ffmpeg

Video transcodieren (Auflösung und Dateigröße kleiner machen)

mit h264

```
ffmpeg -i video_1920x1080.mp4 -vf scale=-1:720 -c:v libx264 -crf 18 -preset veryslow -c:a copy video_1280x720.mp4
```

- -i = input file
- -vf = videofilter, in diesem Fall wird scale verwendet, -1 bedeutet proportional verändert
- -c:v = video codec, in diesem Fall h264
- -crf = Qualitätsstufe: 0 ist lossless und 52 stärkste Kompression
- -preset = wirkt sich auch auf die Qualität aus
- -c:a audio codec, bei copy wird er übernommen

HEVC, h265 2pass-Kodierung

```
ffmpeg -y -i input.mp4 -c:v libx265 -b:v 2600k -x265-params pass=1 -an -f null /dev/null && \
ffmpeg -i input.mp4 -c:v libx265 -b:v 2600k -x265-params pass=2 -c:a copy output.mp4
```

unter Windows

```
ffmpeg -i input.mp4 -c:v libaom-av1 -b:v 2M -pass 1 -an -f null NUL &&
ffmpeg -i input.mp4 -c:v libaom-av1 -b:v 2M -pass 2 -c:a libopus output.mkv
```

- -i = input file
- -c:v = video codec hier AV1 (sehr, seeehr langsam)
- -b:v = video Bitrate bei ca 2 MB
- -pass 1 = erster Durchgang
- -pass 2 = zweiter Durchgang
- -an = Audio auslassen (im ersten Durchgang) „audio none“
- -f = format
- null /dev/null [Linux] = Null Muxer (ohne Datei-Ausgabe), null NUL[Windows]
- -c:a libopus = audio codec opus

AV1-Kodierung mit libsvtav1, entwickelt von Intel in Zusammenarbeit mit Netflix.

```
ffmpeg - i input.mp4 -c:v libsvtav1 -vf format=yuv420p10le -preset 6 -crf 23 -g 30 output.mkv
```

AV1-Kodierung mit libsvtav1 nach sebsauvage's Anleitung

siehe https://sebsauvage.net/wiki/doku.php?id=ffmpeg#re-encodage_en_av1

```
ffmpeg -i input.mkv -map 0:v -c:v libsvtav1 -preset 4 -crf 30 -tune 0 -
pix_fmt yuv420p10le -g 120 -map 0:a? -c:a copy -map 0:s? -c:s copy out.mkv
```

-i input Name der Eingabedatei.

-map 0:v -c:v libsvtav1 es wird der svt-av1-Encoder verwendet. svt-av1 ist schnell

-crf 30: Komprimierungsrate. Der CRF ist die gewünschte visuelle Qualität (die Videobitrate ist daher variabel, um eine konstante visuelle Qualität aufrechtzuerhalten. Ein niedrigerer Wert bedeutet bessere Qualität, aber eine größere resultierende Datei. Oberhalb von 36 beginnt die Qualität recht schnell zu sinken.

-tune 0: Um eine konstante Qualität zu erreichen, benötigt man einen Algorithmus, der diese Qualität misst. Durch das Hinzufügen dieser Option wird ein Algorithmus ausgewählt, der dem menschlichen Sehverhalten näher kommt.

-preset 4: Der svt-av1-Codec verfügt über unzählige Parameter, daher gibt es verschiedene presets, die es ermöglichen es Ihnen, schneller zu komprimieren (auf Kosten der Größe der endgültigen Datei). Je niedriger der Wert, desto kleiner ist die erhaltene Datei, desto länger ist jedoch die Komprimierungszeit. Folgendes ist zu beachten: Die Werte 4 bis 6 liefern gute Ergebnisse mit anständigen Kodierungszeiten. Die Verwendung von Werten unter 4 verlängert die Komprimierungszeit erheblich und führt zu einem vernachlässigbaren Gewinn. Bereits zwischen 4 und 6 ist der Unterschied gering. Weitere Informationen finden Sie auf dieser Seite. Das Interessante an Preset 4 ist, dass ab diesem Schwellenwert die globale Bewegungskompensation aktiviert wird, was bei Filmen mit viel Bewegung von Vorteil sein kann. Ab 7 beginnt die Qualität immer mehr zu leiden. Hohe Voreinstellungen sind eher für die Echtzeitkodierung konzipiert.

-pix_fmt yuv420p10le: Standardmäßig werden Farben/Helligkeit auf 8 Bit kodiert. Diese Option erzwingt die 10-Bit-Kodierung, wodurch Streifenbildung in dunklen Szenen weitgehend vermieden werden kann, ohne die Dateigröße wesentlich zu beeinträchtigen.

-g 120: Fügt alle 120 Frames ein I-Frame hinzu. Bei 24 Bildern pro Sekunde sind das alle 5 Sekunden. Referenzbilder erleichtern den Spielern das „Suchen“, d. h. das sofortige Auflegen des Videostreams, wenn Sie den Wiedergabecursor bewegen. (Standardmäßig legt ffmpeg den Wert 9999 fest, was die Navigation in Videos sehr erschwert.

-map 0:a? -c:a copy: Für Audio alle Audiospuren unverändert kopieren (ohne Neukodierung). Das kann man natürlich ändern, indem man beispielsweise den Opus-Codec verwendet um größe Kompression im Audiobereich zu erzielen-

-map 0:s? -c:s copy: Kopiert auch alle Untertitelspuren. Das '?' (Fragezeichen) wird verwendet, um Fehler zu vermeiden, wenn keine Untertitelspur vorhanden ist (s. o. für Audio).

output.mkv: Output-Datei. Ein mkv-Container ist zu empfehlen, da dieser in der Lage ist, den Videostream, aber auch mehrere Audiostreams und Untertitel zu speichern.

