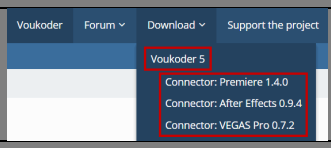
从 x265 教程综合版总结出的参数配置. 目的是把参数直接贴到软件里用. 要求 x265 v3.5+69 或 v3.6

LigH	.hevc GCC10 [单文件 8-10-12bit] 附 x86, Windows XP x86 版 附 libx265.dll
Rigaya	.hevc GCC 9.3 [8-10-12bit] 附 x86 版
Patman	.hevc GCC 11+MSVC1925 [8-10-12bit]
ShortKatz	arm64~64e 加 x86 版 [?] 需 macOS 运行编译命令文件 ?
DJATOM-aMod	opt-Intel 架构与 zen1~2 优化 [10bit], opt-znver3 代表 zen3 优化 [10-12bit] GCC 10.2.1+GCC10.3
MeteorRain-yuuki	lsmash.mkv/mp4 或.hevc [能封装, 但传说 lavf 不如 pipe 可靠] GCC 9.3+ICC 1900+MSVC 1916 [8][10][12bit]+[8-10-12bit]

ffmpeg~ffprobe 顶级开源多系统多媒体 CLI 处理~检测工具

mpv 支持便携的开源多系统现代视频播放器。见安装与配置教程

Voukoder 开源 Premiere、Vegas、Aftereffects 压制导出插件，分为 Voukoder 和 V-Connector 两部分



MediaInfo 开源 GUI 媒体元数据/视音频格式读取器，用于快速查看完整元数据

程序下载与命令行用法

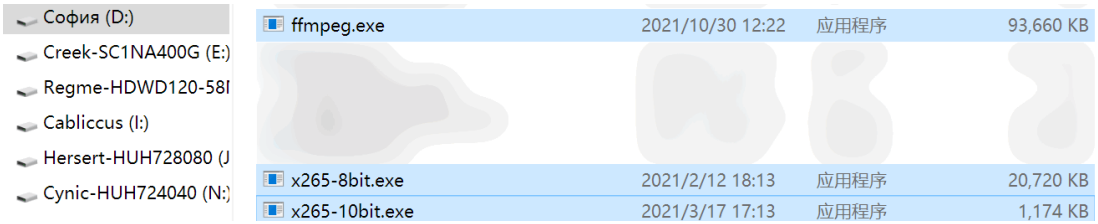
1. 于上表下载 ffmpeg, ffprobe/MediaInfo, x265 并记住路径。如果使用 ffmpeg 内置的 libx265 动态链接库，则可以不下载 x265，但要求是须确保 ffmpeg 的版本为最新

选择架构优化?

尽管 HEVC 标准与 x265 编码器早在 2013 年宣布立项（Intel 推出酷睿 i7-4xxx），但代码编译器和 x265 的源代码一直在改进。因此选择下载独立的编码器程序有速度更快的 Mod 版可选，但前提是操作（查找报错）难度略高于直接调用 libx265 库，并且应选择与当前自用 CPU 架构优化一致的版本（否则不兼容或性能下降）

选择编码器位深?

有同时含 8-10-12bit 的 x265.exe 也有已区分为 x265-8bit.exe, x265-10bit.exe 的版本。一般来说，单个含多位深的程序会方便 ffmpeg/AVS/VS 通过 yuv for mpeg 管道传递位深信息，从而使 x265.exe 自动设置位深。



此处置于 Windows 系统 D 盘根目录下，因此路径为 D:\

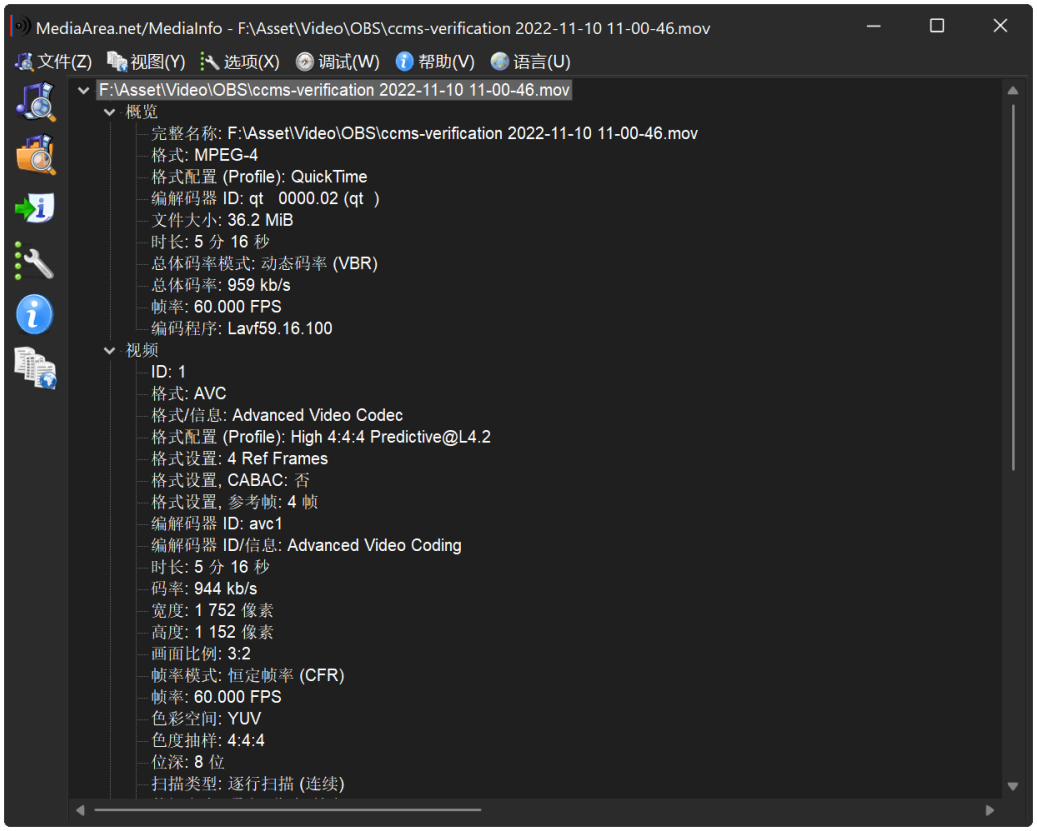
2. CMD/PowerShell/Bash/Terminal 下分别输入 ffmpeg、x265 的路径并回车，即可确认两点：

