FFMPEG 常用指令

时轲 2021.01.20

一 查看信息

-version

-formats

-demuxers

-muxers

-devices

-codecs

-decoders

-encoders

-bsfs

-protocols

-filters

-pix_fmts

-sample_fmts

-layouts

-colors

二 基本命令及参数

ffmpeg [global options] [[input options] -i input_file_url] [[output options]

output_file_url]

1. 主要参数

参数	**说明**
-f fmt	输入或输出的文件格式，通常不需要填写。
-i url	输入文件的地址。
-y （全局参数）	覆盖文件不询问。
-n （全局参数）	不覆盖文件。如果文件存在，立刻退出。
-c [: stream_specifier] codec (输入/输出，每个流)	选择一个编码器 (在输出文件前使用) 或解码器 (在输入文件之前使用）用于一个或者多个流。codec 是编码器/解码器的名称或 copy（仅输出）以不重新编码。如： Ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT
-codec[:stream_specifier]	同 -c
-t duration (输入/输出)	用于输入项时，表示输入 duration 时间的视频。 用于输出项时，表示当前到 duration 时间的视频。
-ss position (输入/输出)	一般在输入项使用，-i 之前。不是十分精确的定位时间。当 transcode 和-accurate_seek 时，一小部分时间段视频会被弃用。 当在输出项使用，解码但丢弃输入，直到时间戳位置。
-frames	停止在计数帧之后写入流。

[:stream_specifer] outputframecount（输出前的每一路流)	
-filter [:stream_specifer] filtergraph （输出前的每一路流)	用 filtergraph 指定创建过滤器视图，来过滤流。Filtergraph 是流的描述，必须具有相同类型的单个输入和单个输出。这些标签需要查询文档。

2. 视频参数

参数	**说明**
-vframe num (输出)	输出视频的帧数。是其过时的命令。
-r [:stream_specifer] fps (input/output)	设置帧率 Hz。新版本与-framerate 不同。
-s [:stream_specifer] size	设置窗口大小，格式 HxW。作为输入项时，是 video_size 专用选项。作为输出项时，是将缩放过滤器放在过滤器图的末尾。
-aspect [:stream_specifer] vaule	视频宽高比例。value 可以是浮点数字符串，也可以是 num：den。如 “4：3” 、“16：9” 、“1.333”、“1.777” 。 如果与 -vcodec 一起使用，会影响存储在容器级别的宽高，但是不影响存储在编码帧的宽高。
-vn （输出）	禁止视频录制。
-vcodec type （输出）	设置视频解码器。是 -codec：v 的别名。

-fv filtergraph （输出）	创建 filtergraph，过滤输出流。

3. **声音参数**

参数	**说明**
-aframes num （输出）	设置输出声音帧的数量。这是-frames:a 的别名。
-ar [stream_specifer] freq （输入/输出 每个流）	设置音频采样率。作为输出项，配置输出流的采样率。作为输入项，仅适用于原始音频采集设备和分路器，并映射到相应分路器的器件。
-ac [stream_specifer] channels （输入/输出 每个流）	设置音频通道数。作为输出项，配置输出流的声道数。作为输入项，仅适用于原始音频采集设备和分路器，并映射到相应分路器的器件。
-an (输出)	禁止录音。
-acodec type (输入/输出)	设置音频编解码器。这是-codecs：a 的别名。
-af filtergraph（输出）	创建 filtergraph，并用它来过滤流。

三 录制

1. 查看设备

```
ffmpeg -devices
```

```
ffmpeg -list_devices true -f dshow -i dummy
```

2. 录音

*

3. 录视频

*

四 分解和复用

1. 抽取音频：

```
ffmpeg -i input.mp4 -acodec copy -vn out.aac
```

acodec:音频编码器，copy 只拷贝，不做编解码。

vn:v 代表视频，n 代表 no，无视频。

2. 抽取视频：

```
ffmpeg -i input.mp4 -vcodec copy -an out.h264
```

Vcodec：视频编码器，copy 只做拷贝，不做编码。

an：a 代表音频，n 代表 no，无音频。

3. 文件转格式：

```
ffmpeg -i out.mp4 -vcodec copy -acodec copy out.flv
```

上命令是音频、视频直接 copy，只是将 mp4 封装改为 flv。

4. 音视频合并

```
ffmpeg -i out.h264 -i out.aac -vcodec copy -acodec copy out.mp4
```

五 处理原始数据集

1. 抽取 yuv 数据

```
ffmpeg -i input.mp4 -an -c:v rawvideo -pixel_format yuv420p out_2.yuv
ffplay -s wxh out.yuv
```

-c:v rawvideo 视频转原始数据

-pixel_format yuv420p 格式转为 420P

2. YUV 转 H264

```
ffmpeg -f rawvideo -pix_fmt yuv420p -s 320x240 -r 30 -i out.yuv -c:v libx264 -f rawvideo out.h264
```

3. 提取 PCM

```
ffmpeg -i out.mp4 -vn -ar 44100 -ac 2 -f s16le out.pcm
ffplay -ar 44100 -ac 2 -f s16le -i out.pcm
```