Video/Audio-delay korrigieren

Fall 1: audio beginnt vor video (150 ms = 0.15 s)

```
ffmpeg -i video.mp4 -itsoffset 0.150 -i video.mp4 -c:v copy -c:a copy -map 0:0 -map 1:1 video_insync.mp4
```

Fall 2: video beginnt vor audio (150 ms = 0.15 s)

```
ffmpeg -i video.mp4 -itsoffset 0.150 -i video.mp4 -c:v copy -c:a copy -map 0:1 -map 1:0 video_insync.mp4
```

Videos zusammenführen (concatenate)

```
ffmpeg -f concat -safe 0 -i liste.txt -c copy OUT.mp4
```

wobei liste.txt unter Windows

liste.txt

```
file 'd:/videos/t1.mp4'
file 'd:/videos/t2.mp4'
```

Video schneiden

```
ffmpeg -i in.mp4 -ss [start] -t [dauer] -c copy out.mp4
```

oder

```
ffmpeg -i in.mp4 -ss [start] -to [bis-zum-zeitpunkt] -c copy out.mp4
```

Audio aus Container extrahieren

```
ffmpeg -i input-video.mp4 -vn -acodec copy output-audio.aac
```

Audio file einem Video hinzufügen

```
ffmpeg -i IN-video-mit-EN-tonspur.mp4 \
-i audio-DE.m4a \
-c copy \
```

```
-map 0:v:0 \
-map 0:a:0 \
-map 1:a:0 \
-metadata:s:a:0 language=eng \
-metadata:s:a:1 language=deu \
OUT-video-EN-DE.mp4
```

Das `mapping`: Datei `0` ist das Video, Datei `1` die deutsche Audiospur und schreibt die Metadaten der Sprache mit hinein, damit man bei switchen im Video die entsprechende Tonspur angezeigt bekommt

bzw. deutsch und englische Spur

```
ffmpeg -i in-ohne-audio.mp4 \
-i deu.aac \
-i eng.aac \
-c copy \
-map 0:v:0 \
-map 1:a:0 \
-map 2:a:0 \
-metadata:s:a:0 language=deu \
-metadata:s:a:1 language=eng \
out.mp4
```

Audio in Video normalisieren (amplify/compress)

```
ffmpeg -i video.mp4 -filter:a "speechnorm=e=12.5:r=0.0001:l=1" video-
normalized.mp4
```

Links

- h264-encoding guide
- libaom AV1 Encoding Guide
- ffmpeg wasm-version for using online

Videoformate 16:9

```
Recommended width and height for videos with 16:9 aspect ratios:

Best Choice:     2nd Best:        3rd Best:
Multiples of 16  Multiples of 8   Multiples of 4
1920 x 1080      1792 x 1008      1856 x 1044
1280 x 720       1152 x 648       1216 x 684
1024 x 576        896 x 504       1088 x 612
 768 x 432        640 x 360        960 x 540
 512 x 288        384 x 216        832 x 468
 256 x 144        128 x 72         704 x 396
```

```
                                    576 x 324
                                    448 x 252
                                    320 x 180
                                    192 x 108
```

Transkodieren mit allen Streams

entscheidend ist `-map 0`, das alle streams einbindet

```
ffmpeg -i input.mkv -c copy -map 0 output.mp4
```

oder mit h264 transcode-Einstellungen

```
ffmpeg \
-analyzeduration 100M -probesize 100M \
-i input.mkv \
-map 0 \
-codec:v libx264 -crf 21 \
-codec:a aac -b:a 384k \
-codec:s copy \
output.mp4
```

Untertitel einbinden

```
ffmpeg -i infile.mp4 -i infile.srt -c copy -c:s mov_text outfile.mp4
```

Lame Audio LAME VBR-Encoding

```
ffmpeg -i input.wav -c:a libmp3lame -q:a 0 output.mp3
```

verwendet den LAME-Encoder mit `-V 0` Preset, also VBR und ca. ~245 kbps, siehe Tabelle

Videostream mitschneiden

Videostream für 1 Stunde ohne reencoding

```
ffmpeg -y -i http://domain.tld/stream.m3u8 -c:v copy -c:a copy -t 1:00:00
out.mp4 > /tmp/streamrec.log 2>&1;
```

```
ffmpeg -y -i http://domain.tld/stream.m3u8 -c copy -t 1:00:00 out.mp4 >
/tmp/streamrec.log 2>&1;
```

- -y : Überschreiben der out.mp4 ohne Nachfragen

- -i : Input Datei oder URL
- -c:v copy: Video-Codec übernehmen (kopieren)
- -c:a copy: Audio-Codec übernehmen (kopieren)
- -t: 1 Stunde aufnehmen
- > /tmp/streamrec.log 2>&1;: nicht nach std-out sondern in Datei /tmp/streamrec.log schreiben
- -c copy: Video- und Audio- Codec übernehmen (kopieren)

Videostream segementieren à 1 Stunde

```
ffmpeg -y -i http://domain.tld/stream.m3u8 -c copy -f segment -segment_time 3600 out%03d.mp4 > /tmp/streamrec.log 2>&1;
```

- -i : Input Datei oder URL
- -y : Überschreiben der out.mp4 ohne Nachfragen
- -c copy: Video- und Audio- Codec übernehmen (kopieren)
- segment: Aufnahme segmentieren
- -segment_time 3600 : Jedes Segment in [s] = 3600s Länge = 1 Stunde
- > /tmp/streamrec.log 2>&1;: nicht nach std-out sondern in Datei /tmp/streamrec.log schreiben