1. 路径的拼写：直接选择并复制以配置命令行
2. 程序版本：越新越好，编码器的大版本更新有 Bug 修复与性能提升，其它软件的更新有新格式兼容，翻译改进等体验提升

图中检查 D:\x265-10bit.exe -V 和 D:\ffmpeg.exe 确认程序存在

3. 使用 MediaInfo（图形界面）或 ffprobe（命令行界面）获取视音频格式细节：

双击打开 MediaInfo.exe 并将视频文件拖放到图形界面中，菜单栏的视图/View 中可以选择树状图（需要精确小数点可以选 JSON），可以选择菜单栏（Language）可选简体中文，即可得到视频信息



假设 ffprobe 位于 D 盘根目录下，则命令为 `D:\ffprobe.exe -i ".\视频.mp4" -select_streams v:0 -v error -show_streams -show_frames -read_intervals "%+#1" -show_entries frame=top_field_first:stream=codec_long_name,width,coded_width,height,coded_height,pix_fmt,color_range,field_order,r_frame_rate,avg_frame_rate,nb_frames -of ini`

```
[frames.frame.0]
top_field_first=0           分场-是否上场优先(1/0)

[streams.stream.0]
codec_long_name=H.264       源视频格式
width=1920                  宽
height=1080                 高
coded_width=1920            编码宽 - 若!=宽则代表横向长方形像素源
coded_height=1088           编码高 - 若!=高则代表纵向长方形像素源
pix_fmt=yuv420p             色彩空间
color_range=tv              色彩范围(pc=full=0~255/tv=limited=16~235)
field_order=progressive     逐行/分场(progressive/interlaced/unknown)
r_frame_rate=24000/1001     帧率
avg_frame_rate=24000/1001   编码帧率 - 若!=帧率则代表可变帧率vfr
nb_frames=20238             总帧数   - 根据压缩速度fps推测完成时间
```

图中为 ffprobe 输出，得到了编码格式名，视频帧大小和实际大小，色彩空间格式与范围，视频帧率，平均帧率，总帧数。依此可以判断：

交错/分行扫描?

这类视频并非使用帧率，而是“场率”为画面基础。有上场优先、下场优先；搭配原生帧率，有 NTSC 电

视标准丢帧，有 PAL 电视标准丢帧，有假丢帧等多种“相信后人智慧”的兼容性需求。需要进一步根据帧率，如果一定要处理成现代的逐行扫描格式，可以参考这篇教程

可变帧率?

帧率模式显示 VFR 或 `avg_frame_rate` 异于 `r_frame_rate`，此时需要确保视频在剪辑前被重编码（渲染为恒定帧率 CFR），以保证剪辑软件/工具链上全部视频滤镜的兼容性，以及避免音画不同步。如 ffmpeg 可以添加 `-vsync cfr` 转换为恒定帧率 Constant Frame Rate

音频格式兼容?

如果要更换封装文件，则需要确认其中的音频流是否兼容到目标格式，如果不兼容则需要转码。格式兼容列表可见于维基百科：Comparison of video container formats - Video coding formats support。兼容性不错的 QAAC 音频编码可以参考 这篇教程 或 Github

长方形像素?

视频大小和实际编码大小不同，代表了日本电视台缩宽，旧版优酷缩高的古代视频压缩手段. 能换源则尽可能换

压制用时?

时长秒数= 总帧数÷压缩速度 fps。通过系统查看封装文件属性，或 MediaInfo、ffprobe 得到视频时长，即可在视频编码器不预估完成时间（如某些情况下未提供总帧数信息）的情况下手动计算

ffmpeg 参数：-pix_fmt 与 -strict

ffmpeg 能够像 MediaInfo 一样自动检测元数据并设定 `-pix_fmt` 参数，但有时源视频的元数据中会缺

少这些信息（MediaInfo 同样看不到），便要手动设定。需要确认时可以使用 ffprobe 查找，有: yuv420p, yuv422p, yuv444p, yuv420p10le, yuv420p12le, yuv422p10le, yuv422p12le, yuv444p10le, yuv444p12le, yuv444p10le, yuv444p12le, gray, gray10le, gray12le, nv12, nv16

在使用管道/pipe 参数时，超过 8bit 的 YUV for MPEG 流并不合规，因此需要额外提供 -strict 参数解除合规性限制，而在使用 ffmpeg 内置库时则不会用到管道，故同时无需指定 -pix_fmt 与 -strict。

```
[yuv4mpegpipe @ 0000018dde853540] 'yuv420p10le' is not an official yuv4mpegpipe pixel format. Use '-strict -1' to encode to
 this pixel format.
[out#0/yuv4mpegpipe @ 0000018dde8d9e40] Could not write header (incorrect codec parameters ?): Invalid argument
[vf#0:0 @ 0000018dde85e640] Error sending frames to consumers: Invalid argument
[vf#0:0 @ 0000018dde85e640] Task finished with error code: -22 (Invalid argument)
[vf#0:0 @ 0000018dde85e640] Terminating thread with return code -22 (Invalid argument)
[out#0/yuv4mpegpipe @ 0000018dde8d9e40] Nothing was written into output file, because at least one of its streams received
no packets.
```

一般命令行参数格式，ffmpeg 常用操作，命令行操作技巧

见 x265 教程完整版

x265 HDR 设置参数:

x265 HDR 标识

--master-display <手动告知播放器拿什么色彩空间解码

DCI-P3: G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(maxCLL × 10000,1)

bt709: G(15000,30000)B(7500,3000)R(32000,16500)WP(15635,16450)L(maxCLL × 10000,1)

bt2020: G(8500,39850)B(6550,2300)R(35400,14600)WP(15635,16450)L(maxCLL × 10000,1)

