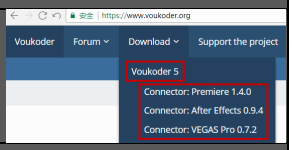
观看本教程意味着你已经阅读了 x264 教程完整版，对视频压缩相关的软硬件和技术有了初步了解。本教程的目的是把参数直接贴到软件里用，但不一定符合实际要求，因此建议搭配 x264 教程完整版做适当调整。

教程地图，软件下载

地图见 iavoe.github.io。点击下方表格中的超链接以下载软件。

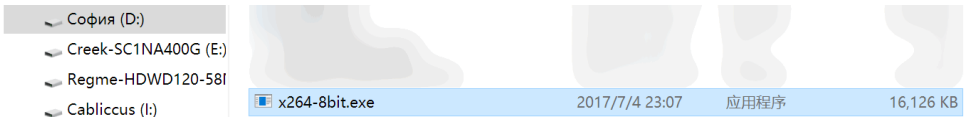
ffmpeg~ffprobe 顶级开源多系统多媒体 CLI 处理~检测工具
mpv 支持便携的开源多系统现代视频播放器。见安装与配置教程
Voukoder 开源 Premiere、Vegas、Aftereffects 压制导出插件，分为 Voukoder 和 V-Connector 两部分 
MediaInfo 开源 GUI 媒体元数据/视音频格式读取器，用于快速查看完整元数据
x264 by Patman, **LigH**√lavf 编解码
x264 tMod by jspdr√lavf 编解码，支持 MCF 线程管理库(比 posix 和 win32 性能更好)

程序下载与命令行用法

1. 于上表下载 ffmpeg，ffprobe/MediaInfo，x264 并记住路径

选择编码器位深?

有同时含 8-10-12bit 的 x264.exe，以及区分为 x264-8bit.exe，x264-10bit.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 管道传递位深信息到编码器，使 x264.exe 自动设置位深。

