2*]
  Radius for median (luma)
- *radius_chroma* = *%d* [default *2*]
  Radius for median (chroma)
- *threshold_luma* = *%d* [default *2*]
  Trigger threshold (luma)
- *threshold_chroma* = *%d* [default *2*]
  Trigger threshold (chroma)
- *interlace* = *%d* [default *0*]
  Treat input as interlaced
- *pre* = *%d* [default *1*]
  Run as a PRE filter

yuy2tov12 - YUY2 to YV12 converter plugin

yuy2tov12 was written by Thomas Oestreich. The version documented here is v0.0.2 (2003-09-04). This is a video filter. It can handle YUV mode only. It is a pre-processing only filter.

NOTES

- Most source material parameter are auto-detected.

- Clipping region options are expanded symmetrically. Examples:

  - -j 80 will be expanded to -j 80,0,80,0

  - -j 80,8 will be expanded to -j 80,8,80,8

  - -j 80,8,10 will be expanded to -j 80,8,10,8

- maximum image size is 1920x1088.

- The video frame operations ordering is fixed: "-j -I -X -B -Z -Y -r -z -l -k -K -G -C" (executed from left to right).

- Shrinking the image with '-B' is not possible if the image width/height is not a multiple of 8, 16 or 32.

- Expanding the image with '-X' is not possible if the image width/height is not a multiple of 8, 16 or 32.

- The final frame width/height should be a multiple of 8. (to avoid encoding problems with some codecs)
  1. Reducing the video height/width by 2,4,8 Option '-r factor' can be used to shrink the video image by a constant factor, this factor can be 2,4 or 8.
  2. Clipping and changing the aspect ratio *transcode* uses 3 steps to produce the input image for the export modules
     1. Clipping of the input image.
     2. Changing the aspect ratio of the 1) output.
     3. Clipping of the 2) output.

- *Bits per pixel* (bits/pixel) is a value transcode calculates and prints when starting up. It is mainly useful when encoding to MPEG4 (xvid, divx, etc). You'll see line like

  ```
  [transcode] V: bits/pixel | 0.237
  ```

  Simplified said, bits/pixel quantifies how good an encode will be. Although this value depends heavily on the used input material, as a general rule of thump it can be said that values greater or close to 0.2 will result in good encodes, encodes with values less than 0.15 will have noticeable artifacts.

  Bits per pixel depends on the resolution, bitrate and frames per second. If you have a low value (< 0.15), you might want to raise the bitrate or encode at a lower resolution. The exact formula is

bitrate*1000
bpp = ---------------
width*height*fps

- *AC3 / Multiple channels*

  When you do import an audio stream which has more then two audio channels - this is usually the case for AC3 audio - transcode will automagically downmix to two channels (stereo). You'll see line like

  [transcode] A: downmix | 5 channels -> 2 channels

  This is done, because most encoders and audio filters can not handle more than 2 channels correctly. The PCM internal representation does not support more than two channels, audio will be downmixed to stereo **No** downmix will happen, if you use AC3 as the internal audio codec or use audio pass-through.

EXAMPLES

The following command will read it's input from the DVD drive (I assume */dev/dvd* is a symbolic link to the actual DVD device) and produce a splitted divx4 movie according to the chapter information on the DVD medium. The output files will be named *my_movie-ch00.avi*, *my_movie-ch01.avi* ...

```
transcode -i /dev/dvd/ -x dvd -V -j 16,0 -B 5,0 \
-Y 40,8 -s 4.47 -U my_movie -y xvid -w 1618
```

Option **-V** tells *transcode* to use YUV as internal video colorspace, which saves a lot of CPU/PCI bandwidth.

Option **-j 16,0** will be expanded to **-j 16,0,16,0** and results in 16 rows from the top and the bottom of the image to be cut off. This may be usefull if the source consists of black top and bottom bars.

Option **-B 5,0** tells *transcode* to shrink the resulting image by 5*32=160 rows in height.

Option **-Y 40,8** will be expanded to **-Y 40,8,40,8** and tells *transcode* to ...

Option **-s 4.47** tells *transcode* to increase audio volume by a factor 4.47.

Option **-U my_movie** tells *transcode* to operate in chapter mode and produce output to files named *my_movie-ch00.avi*, *my_movie-ch01.avi*...
You can either merge the files afterwards with avimerge or add the option --no_split to the command line.

Option **-y xvid** tells *transcode* to use the export module export_xvid.so which in turn uses the XviD encoder to encode the video.

Option **-w 1618** tells *transcode* to set the encoder bitrate to 1618 which is lower than the default of 1800 and results in smaller files with the loss of visual quality.

Lets assume that you have an NTSC DVD (720x480) and you want to make an NTSC-SVCD

The frame size of the DVD movie is 720x480 @ 16:9. For the purpose of frame resizing, the width 720 is not relavant (that is, it will not be used in the following reasoning). It is not needed because the original frame size is really defined by the frame height and aspect ratio. The _final result_ should be 640x480, encoded as 480x480 @ 4:3 (the height 480 multiplied by the aspect ratio 4:3 gives the width 640). This same frame size (640x480) can also be encoded as 640x360 @ 16:9 (the height 360 multiplied by the aspect ratio 16:9 gives the width 640).

As the _original video_ has aspect ratio 16:9, first we resize to 640x360, keeping that aspect ratio. But the aspect ratio has to be changed to 4:3. To find the frame size in the new aspect ratio the height 360 is multiplied by the new aspect ratio, giving the width 480. This is accomplished with the transcode options "--export_asr 2 -Z 480x360,fast".

To avoid stretching the video height in this change (because the new aspect ratio is less than the original), black borders should be added at the top and bottom of the video, bringing the frame to the desired 480x480 @ 4:3 size. The transcode option for this is "-Y -60,0,-60,0".

If for some reason (maybe a subtitle filter) the black borders (of height 60 each) should be added before resizing the frame and changing the aspect ratio to 4:3. One reason for that would be the need of running a _pre_ filter after adding the black borders. Then the options "-j" or "--pre_clip" can be used instead of "-Y". In this case the black border height has to be recalculated by applying the aspect ratio 4:3 to the value already found: 60 * (4/3) = 80. The transcode options "-j -80,0,-80,0" or "--pre_clip -80,0,-80,0" are then used instead of "-Y -60,0,-60,0", and "-Z 480x360,fast" is replaced by "-Z 480x480,fast".

AUTHORS

transcode was written by Thomas Östreich <ostreich@theorie.physik.uni-goettingen.de> and Tilmann Bitterberg with contributions from many others. See *AUTHORS* for details.

SEE ALSO

avifix(1), **avisync**(1), **avimerge**(1), **avisplit**(1), **tcprobe**(1), **tcscan**(1), **tccat**(1), **tcdemux**(1), **tcextract**(1), **tcdecode**(1), **tcmodinfo**(1), **tcxmlcheck**(1), **transcode**(1)

WWW

Frequently asked questions (FAQ) at
http://www.theorie.physik.uni-goettingen.de/~ostreich/transcode/html/faq.html
Example transcode sessions
http://www.theorie.physik.uni-goettingen.de/~ostreich/transcode/html/index.html

BUGS

see http://www.theorie.physik.uni-goettingen.de/~ostreich/transcode/pending.html

---

Index