- 找到 HDR 元数据中的色彩范围，确认用以下哪个色彩空间后填上参数
- L 的值没有标准，每个 HDR 视频元数据里可能都不一样

DCI-P3: G(x0.265, y0.690), B(x0.150, y0.060), R(x0.680, y0.320), WP(x0.3127, y0.329)

bt709: G(x0.30, y0.60), B(x0.150, y0.060), R(x0.640, y0.330), WP(x0.3127,y0.329)

bt2020: G(x0.170, y0.797), B(x0.131, y0.046), R(x0.708, y0.292), WP(x0.3127,y0.329)>

--max-cll <maxCLL,maxFALL>最大,平均光强度, MediaInfo 查不出来就不用填

--colormatrix <照源, 例: gbr bt709 fcc bt470bg smpte170m YCgCo bt2020nc bt2020c smpte2084 ictcp>

--transfer <照源, 例: gbr bt709 fcc bt470bg smpte170m YCgCo bt2020nc bt2020c smpte2084 ictcp>

杜比视界 dolby vision/DV 有 DV-MEL (BL+RPU)和 DV-FEL (BL+EL+RPU)两种带 RPU 的格式, x265 支持共 3 种样式/profile 的 DV-MEL

样式	编码	BL:EL 分辨率	x265 支持	伽马	色彩空间
4	10bit hevc	1:1/4		SDR	YCbCr
5		仅 BL (DV-MEL)	✓		ICtCp

7		4K=1:1/4; 1920x1080=1:1		UHD 蓝光	YCbCr
8.1		仅 BL (DV-MEL)	✓	HDR10	
8.2			✓	SDR	
8.4				HLG	
9	8bit avc	仅 BL (DV-MEL)		SDR	YCbCr

--dolby-vision-profile<选择 5/8.1 (HDR10)/8.2 (SDR)>8.1 需要写 master-display 和 hdr10-opt
--dolby-vision-rpu<路径>导入 rpu 二进制文件(.bin)用

目标色深

ffmpeg 有能够发送视频帧元数据的 yuv-for-mpeg pipe（管道），和只发送视频帧的 raw pipe，而管道下游的 x265.exe 根据版本和 mod 不同，不一定能够识别 yuv-for-mpeg 的元数据；同时，x265 的位深设定是仅 CLI（CLI-ONLY）参数，例如在 ffmpeg 的 libx265 中，位深由 ffmpeg 自身指定。因此，本教程中 ffmpeg pipe 的参数会要求 -D 选项指定视频色深，而 ffmpeg libx265 则没有。

x265 管道输入参数变更

x265 v4.0 版中引入了 Multiview Encoding（多视角输入编码），因此 ffmpeg pipe 的格式从自 x264 以来的"-"变更为"--input -"参数

选择规格 Profile，级别 Level

根据视频位深选择规格 Profile，分辨率和帧率选择级别 Level，最后细分到档次 Tier（Main/High）

级别 Level	最大亮度像素流量 (Luma Samples)	最大亮度平面面积 (Luma Size)	8-10bit 最大码率	12bit 最大码率	4:4:4 12bit 最大码率	最高分辨率@帧率
1	552,960	36,864	Main: 128 Kbps High: –	Main: 192 Kbps High: –	Main: 384 Kbps High: –	176 × 144@15 fps
2	3,686,400	122,880	Main: 1500 Kbps High: –	Main: 2250 Kbps High: –	Main: 4500 Kbps High: –	352 × 288@30 fps
2.1	7,372,800	245,760	Main: 3000 Kbps High: –	Main: 4500 Kbps High: –	Main: 9000 Kbps High: –	640 × 360@30 fps
3	16,588,800	552,960	Main: 6000 Kbps High: –	Main: 9000 Kbps High: –	Main: 18 Mbps High: –	960 × 540@30 fps

3.1	33,177,600	983,040	**Main:** 10 Mbps **High:** –	**Main:** 15 Mbps **High:** –	**Main:** 30 Mbps **High:** –	1280 × 720@33.7 fps
4	66,846,720	2,228,224	**Main:** 12 Mbps **High:** 30 Mbps	**Main:** 18 Mbps **High:** 45 Mbps	**Main:** 36 Mbps **High:** 90 Mbps	1280 × 720@68 1920 × 1080@32 fps
4.1	133,693,440	2,228,224	**Main:** 20 Mbps **High:** 50 Mbps	**Main:** 30 Mbps **High:** 75 Mbps	**Main:** 60 Mbps **High:** 150 Mbps	1920 × 1080@64 fps 2048 × 1080@60 fps
5	267,386,880	8,912,896	**Main:** 25 Mbps **High:** 100 Mbps	**Main:** 37.5 Mbps **High:** 150 Mbps	**Main:** 75 Mbps **High:** 300 Mbps	3840 × 2160@32 fps 4096 × 2160@30 fps
5.1	534,773,760	8,912,896	**Main:** 40 Mbps **High:** 160 Mbps	**Main:** 60 Mbps **High:** 240 Mbps	**Main:** 120 Mbps **High:** 480 Mbps	3840 × 2160@64 fps 4096 × 2160@60 fps
5.2	1,069,547,520	8,912,896	**Main:** 60 Mbps **High:** 240 Mbps	**Main:** 90 Mbps **High:** 360 Mbps	**Main:** 180 Mbps **High:** 720 Mbps	3840 × 2160@128 fps 4096 × 2160@120 fps
6	1,069,547,520	35,651,584	**Main:** 60 Mbps **High:** 240 Mbps	**Main:** 90 Mbps **High:** 360 Mbps	**Main:** 180 Mbps **High:** 720 Mbps	7680 × 4320@32 fps 8192 × 4320@30 fps
6.1	2,139,095,040	35,651,584	**Main:** 120 Mbps **High:** 480 Mbps	**Main:** 180 Mbps **High:** 720 Mbps	**Main:** 360 Mbps **High:** 1440 Mbps	7680 × 4320@64 fps 8192 × 4320@60 fps
6.2	4,278,190,080	35,651,584	**Main:** 240 Mbps **High:** 800 Mbps	**Main:** 360 Mbps **High:** 1200 Mbps	**Main:** 720 Mbps **High:** 2400 Mbps	7680 × 4320@128 fps 8192 × 4320@120 fps