4. PCM 转 WAV

```
ffmpeg -f s16be -ar 8000 -ac 2 -acodec pcm_s16be -i input.raw output.wav
```

五 滤镜

使用 livavfilter 库中的过滤器处理原始音频和原始视频。多个过滤器可以组合成一个过滤器图形。FFmpeg 区分简单和复杂两类过滤器图形。

1．1 简单滤镜

简单的过滤图形，只有一个输入和输出，是相同的类型。通过在编码和解码之间，插入一个额外的步骤，滤镜来执行。

简单的 filtergraph 配置 per-stream-filter 选项，音频使用-af，视频使用-vf 的别名。步骤如下：

Input -> deinterlace -> scale -> output

注意，某些滤镜会更改帧属性，不更改帧内容。例如，fps 过滤器更改帧数，但不更改帧内容。例如 setpts 过滤器，只设置时间戳，不改变帧。

1.2 复杂滤镜

复杂过滤图表是不能被描述为一个流的线性处理链的过滤器图表。例如当图形有多个输入和/或输出，或者输出流类型与输入不同时，就是这个情况。如下：

input1 --------T

I T-------- output0

Input2 --------T -> comlpx filter graph ->

I L-------- output1

Input3 --------J

复杂的过滤图表使用-filter_complex 进行配置。此选项是全局性的，本质上不能与单个流或文件关联。-lavfi 等同与-filter_complex。

一个复杂过滤器图的例子，是覆盖过滤器。它有两个视频输入和一个视频输出，包含一个视频叠加到上面。它的音频对饮是 amix 过滤器。

2.1 添加水印

```
ffmpeg -i out.mp4 -vf "movie=logo.png,scale=64:48[watermask];[in][watermask] overlay=30:10 [out]" water.mp4
```

-vf 中的 movie 指定 logo 位置。

scale 指定 logo 大小。

overlay 指定 logo 摆放的位置。

2.2 删除水印

```
ffplay -i test.flv -vf delogo=x=806:y=20:w=70:h=80:show=1
```

用这个方法找到水印位置。

```
ffmpeg -i test.flv -vf delogo=x=806:y=20:w=70:h=80 output.flv
```

用这个方法删除水印。

3 视频缩小一倍

```
ffmpeg -i out.mp4 -vf scale=iw/2:-1 scale.mp4
```

-vf scale 指定简单过滤器 scale。

iw/2：-1 按整形取视频宽度，-1 表示高度随宽度一起变化。

4 视频裁剪

```
ffmpeg -i VR.mov -vf crop=in_w-200:in_h-200 -c:v libx264 -c:a copy -video_size 1280x720 vr_new.mp4
```

crop 格式： crop = out_w:out_h: x:y

out_w: 输出宽度。可以使用 in_w 表示输入视频的宽度。

out_h: 输出高度。可以使用 in_h 表示输入视频的高度。

x: X 坐标。

y: Y 坐标。

如果 x 和 y 设置 0，代表从左上角开始裁剪。否则从中心开始裁剪。

5 倍速播放

```
ffmpeg -i out.mp4 -filter_complex "[0:v]setpts=0.5*PTS[v];[0:a]atempo=2.0[a]" -map "[v]" -map "[a]" speed2.0.mp4
```

-filter_complex 复杂滤镜，[0:v]表示第一个（文件索引号是 0）文件作为视频输入。setpts=0.5*PTS 表示每帧视频 pts 时间戳乘 0.5。[v]表示输出的别名。

-map 可用于处理复杂输出。如可以将指定的多路流输出到一个输出文件，也可以输出到多个视频文件。“[v]” 复杂滤镜输出的的别名作为输出文件的一路流。上面的 map 用法是将复杂滤镜的音频、视频，输出到指定文件中。

6 对称视频

```
ffmpeg -i out.mp4 -filter_complex "[0:v]pad=w=2*iw[a];[0:v]hflip[b];[a][b]overlay=x=w" duicheng.mp4
```

Hflip 水平翻转。如果修改为垂直翻转，可以用 vfilp。

7.1 画中画

```
ffmpeg -i out.mp4 -i out1.mp4 -filter_complex "[1:v]scale=w=176:h=144:force_original_aspect_ratio=decrease[ckout];[0:v][ckout]overlay=x =W-w-10:y=0[out]" -map "[out]" -movflags faststart new.mp4
```

8 多路视频拼接

*

六 音视频拼接与裁剪

裁剪

ffmpeg -i out.mp4 -ss 00:00:00 -t 10 out1.mp4

-ss 指定裁剪的开始时间，精确到秒

-t 被剪后的时长

合并

input.txt 文件里：

file ‘video.mp4’

file ‘new.mp4’

file ‘duicheng.mp4’

```
ffmpeg -f concat -i inputs.txt -c copy output.mp4
```

HLS 切片

```
ffmpeg -i out.mp4 -c:v libx264 -c:a libfdk_aac -strict -2 -f hls out.m3u8
```

-strict -2 指明音频使用 AAC

-f hls 转 m3u8 格式

七 视频图片互转

视频转 JPEG

```
ffmpeg -i test.flv -r 1 -f image2 image-%3d.jpeg
```

视频转 GIF

```
ffmpeg -i out.mp4 -ss 00:00:00 -t 10 out.gif
```

图片转视频

```
ffmpeg -f image2 -i image-%3d.jpeg images.mp4
```

八 直播相关

推流

```
ffmpeg -re -i out.mp4 -c copy -f flv rtmp://server/live/streamName
```

拉流保存

```
ffmpeg -i rtmp://server/live/streamName -c copy dump.flv
```

转流

```
ffmpeg -i rtmp://server/live/originalStream -c:a copy -c:v copy -f flv rtmp://server/live/h264Stream
```

推相机实时流

```
ffmpeg -framerate 15 -f avfoundation -i "1" -s 1280x720 -c:v libx264 -f flv rtmp://localhost:1935/live/room
```

九 播放 YUV 数据

```
ffplay -pix_fmt nv12 -s 192x144 1.yuv
```

播放某个平面

```
ffplay -pix_fmt nv21 -s
```