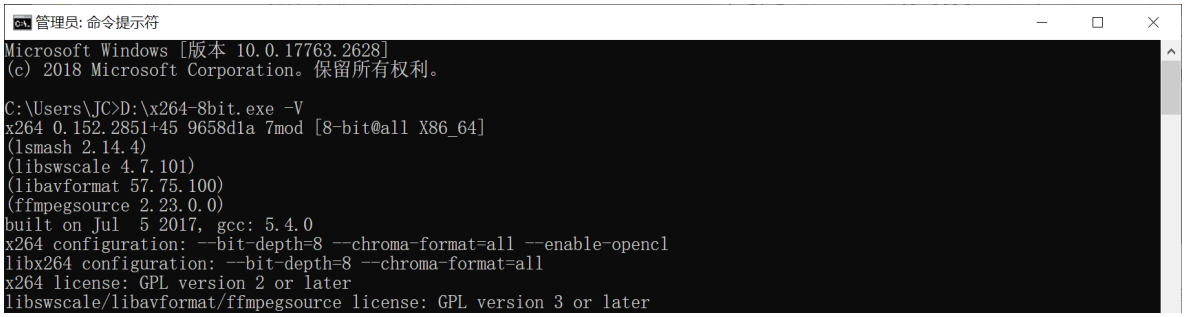


此处置于 Windows 系统 D 盘根目录下，因此路径为 `D:\`

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

路径拼写：直接选择并复制以配置命令行

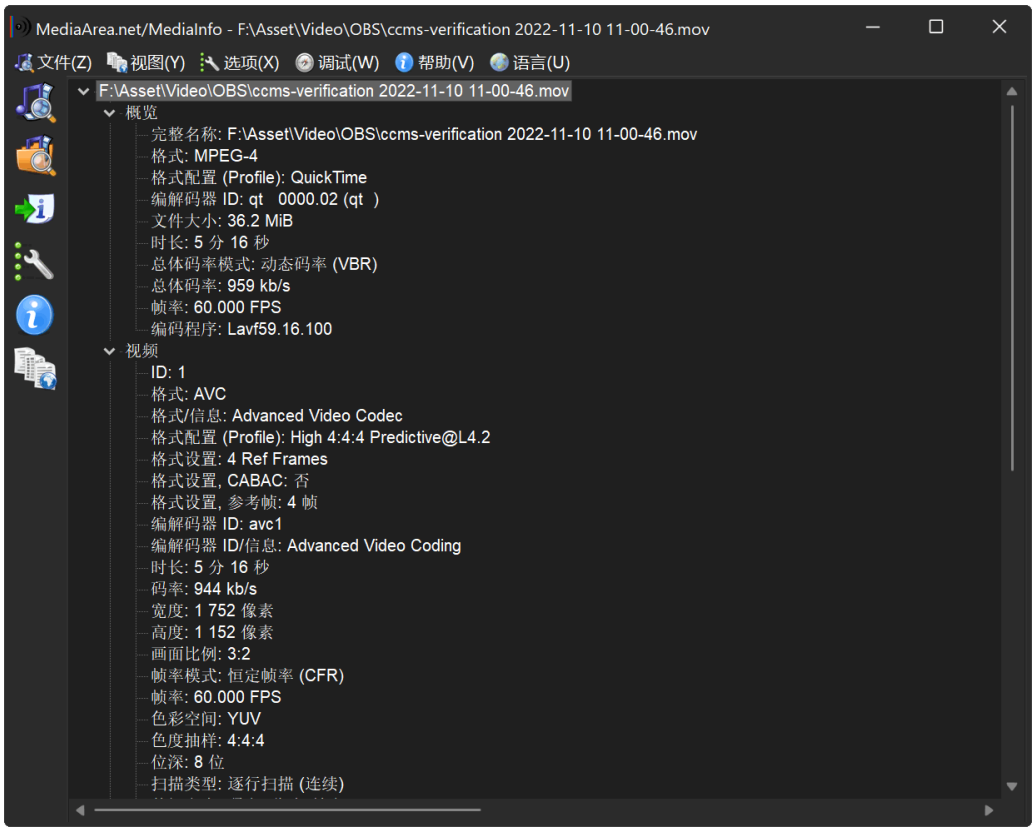
程序版本：越新越好，编码器的大版本更新有 Bug 修复与性能提升，其它软件的更新有新格式兼容，翻译改进等体验提升



图中检查 D:\x265-4bit.exe -V 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

配置最终参数

在确保了兼容和可行性后，即可使用命令行进行视频压制。在 CMD/PowerShell/Bash/Terminal 中，输入与上述 ffprobe 同类的命令即可。使用 x264 时，导入命令的部分被放在了命令行末尾，与 ffmpeg/ffprobe 不同，x264 的程序要求不使用专门的“-i”命令指定导入文件和路径，而是放在命令行末尾的一段命令

[参数格式] `x264.exe --me esa --merange 24 [...] --output "导出.mp4" "导入.mp4"`

[参数用例] `D:\x264-8bit.exe --me umh --subme 11 --merange 32 -I 270 -i 1 -b 11 --b-adapt 2 -r 3 --direct auto --crf 19 --qpmin 13 --rc-lookahead 90 --aq-mode 3 --aq-strength 1 --trellis 2 --deblock 0:0 --psy-rd 0.7:0.2 --fullrange --vf hqdn3d:1.1,1.1,1.1,1.1 --output "F:\导出.mp4" "D:\导入.mp4"`

空格，标点与拼写错误

在命令行程序中，参数和参数值、命令和命令之间通常使用空格作为分隔符，因此路径（参数值）中如果有空格，则应该用英文直引号 "" 括起来。同样的，程序往往使用英文逗号 , 或英文冒号 : 来分隔多值。如果命令拼写错误，则命令行程序找不到命令，也会报错。空格与引起的报错往往以空格之后的“命令”不明的形式呈现、分隔符引起的报错往往以不明参数值的形式呈现、拼写引起的报错往往以命令不明形式呈