为了方便使用，本教程设定为一直打开 --high-tier 选项，详见 x265 教程完整版

有兼容性问题的参数

ffmpeg

-strict unofficial / -1（允许超过 8bit 的 y4m 以及其他格式，提高管道/pipe 兼容性，但不能提早发现一些下游兼容性问题，在使用 ffmpeg 内置库时）

-hwaccel auto（硬件解码，由于部分硬件厂商的解码实现较差，所以可能会花屏，同时可能与 analyze-src-pics 有冲突）

x265

--allow-non-conformance（允许不合规参数以提高压缩率和画质，但可能会遇到播放、剪辑兼容问题）

非必要参数

ffmpeg

-hide_banner（解决命令行窗口被版权协议等信息填满的问题）

-pix_fmt, -strict（见上方：-pix_fmt 与 -strict 参数）

x265

--hash（每帧校验，纠错的效果和没有纠错差不多）

--radl（支持 I 帧前放置 RADL 帧，会改动 GOP 结构，虽然播放没问题，但分段拼合时兼容性差）

--mcstf（不稳定，可能会导致压制失败）

--rd 5（大多情况下 3 就够了，但因压力高于 Cinebench R23，所以可用于测试 CPU 超频稳定性）

--qp-adaptation-range（有人在 x265 v4.1 遇到了编码后视频帧播放一小段后冻结的问题，但这可能是硬解错误）

客观画质指标跑分

进行画质跑分的原因有二，差或面积太小的显示器可能会隐藏画面瑕疵，而好的显示器太贵，与升级 CPU 等设备的预算冲突；发布画质敏感内容时，需要数据来避免陷入老王卖瓜，自卖自夸的不根之论中。如果源需要经过滤镜处理，那么操作上的确会多出“导出无损渲染结果”的一步，以便测试。关于更详细的说明见 AV1 教程。

块大小感知加权峰值信噪比 XPSNR

速度极快，ffmpeg 内置，倾向于计算源与压缩结果之间的差异，注重暂停画质。推荐每完成一次编码后就运行，以进行快速自查，分数取值为 dB

- 视觉无损：大于等于 45dB
- 优秀：大于等于 42dB
- 良好：大于等于 38dB
- 达标：大于等于 32dB

[普通版本]

```
ffmpeg -i .\源视频.mkv -i .\压缩后.ivf -lavfi xpsnr="stats_file=-" -f null -
```

[time base 对齐版本，应对 DTS 未单调增加错误]

```
ffmpeg.exe -i ".\压缩前.mkv" -i ".\压缩后.mkv" -lavfi "[0:v]setpts=N*(1/时间基)[src];
[1:v]setpts=N*(1/时间基)[enc]; [src][enc]xpsnr=stats_file=-" -f null -
```

多方法融合 VMAF

速度快，ffmpeg 内置，倾向于检查视觉模型观感体验，而非压缩前后差异，注重播放画质。分数取值范围为 0~100，可以同时使用 VMAF 4K（远距离，客厅或影院观看，对高频细节敏感）和 VMAF（显示器，对二次编码、重采样、染色等失真敏感）两种模型一并计算。

[普通版本]
```
ffmpeg -i .\源视频.mkv -i .\压缩后.ivf -lavfi
libvmaf="model=version=vmaf_v0.6.1\\:name=vmaf_1080p|version=vmaf_4k_v0.6.1\\:name=vmaf_4k"
-f null -
```
[time base 对齐版本，应对 DTS 未单调增加错误]
```
ffmpeg.exe -i ".\压缩前.mkv" -i ".\压缩后.mkv" -lavfi "[0:v]setpts=N*(时间基)[src];
[1:v]setpts=N*(时间基)[enc];
[src][enc]libvmaf=model=version=vmaf_v0.6.1\\:name=vmaf_1080p|version=vmaf_4k_v0.6.1\\:name
=vmaf_4k" -f null -
```

时间基获取与对齐

由于不同编码其与封装格式所产生的 time base 不同，直接对比可能会出现“三帧烂一帧”的对齐错误，因此有必要使用 ffprobe（MediaInfo 看不到）对比编码前后的两处 time_base 值，然后将两者的积作为画质指标的时间基使用。同时，这种对比不支持时间基变化的可变帧率（VFR）视频。

```
ffprobe.exe -v error -select_streams v:0 -show_entries stream=time_base -of
default=noprint_wrappers=1:nokey=0 ".\视频.mkv"
```

例如，若从源视频得到 time base 1/1000，编码后则是 time base 1/24（两者帧率同为 24fps），则对齐的时间基为二者积（或最小公倍数）：1/24000，因此设为：

```
[0:v]setpts=N*(1/24000)[src]; [1:v]setpts=N*(1/24000)[enc];
```

画质得分优劣

- 视觉无损： XPSNR 大于等于 45dB，VMAF 大于等于 95
- 优秀： XPSNR 大于等于 42dB，VMAF 大于等于 90
- 良好： XPSNR 大于等于 38dB，VMAF 大于等于 80

- 达标:　　XPSNR 大于等于 32dB，VMAF 大于等于 70

结果示例（量化强度中等）

结果 1：尽管压缩结果与源的差距（失真损失）极大，但由于视频内容变化剧烈，导致播放时看不出毛病，但仍然应该降低量化强度以提高暂停画质

```
XPSNR average, 6314 frames  y: 20.9812  u: 38.0531  v: 35.0405 (minimum: 20.9812)
VMAF 4k: 98.125428, VMAF 1080p: 96.795521
```

