# transcli命令生成

# 一、<source>标签

1. <audio stream="0" channel="original" decoder="auto" track="0"/>

这里的channel="original" 影响FFmpeg命令的参数-c:a pcm\_s16le -f s16le -ac 2 -ar 24000

track=1：输出保持和原始输入文件音轨数相同。

track=0：输出单音轨。

track>1：对应输出的音轨数。

1. <video stream="0" rectangle="{0,0,480,360}" crop="calc" threshold="26" decoder="auto"/>

crop设置为auto时，调用FFmpeg自动检测黑边，threshold设置黑边判断的灰度阈值。

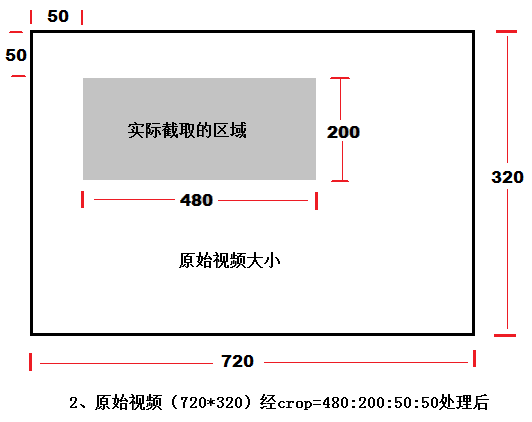
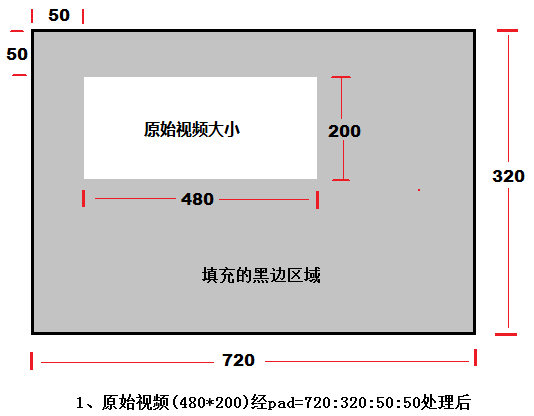
./codecs/ffmpeg.exe -i "\*\*\*.mp4" -an -t 200 -v panic -c:v rawvideo -vf "select=isnan(prev\_selected\_t)+gte(t-prev\_selected\_t\,5),cropdetect=26:4" -f null -y NUL

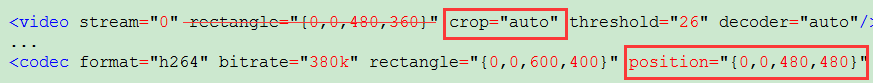
FFmpeg的Filter：

pad=720:320:50:50，四个参数分别表示width:height:x:y

crop=480:200:50:50，四个参数分别表示width:height:x:y

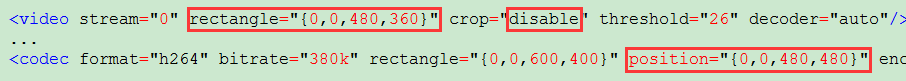
分别用下面的2张图表示：





当crop是auto时，rectangle选项的值失效，会自动检测黑边，并缩放到position的大小。

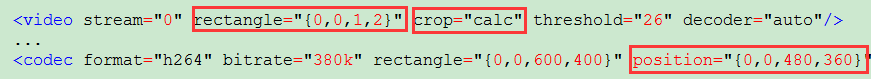
比如原始视频尺寸是1920\*1080，黑边在左右两边，宽度为200，那么生成FFmpeg的filter命令是：-vf crop=1520:1080:200:0,scale=480:480。



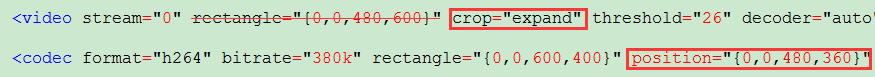
当crop是disable时，先根据rectangle的值crop，然后根据postion的值scale。

生成FFmpeg的filter命令是：-vf crop=480:360:0:0,scale=480:480。

crop是manual和disable等同。



当crop是calc时，先根据rectangle的宽高比作为crop的宽高比，（rectangle=”{0,0,1,2}”和rectangle=”{0,0,360,720}”等同），然后以position作为scale的大小，那么生成的FFmpeg的filter命令是：-vf crop=360:720:460:0,scale=480:360。



当crop是expand时，rectangle参数的值没用。先根据目标尺寸（比如上面的480\*360），以及原视频宽高比（比如1280:720=480:270），计算出黑边的大小，这里黑边加在左右两边，宽度为45。

expand和前面的参数不一样的是：它先进行scale再进行pad。

生成FFmpeg的filter命令是：-vf scale=480:270,pad=480:360:0:44。

1. <clipmode>

当设置为1时,会执行CTransWorker::processLosslessClip函数，这样下面的<clip>标签才起作用，此函数用MP4BOX命令将源splitx，并存储为tmp文件：