现

在 API 命令行程序中的符号不同。拼写起来会麻烦一些，但规则与报错规律同样适用。

一般命令行参数格式，ffmpeg 常用操作，命令行操作技巧

虽然是 x265 编码器的教程，但格式一致，ffmpeg 部分也通用。因此见 x265 教程完整版

ffmpeg 有兼容性问题的参数

`-hwaccel auto`：自动选择硬件解码，由于部分硬件厂商的解码画质差，或忽略关键信息，所以可能会花屏

ffmpeg 非必要参数

`-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

x264 HDR 设置参数:

x264

--master-display

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

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

HDR 标识

bt2020: G(8500,39850)B(6550,2300)R(35400,14600)WP(15635,16450)L(?,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)>

-- cll <和 master-display 的 L 最大值一样>

--colormatrix <照源，例: gbr bt709 fcc bt470bg smpte170m YCgCo bt2020nc bt2020c smpte2085 ictcp>

--transfer <照源，例: gbr bt709 fcc bt470bg smpte170m YCgCo bt2020nc bt2020c smpte2085 ictcp>

通用·简单

去掉全部自定义项目，方便急用但降低了特定画面的压缩率

前瞻进程 --rc-lookahead 90 --bframes 12 --b-adapt 2

动态-帧内搜索 --me umh --subme 9 --merange 48 --no-fast-pskip --direct auto --weightb

帧控-参考 --keyint 360 --min-keyint 5 --ref 3

量化 --crf 20 --qpmin 9 --chroma-qp-offset -2

自适量 --aq-mode 3 --aq-strength 0.7 --trellis 2

环滤/RDO --deblock 0:0 --psy-rd 0.77:0.22 --fgo 10

降噪 --nr 8

色彩范围 --fullrange<非 7mod x264 用，检查源视频是否使用完整色彩范围>

动-帧快速搜索 --me hex --subme 8 --merange 32 --direct auto --weightb

参考冗余优先 --sliced-threads <降低 CPU 占用，减速但时域复杂画面的压缩率可能提高，参考错误降低>

放/裁/边/降噪 --vf crop:左,上,右,下/resize:缩放后宽,缩放后高,,,,bicubic/pad:左,上,右,下,直接宽,直接高

滤镜 /hqdn3d:1,1,1,1.5

划区压制 --zones 0,<片头 OP 结束帧>,crf=30 --zones<片尾 ED 开始帧>,<片尾 ED 结束帧>,crf=30

压制范围 --seek 从第<>帧开始压 --frame 压制<>帧后停止 --fps 元数据没写多少时手动指定帧数

α——x264 CLI 命令

x264.exe --rc-lookahead 90 --bframes 12 --b-adapt 2 --me umh --subme 9 --merange 48 --no-fast-pskip --direct auto --weightb --keyint 360 --min-keyint 5 --ref 3 --crf 20 --qpmin 9 --chroma-qp-

offset -2 --aq-mode 3 --aq-strength 0.7 --trellis 2 --deblock 0:0 --psy-rd 0.77:0.22 --fgo 10 --nr 4 --output ".\输出.mp4" ".\导入.mp4"

β——libx264 私有 CLI，兼容 libav，不支持 fgo

ffmpeg.exe-y -i ".\导入.mp4" -c:v libx264 -x264-params "rc-lookahead=90:bframes=12:b-adapt=2:me=umh:subme=9:merange=48:fast-pskip=0:direct=auto:weightb=1:keyint=360:min-keyint=5:ref=3:crf=20:qpmin=9:chroma-qp-offset=-2:aq-mode=3:aq-strength=0.7:trellis=2:deblock=0,0:psy-rd=0.77,0.22:nr=4" -fps_mode passthrough -c:a copy ".\输出.mp4"