结果 2：无明显问题，或可略微降低量化，将 VMAF 4k 分数提高到 90，XPSNR Y 提至 42

```
XPSNR average, 36996 frames  y: 39.3288  u: 42.4070  v: 42.9840 (minimum: 39.3288)
VMAF 4k: 88.251216, VMAF 1080p: 82.140527
```

结果 3：XPSNR 的分数可以，但两个 VMAF 模型之间得分的差距较大，这是因为画面中有“二次编码”，“上采样”，“块失真”，“色带”等痕迹（此处是含一些低分辨率素材渲染的 3D 动画）。而 XPSNR 在 U、V 得分高的原因单纯是因为源视频的色彩较简单，容易压缩。可以尝试降低量化强度（或同时增加色度平面的量化强度以平衡文件体积），让 VMAF 4k 达到 90 分

```
XPSNR average, 15691 frames  y: 32.9889  u: 45.2554  v: 44.5165 (minimum: 32.9889)
VMAF 4K: 83.837285, VMAF 1080p: 77.317822
```

结果 4：XPSNR 的分数可以，但两个 VMAF 都给出偏低的分，可以考虑降低量化强度（或同时增加色度平面的量化强度以平衡文件体积），让 VMAF 4k 达到 80 分

```
XPSNR average, 301 frames  y: 32.2371  u: 40.8458  v: 42.8932 (minimum: 32.2371)
VMAF 4k: 77.517816, VMAF 1080p: 68.330515
```

结果 5：无明显问题，可以不改

```
XPSNR average, 1199 frames  y: 39.8871  u: 42.1991  v: 42.3623 (minimum: 39.8871)
VMAF 4k: 91.383935, VMAF 1080p: 85.874477
```

通用·简单

~~去掉所有自定义项目~~填 Profile，Level，方便急用且速度仅比 preset slow 慢几 fps

兼容性 --profile<8/10/12bit: main/main10/main12, YUV4:2:2: main422-10/main422-12, YUV4:4:4: main444-8/main444-10/main444-12> --high-tier

预设-转场 --preset slow

动态搜索 --me umh --subme 5 --merange 48 --weightb

自适应量化 --aq-mode 4

帧控 --bframes 5 --ref 3

多处理器分配 --pools ,,,, (举例-,+表示该电脑有两个 CPU 节点，用第二个. 同时占用多个会造成严重的内存延迟)

其它 去黑边加速: --display-window <整数"←,↑,→,↓"像素>, ≥22 核 cpu 优化: --pme, 分场视频: --field, 抖动高质量降色深: --dither, 开始; 结束帧: --seek; --frames, crf/abr 缓解噪点影响: --rc-grain

目标色彩空间 [ffmpeg] -pix_fmt yuv420p / yuv422p / yuv444p / yuv420p10 / yuv422p10 / yuv444p10...

α——(ffmpeg pipe) x265 CLI 命令

- fmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -pix_fmt ◯ -strict -1 - | x265.exe --profile ◯ --high-tier --preset slow --me umh --subme 5 --merange 48 --weightb --aq-mode 4 --bframes 5 --ref 3--y4m --input - --output ".\输出.hevc"

β——ffmpeg libx265 CLI，拷贝音频并封装为 mp4

- ffmpeg.exe -y -i ".\导入.mp4" -c:v libx265 -profile:v ◯ -x265-params "high-tier=1:preset=slow:me=umh:subme=5:merange=48:weightb=1:bframes=5:ref=3" -fps_mode passthrough -c:a copy ".\输出.mp4"

通用·标准

含大量自定义项目，可以配出高压或高速参数

兼容性 --profile<8/10/12bit: main/main10/main12, YUV4:2:2: main422-10/main422-12, YUV4:4:4: main444-8/main444-10/main444-12> --high-tier

分块-变换 --tu-intra-depth 3 --tu-inter-depth 3 --limit-tu 1 --rdpenalty 1 --rect

动搜-补偿 --me umh --subme <24fps: 3, 48fps: 4, 60fps: 5, 100fps: 6> --merange <1920:1080: 48, 2560:1440: 52, 3840:2160: 56> --weightb

溯块-帧控 --ref 3 --max-merge <快: 2, 中: 3, 慢: 5> --early-skip --no-open-gop --min-keyint 5 --fades --bframes 8 --b-adapt 2 <锐利线条: --pbratio 1.2>

帧内编码 <快: --fast-intra / 中: 不填 / 慢: --b-intra / 极慢且有兼容性问题: --constrained-intra>

量化 --crf <超清: 18~20, 高清: 19~22> --crqpoffs -3 --cbqpoffs -1

率失优量化 --rdoq-level <快: 1，很慢: 2>

自适应量化 <动漫源改--hevc-aq, 关 aq-mode > --aq-mode 4 --aq-strength <多面: 0.8, 多线: 1>

模式决策 --rd 3 --limit-modes --limit-refs 1 --rskip <快: 2, 中: 1, 慢: 0> --rc-lookahead <3×帧率, 大于 bframes> --rect <很慢: --amp>

率失真优化 --psy-rd <录像: 1.6，动画: 0.6, ctu64: +0.6, ctu16: -0.6> --splitrd-skip

去块-取迁 --limit-sao --sao-non-deblock --deblock 0:-1

目标色深 -D 8/10/12 <单程序兼容多色深时须手动指定, 默认 8bit, 低勿转高, 高转低开 --dither>

多处理器分配 --pools ,,,, (举例-,+表示该电脑有两个 CPU 节点, 用第二个. 同时占用多个会造成严重的内存延迟)

其它 **去黑边加速:** --display-window <整数"←, ↑ ,→, ↓ "像素>, **≥22 核 cpu 优化:** --pme, **分场视频:** --field, **抖动高质量降色深:** --dither, **开始; 结束帧:** --seek; --frames, **crf/abr 缓解噪点影响:** --rc-grain