./tools/MP4Box.exe "E:\m3u8\Introduction\_to\_Deep\_Learning\_with\_Python.mp4" -splitx 2.00:70.00 -out "E:\transcli\bin\x64\temp\5344\_3110129\_1\_0.mp4"，…

然后将分割的多个tmp文件合并：

./tools/MP4Box.exe -cat "E:\transcli\bin\x64\temp\5344\_3110129\_1\_0.mp4" -cat "E:\transcli\bin\x64\temp\5344\_3177818\_1\_1.mp4" -new "./Introduction\_to\_Deep\_Learning\_with\_Python.flv"

# 二、<target>标签

1. <delogo>

模糊功能。支持绝对坐标和相对坐标:

<pos position="{10,10,90,90}" start="00:00:10.0" end="00:00:40" />

<pos position="{100,100,220,220}" start="00:00:30.0" end="00:1:20" />

FFmpeg生成的命令: -filter:v delogo=10:10:80:80:2:0:10:40,delogo=100:100:120:120:2:0:30:80

在FFmpeg中新增了start和end两个参数，用来限定delogo的处理时段。

1. <audio disable="false">

<codec format="aac" bitrate="48k" samplerate="24000" ~~timeshift="1.5s"~~ brdown="true"/>

这里的bitrate=”48k”，设置conf.target.acodec.bitrate的值，最终会在CAudioEncoder类里面调用Initialize函数初始化，对应的是音频编码器参数，不影响FFmpeg参数。

samplerate="24000"，对应FFmpeg解码成PCM的采样率，影响FFmpeg的参数：

-c:a pcm\_s16le -f s16le -ac 2 -ar 24000

brdown="false"降码率功能的设置详见CTransWorker::setAudioEncAttrib函数。

1. <volgain mode="auto" value="0.0"/>

FFmpeg生成的命令：-filter:a aresample=async=1:first\_pts=0:min\_comp=0.05:min\_hard\_comp=0.15,volume=-5.54dB

1. <videoenhance>1</videoenhance>

调用画质增强算法CVideoEnhancer，参照CVideoEncoder::FilterFrame函数。

1. <profile>high</profile>、<rcmode>4</rcmode>、<ref>4</ref>、<fps>0</fps>、<dar>0.0</dar>、<keyint>5</keyint>、<bframes>4</bframes>

"videoenc.x264.profile"、"overall.video.mode"、"videoenc.x264.frameref"、"videoenc.x264.keyint"、"videofilter.frame.fps"、"overall.video.arNum"

"videoenc.x264.bframes"

在CVideoEncoder类的Initialize函数处初始化。

在CTransWorkerSeperate::setVideoPref中调用pEncoder->SetVideoInfo设置帧率和宽高比。

1. <codec format="h264" bitrate="380k" rectangle="{0,0,600,400}" position="{0,0,480,320}" encode="true"/>

这里的position的作用是放缩(scale)，比如position=”-100,-100,480,320”，出来的视频尺寸是580\*420。具体加黑边与否和<source>标签的<video>的crop参数有关。

FFmpeg生成的命令：-filter:v yadif=0:0,delogo=1

00:100:120:120:2:0:30:80,scale=580:420,hqdn3d=4:3

rectangle="{0,0,600,400}"标签用处不大，主要是在scale值均为负数的时候做替代用的。

encode=”false”表示不进行视频编码。

1. <subtitle id="-1" timeshift="0s" extract="-2"/>

参照CTransWorkerSeperate::adjustSubtitleAttrib函数。

1. <logo>

标签的解析参考encoderCommon.cpp的parseWaterMarkInfo函数，使用watermark工程来添加logo。各个属性的作用参考《转码内核系统总结（郑建峰）》。

1. <filter>

标签中，<color>和<noise>无用。

<deinterlace mode="1" algorithm="0"/>

当mode=”1”时起作用：

-filter:v yadif=0:0,delogo=10:10:80:80:2:0:10:40

<denoise mode="2" luma="4" chroma="3" temporal="6"/>

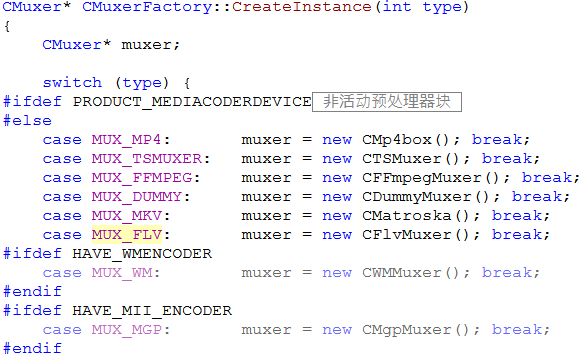
当mode=”2”时起作用，

-filter:v hqdn3d=4:3,pad=480:320:0:24

CDecoderFFMpeg::GenVideoFilterOptions

1. <mux format="flv" disable="false"/>

参考CCliHelperPPLive::AdjustPreset函数，根据format指定的类型选取对应的Muxer。



1. <imagetail image="E:\2015-1-2\11363074966.jpg" folder="E:\2015-1-2\" duration="2" cropmode="0"/>