α——(ffmpeg pipe) x265 CLI 命令-共 11+2 个自定域

- ffmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -strict -1 - | x265.exe --profile ◯ --high-tier --ctu ◯ --min-cu-size 16 --tu-intra-depth 3 --tu-inter-depth 3 --limit-tu 1 --rdpenalty 1 --me umh --subme ◯ --merange ◯ --weightb --ref 3 --max-merge ◯ --early-skip --no-open-gop --min-keyint 5 --fades --bframes 8 --b-adapt 2 --pbratio 1.2 --fast-intra --b-intra --crf ◯ --crqpoffs -3 --cbqpoffs -1 --rdoq-level ◯ --aq-mode 4 --aq-strength ◯ --rd 3 --limit-modes --limit-refs 1 --rskip ◯ --rc-lookahead ◯ --rect --amp --psy-rd ◯ --splitrd-skip --limit-sao --sao-non-deblock --deblock 0:-1--y4m --input - --output ".\输出.hevc"

β——ffmpeg libx265 CLI，拷贝音频并封装为 mp4

- ffmpeg.exe -y -i ".\导入.mp4" -c:v libx265 -profile:v ◯ -x265-params "high-tier=1:ctu=◯:min-cu-size=16:tu-intra-depth=3:tu-inter-depth=3:limit-tu=1:rdpenalty=1:me=umh:subme=◯:merange=◯:weightb=1:ref=3:max-merge=◯:early-skip=1:open-gop=0:min-keyint=5:fades=1:bframes=8:b-adapt=2:pbratio=1.2:fast-intra=1:b-intra=1:crf=◯:crqpoffs=-3:cbqpoffs=-1:rdoq-aq-mode=4:aq-strength=◯:rd=3:limit-modes=1:limit-refs=1:rskip=◯:rc-lookahead=◯:rect=1:amp=1:psy-rd=◯:splitrd-skip=1:limit-sao=1:sao-non-deblock=1:deblock=0,-1" -fps_mode passthrough -c:a copy ".\输出.mp4"

高压·录像/3D 动画 建议高清源，否则画质不如通用-简单，更慢，但一般压缩率更高

兼容性 --profile<8/10/12bit: main/main10/main12, YUV4:2:2: main422-10/main422-12, YUV4:4:4: main444-8/main444-10/main444-12> --high-tier

分块-变换 --ctu 64 --tu-intra-depth 4 --tu-inter-depth 4 --limit-tu 1 --rect --tskip --tskip-fast

动搜-补偿 --me star --subme <24fps: 3, 48fps: 4, 60fps: 5, 100fps: 6> --merange <1920:1080: 48, 2560:1440: 52, 3840:2160: 56> --weightb

溯块-帧控 --ref 4 --max-merge 5 --no-open-gop --min-keyint 3 --keyint <9×帧率> --fades --bframes 8 --b-adapt 2 --analyze-src-pics

帧内编码 --b-intra <极慢且可能会造成块失真，增加压缩率: --constrained-intra>

量化 --crf 21.8 --crqpoffs -3 --ipratio 1.2 --pbratio 1.5

率失优量化 --rdoq-level 2

自适应量化 --aq-mode 4 --aq-strength <多面: 1~多线: 1.3> --qg-size 8

模式决策 --rd 5 --limit-refs 0 --rskip 0 --rc-lookahead <1.8×帧率，大于 bframes>

率失真优化 --psy-rd <录像: 1.6，动画: 0.6, ctu64: +0.6, ctu16: -0.6>

去块 --deblock 0:-1

取样迁就偏移 --limit-sao --sao-non-deblock --selective-sao 3

多处理器分配 --pools ,,,, (举例-,+表示该电脑有两个 CPU 节点，用第二个. 同时占用多个会造成严重的内存延迟)

其它 去黑边加速: --display-window <整数"←, ↑ ,→, ↓ "像素>, ≥22 核 cpu 优化: --pme, 分场视频: --field, 抖动高质量降色深: --dither, 开始; 结束帧: --seek; --frames, crf/abr 缓解噪点影响: --rc-grain

α——(ffmpeg pipe) x265 CLI 命令

- ffmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -strict -1 - | x265.exe --profile ◯ --high-tier --ctu 64 --tu-intra-depth 4 --tu-inter-depth 4 --limit-tu 1 --rect --tskip --tskip-fast --me star --subme ◯ --merange ◯ --weightb --ref 4 --max-merge 5 --no-open-gop --min-keyint 3 --keyint ◯ --fades --bframes 8 --b-adapt 2 --analyze-src-pics --b-intra --crf 21.8 --crqpoffs -3 --ipratio 1.2 --pbratio 1.5 --rdoq-level 2 --aq-mode 4 --aq-strength ◯ --qg-size 8 --rd 5 --limit-refs 0 --rskip 0 --rc-lookahead ◯--psy-rd ◯ --deblock 0:-1 --limit-sao --sao-non-deblock --selective-sao 3--y4m --input - --output ".\输出.hevc"

β——ffmpeg libx265 CLI，拷贝音频并封装为 mp4

- ffmpeg.exe -y -i ".\导入.mp4" -c:v libx265 -profile:v ◯ -x265-params "high-tier=1:ctu=64:tu-intra-depth=4:tu-inter-depth=4:limit-tu=1:rect=1:tskip=1:tskip-fast=1:me=star:subme= ◯ :merange= ◯ :weightb=1:ref=4:max-merge=5:open-gop=0:min-keyint=3:keyint= ◯ :fades=1:bframes=8:b-adapt=2:analyze-src-pics=1:b-intra=1:crf=21.8:crqpoffs=-3:ipratio=1.2:pbratio=1.5:rdoq-level=2:aq-mode=4:aq-strength= ◯ :qg-size=8:rd=5:limit-refs=0:rskip=0:rc-lookahead= ◯ :psy-rd= ◯ :deblock=0,-1:limit-sao=1:sao-non-deblock=1:selective-sao=3" -fps_mode passthrough -c:a copy ".\输出.mp4"

剪辑素材存档 通过减少 P 帧，B 帧数量来降低解码压力，从而降低剪辑软件负载；兼容≥画质+压缩

兼容性 --profile<8/10/12bit: main/main10/main12, YUV4:2:2: main422-10/main422-12, YUV4:4:4: main444-8/main444-10/main444-12> --high-tier

分块-变换 --ctu 32 --tskip

动态搜索 --me star --subme <24fps: 3, 48fps: 4, 60fps: 5, 100fps: 6> --merange <1920:1080: 48, 2560:1440: 52, 3840:2160: 56> --analyze-src-pics

帧内搜索 --max-merge 5 --early-skip --b-intra

帧控制 --no-open-gop --min-keyint 1 --keyint <5×帧率>--ref 3 --fades --bframes 4 --b-adapt 2

量化 --crf 17 --crqpoffs -3 --cbqpoffs -2

模式决策 --rd 3 --limit-modes --limit-refs 1 --rskip 1 --rc-lookahead <4×帧率, 大于 bframes>

率失真优化 --splitrd-skip

环路滤波去块 --deblock -1:-1

主控 --tune grain

其它 去黑边加速: --display-window <整数"←, ↑ ,→, ↓ "像素>, ≥22 核 cpu 优化: --pme, 分场视频: --field,

α——(ffmpeg pipe) x265 CLI 命令

- ffmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -strict -1 - | x265.exe --profile ◯ --high-tier --ctu 32 --tskip --me star --subme ◯ --merange ◯ --analyze-src-pics --max-merge 5 --early-skip --b-intra --no-open-gop --min-keyint 1 --keyint ◯ --ref 3 --fades --bframes 7 --b-adapt 2 --crf 17 --crqpoffs -3 --cbqpoffs -2 --rd 3 --limit-modes --limit-refs 1 --rskip 1 --rc-lookahead ◯ --splitrd-skip --deblock -1:-1--tune grain --y4m --input - --output ".\输出.hevc"

β——ffmpeg libx265 CLI，拷贝音频并封装为 mp4

- ffmpeg.exe -y -i ".\导入.mp4" -c:v libx265 -profile:v ◯ -x265-params "high-tier=1:ctu=32:tskip=1:me=star:subme=◯:merange=◯:analyze-src-pics=1:max-merge=5:early-skip=1:open-gop=0:min-keyint=1:keyint=◯:ref=3:fades=1:bframes=7:b-adapt=2:b-intra=1:crf=17:crqpoffs =-3:cbqpoffs=-2:rd=3:limit-modes=1:limit-refs=1:rskip=1:rc-lookahead=◯:splitrd-skip=1:deblock=-1,-1:tune=grain" -fps_mode passthrough -c:a copy ".\输出.mp4"

高压·动漫·字幕组

建议 YUV4:2:0; 8~10bit

兼容性 --profile<8/10/12bit: main/main10/main12, YUV4:2:2: main422-10/main422-12, YUV4:4:4: main444-8/main444-10/main444-12> --high-tier

分块-变换 --tu-intra-depth 4 --tu-inter-depth 4 --max-tu-size 16 --tskip --tskip-fast

动搜-补偿 --me umh --subme <24fps: 3, 48fps: 4, 60fps: 5, 100fps: 6> --merange <1920:1080: 48, 2560:1440: 52, 3840:2160: 56> --weightb --max-merge 5 --early-skip

溯块-帧控 --ref 3 --no-open-gop --min-keyint 5 --keyint <12×帧率> --fades --bframes 16 --b-adapt 2 --bframe-bias 20

帧内编码 --b-intra <极慢且可能会造成画面问题: + --constrained-intra>

量化 --crf 22 --crqpoffs -4 --cbqpoffs -2 --ipratio 1.6 --pbratio 1.3 --cu-lossless --psy-rdoq 2.3 --rdoq-level 2

率失优量化 --hevc-aq --aq-strength 0.9 --qg-size 8

自适应量化 --rd 3 --limit-modes --limit-refs 1 --rskip 1 --rc-lookahead <2.5×帧率, 大于 bframes> --

模式决策 rect --amp

率失真优化 --psy-rd 1.5 --splitrd-skip --rdpenalty 2

去块 --deblock 0:-1

取样迁就偏移 --limit-sao --sao-non-deblock

其它 去黑边加速: --display-window <整数"←, ↑, →, ↓ "像素>, ≥22 核 cpu 优化: --pme, 分场视频: --field, 抖动高质量降色深: --dither, 开始; 结束帧: --seek; --frames, crf/abr 缓解噪点影响: --rc-grain, 外/内网 NAS 串流: --single-sei --idr-recovery-sei

α——(ffmpeg pipe) x265 CLI 命令

- ffmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -strict -1 - | x265.exe --profile ◯ --high-tier --tu-intra-depth 4 --tu-inter-depth 4 --max-tu-size 16 --tskip --tskip-fast --me umh --subme ◯ --merange ◯ --weightb --max-merge 5 --early-skip --ref 3 --no-open-gop --min-keyint 5 --keyint ◯ --fades --bframes 16 --b-adapt 2 --bframe-bias 20 --constrained-intra --b-intra --crf 22 --crqpoffs -4 --cbqpoffs -2 --ipratio 1.6 --pbratio 1.3 --cu-lossless --psy-rdoq 2.3 --rdoq-level 2 --hevc-aq --aq-strength 0.9 --qg-size 8 --rd 3 --limit-modes --limit-refs 1 --rskip 1 --rc-lookahead ◯ --rect --amp --psy-rd 1.5 --splitrd-skip --rdpenalty 2 --deblock -1:0 --limit-sao --sao-non-deblock --y4m --input - --output ".\输出.hevc"