<enum>Crop to fit</enum>

<enum>Pad to fit</enum>



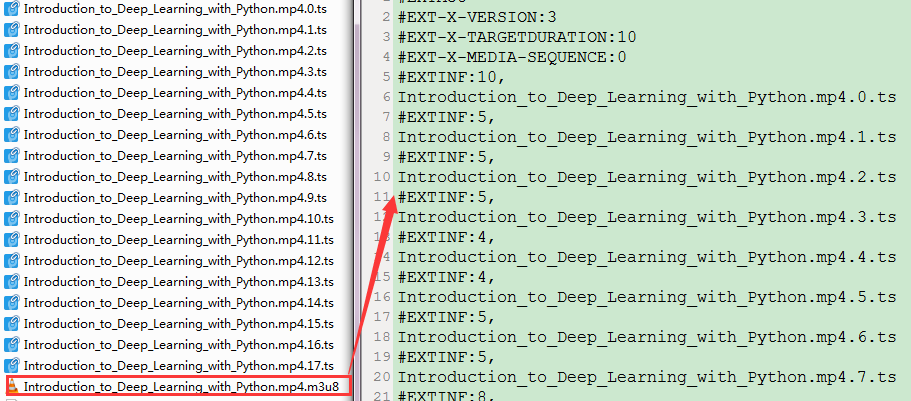
1. <playlist type=".m3u8" name="pl" postfix=".mp4" live="false"/>

参考CPlaylistGenerator::GenPlaylist函数，<playlist>标签的作用是生成播放列表文件。

1. <hls dur="5" listsize="100000" postfix=".mp4" startindex="1"/>

mux的过程参见muxers.cpp文件对应的mux功能。

./codecs/ffmpeg.exe -i "E:\transcli\bin\x64\temp\9484\_112632285\_1\_0.mp4" -v error -map 0 -vcodec copy -acodec copy -vbsf h264\_mp4toannexb -f hls -hls\_time 5 -hls\_list\_size 100000 -start\_number 1 -y "./Introduction\_to\_Deep\_Learning\_with\_Python.mp4..m3u8"



1. <thumbnail start="6" end="0" interval="6" row="10" col="10" width="180" height="0" type="2" count="0" quality="85" stitch="true" pack="true" postfix="\_shot"/>

使用watermark进行截图。

设置过程见encoderCommon.cpp文件的parseThumbnailInfo和parseThumbnailInfo1函数。初始化过程CVideoEncoder::InitWaterMark。

1. <avalign blankaudio="true" blankvideo="true" padding="true"/>

插入空白音视频帧，padding="true"时，前面的设置才有用。

1. <x264 option="ref=4:bframes=4:me=umh:subme=9:merange=24:rc-lookahead=60:trellis=2:b-adapt=2:b-pyramid=2:weightb=true:weightp=2:aq-mode=1:mbtree=true:nal-hrd=none:scenecut=40:psnr=true:ssim=true">

<x265 option="ref=1:bframes=4:me=1:subme=2:sao=true:amp=true:rect=true:b-adapt=2:wpp=true:ctu=64:open-gop=false:rd=3:lft=true:b-pyramid=true:weightp=false:tu-intra-depth=4:tu-inter-depth=4:b-intra=false:psnr=false:ssim=false" />

在CX264Encode::Initialize和CX265Encode::Initialize

函数分别初始化X264和X265编码器的参数。

<zone start="00:00:02.0" end="00:01:10.0" ratio="0.6"/>

这里最多只能有2个zone，主要用来设置m\_x264Param.rc.psz\_zones

的值，参见CX264Encode::Initialize函数。

1. <mp4 option="-brand=3grp/3gp6/3gp4,-version=512,-hint=true,-sbr=0/1/2" />

-brand: sets the major brand of a file. Brands are used to identify the most common usage of a file (MPEG-4 presentation, 3GP movie, etc...)

-hint: for RTP/RTSP

-sbr: forces importing the AAC-ADTS file as AAC SBR (aka HE-AAC, aka aacPlus) with backward compatible signaling (eg non SBR aware decoders should play the file).

-sbrx: forces importing the AAC-ADTS file as AAC SBR (aka HE-AAC, aka aacPlus) with non-backward compatible signaling (eg non SBR aware decoders should NOT play the file).