β——ffmpeg libx265 CLI，拷贝音频并封装为 mp4

- ffmpeg.exe -y -i ".\导入.mp4" -c:v libx265 -profile:v ◯ -x265-params "high-tier=1:tu-intra-depth=4:tu-inter-depth=4:max-tu-size=16:tskip=1:tskip-fast=1:me=umh:subme=◯:merange=◯:weightb=1:max-merge=5:early-skip=1:ref=3:open-gop=0:min-keyint=5:keyint=◯:fades=1:bframes=16:b-adapt=2:bframe-bias=20:b-intra=1:crf=22:crqpoffs=-4:cbqpoffs=-2:ipratio=1.6:pbratio=1.3:cu-lossless=1:psy-rdoq=2.3:rdoq-level=2:hevc-aq=1:aq-strength=0.9:qg-size=8:rd=3:limit-modes=1:limit-refs=1:rskip=1:rc-lookahead=◯:rect=1:amp=1:psy-rd=1.5:splitrd-skip=1:rdpenalty=2:deblock=-1,0:limit-sao=1:sao-non-deblock=1" -fps_mode passthrough -c:a copy ".\输出.mp4"

动漫/原画·高算力 HEDT 工作站压力高，画质高，压缩率不高，不适合大部分情况

兼容性 --profile<8/10/12bit: **main**/**main10**/**main12**, YUV4:2:2: **main422-10**/**main422-12**, YUV4:4:4: **main444-8**/**main444-10**/**main444-12**> --high-tier

分块-变换 --tu-intra-depth 4 --tu-inter-depth 4 --max-tu-size 4 --limit-tu 1 --rect --amp --tskip

动搜-补偿 --me star --subme <24fps: **3**, 48fps: **4**, 60fps: **5**, 100fps: **6**> --merange <1920:1080: **52**, 2560:1440: **56**, 3840:2160: **64**> --analyze-src-pics --weightb --max-merge 5

溯块-帧控 --ref 3 --no-open-gop --min-keyint 1 --keyint <**12×帧率**> --fades --bframes 16 --b-adapt 2

帧内编码 --b-intra

量化 --crf 18.1 --crqpoffs -5 --cbqpoffs -2 --ipratio 1.67 --pbratio 1.33 --cu-lossless

率失优量化 --psy-rdoq 2.5 --rdoq-level 2

自适应量化 <普通: **--hevc-aq --aq-strength 1.4**; Jpsdr Mod: **--aq-auto 10 --aq-bias-strength 1.3 --aq-strength-edge 1.4 --aq-bias-strength 1.1**> --qg-size 8

模式决策 --rd 5 --limit-refs 0 --rskip 2 --rskip-edge-threshold 3 --rc-lookahead <**2.5×帧率**, 大于bframes> --no-cutree

率失真优化 --psy-rd 1.5 --rdpenalty 2 <实验性: --qp-adaptation-range 5>

去块 --deblock -2:-2

取样迁就偏移 --limit-sao --sao-non-deblock --selective-sao 1

α——(ffmpeg pipe) 普通 x265 CLI 命令

- ffmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -strict -1 - | x265.exe --profile ◯ --high-tier --tu-intra-depth 4 --tu-inter-depth 4 --max-tu-size 4 --limit-tu 1 --rect --amp --tskip --me star --subme ◯ --merange ◯ --analyze-src-pics --weightb --max-merge 5 --ref 3 --no-open-gop --min-keyint 1 --keyint ◯ --fades --bframes 16 --b-adapt 2 --b-intra --crf 18.1 --crqpoffs -5 --cbqpoffs -2 --ipratio 1.67 --pbratio 1.33 --cu-lossless --psy-rdoq 2.5 --rdoq-level 2 --hevc-aq --aq-strength 1.4 --qg-size 8 --rd 5 --limit-refs 0 --rskip 2 --rskip-edge-threshold 3 --rc-lookahead ◯--no-cutree --psy-rd 1.5 --rdpenalty 2 --deblock -2:-2 --limit-sao --sao-non-deblock --selective-sao 1--y4m --input - --output ".\输出.hevc"

β——(ffmpeg pipe) x265 jpsdr-Mod CLI 命令

- ffmpeg.exe -y -i ".\导入.mp4" -an -f yuv4mpegpipe -strict -1 - | x265.exe --profile ◯ --high-tier --tu-intra-depth 4 --tu-inter-depth 4 --max-tu-size 4 --limit-tu 1 --rect --amp --tskip --me star --subme ◯ --merange ◯ --analyze-src-pics --weightb --max-merge 5 --ref 5 --no-open-gop --min-keyint 1 --keyint ◯ --fades --bframes 16 --b-adapt 2 --b-intra --crf 18.1 --crqpoffs -5 --cbqpoffs -2 --ipratio 1.67 --pbratio 1.33 --cu-lossless --psy-rdoq 2.5 --rdoq-level 2 --aq-auto 10 --aq-bias-strength 1.3 --aq-strength-edge 1.4 --aq-bias-strength 1.1 --qg-size 8 --rd 3 --limit-refs 0 --rskip 2 --rskip-edge-threshold 3 --rc-lookahead ◯--no-cutree --psy-rd 1.5 --rdpenalty 2 --deblock -2:-2 --limit-sao --sao-non-deblock --selective-sao 1--y4m --input - --output ".\输出.hevc"

γ——普通 ffmpeg libx265 CLI，拷贝音频并封装为 mp4

- ffmpeg.exe -y -i ".\导入.mp4" -c:v libx265 -profile:v ◯ -x265-params "high-tier=1:tu-intra-depth=4:tu-inter-depth=4:max-tu-size=4:limit-tu=1:rect=1:amp=1:tskip=1:me=star:subme=◯:merange=◯:analyze-src-pics=1:weightb=1:max-merge=5:ref=3:open-gop=0:min-keyint=1:keyint=

◯:fades=1:bframes=16:b-adapt=2:b-intra=1:crf=18.1:crqpoffs=-5:cbqpoffs=-2:ipratio=1.6:pbratio=1.33:cu-lossless=1:psy-rdoq=2.5:rdoq-level=2:hevc-aq=1:aq-strength=1.4:qg-size=8:rd=5:limit-refs=0:rskip=2:rskip-edge-threshold=3:rc-lookahead=◯:cutree=0:psy-rd=1.5:rdpenalty=2:deblock=-2:-2:limit-sao=1:sao-non-deblock=1:selective-sao=1" -fps_mode passthrough -c:a copy ".\输出.mp4"