γ——libx264 私有 CLI，不支持 fgo

上面的 -x264-params 改成 -x264opts，但功能完全相同

δ——libx264 ffmpeg CLI 命令

ffmpeg.exe-y -i ".\导入.mp4" -c:v libx264 -bf 12 -b_strategy 2 -me_method umh -subq 9 -me_range 48 -flags2 -fastpskip -directpred 3 -flags2 +wpred -g 360 -keyint_min 5 -refs 3 -crf 20 -qmin 9 -chromaoffset -2 -aq-mode 3 -aq-strength 0.7 -trellis 2 -deblockalpha 0 -deblockbeta 0 -psy-rd 0.77:0.22 -nr 4 -flags2 +bpyramid -fps_mode passthrough -c:a copy ".\输出.mp4"

通用·标准

配置起来慢些但自定义范围广。由于 x264 参数不多，所以这套参数足以涵盖高画质高压-录像-动画情形

分块-前瞻 --partitions all --rc-lookahead <3×帧率> --bframes 12 --b-adapt 2

动态-帧内搜索 --me umh --subme <电影 10~11，动漫 9 (11 仅 7mod)> --merange <快 20，高压 48，4 的倍数> --no-fast-pskip --direct auto --weightb

帧控-参考 --keyint <8~10×帧率> --min-keyint <1 增加 IDR，5 常用> --ref 3

量化 --crf 19 --chroma-qp-offset <常用-1~-2，动漫-3~-5>

自适应量化 --aq-mode 3 --aq-strength <一般 0.7，原画 1.1> --trellis 2

环滤/RDO --deblock <一般 0:0，原画-1:-1> --psy-rd <动漫 0.4~.6:0.1~.15，录像 0.7~1.3:0.12~.2> --fgo 12

CRF-VBR 压缩 --nal-hrd --vbv-bufsize <最大 kbps 每秒> --vbv-maxrate <bufsize 倍数的 kbps>

色彩范围 --fullrange<非 7mod x264 用，检查源视频是否使用完整色彩范围>

参考冗余优先 --sliced-threads <降 CPU 占用，减速但时域复杂画面的压缩率可能提高，参考错误降低>

放/裁/边/降噪 --vf crop:左,上,右,下/resize:缩放后宽,缩放后高,,,,bicubic/pad:左,上,右,下,直接宽,直接高

参数划区压制 /hqdn3d:1.1,1.1,1.1,1.1

压制范围 --zones 0,<片头 OP 结束帧>,crf=30 --zones<片尾 ED 开始帧>,<片尾 ED 结束帧>,crf=30

α——x264 CLI 命令

x264.exe --partitions all --rc-lookahead ◯ --me umh --bframes 12 --b-adapt 2 --subme ◯ --merange ◯ --no-fast-pskip --direct auto --weightb --keyint ◯ --min-keyint ◯ --ref 3 --crf 18 --chroma-qp-offset ◯ --aq-mode 3 --aq-strength ◯ --trellis 2 --deblock ◯ --psy-rd ◯ --fgo 12 --output ".\输出.mp4" ".\导入.mp4"

β——libx264 私有 CLI，兼容 libav，不支持 fgo

ffmpeg.exe-y -i ".\导入.mp4" -c:v libx264 -x264-params "partitions=all:rc-lookahead=◯:me=umh:bframes=12:b-adapt=2:subme=◯:merange=◯:fast-pskip=0:direct=auto:weightb=1:keyint=◯:min-keyint=◯:ref=3: crf=18:chroma-qp-offset=◯:aq-mode=3:aq-strength=◯:trellis=2:deblock=0,-1:psy-rd=◯,◯:nr=4" -fps_mode passthrough -c:a copy ".\输出.mp4"

γ——libx264 私有 CLI，不支持 fgo

上面的 `-x264-params` 改成 `-x264opts`，功能大致相同

δ——libx264 ffmpeg CLI 命令

注：ffmpeg 的 x264 受版权限制而重写了部分参数名；有的 ffmpeg 版本可能又没有重写，较为不便

ffmpeg.exe-y -i ".\导入.mp4" -c:v libx264 -me_method umh -subq ◯ -me_range ◯ -flags2 -fastpskip -directpred 3 -flags2 +wpred -g ◯ -keyint_min ◯ -bf 12 -b_strategy 2 -refs 3 -crf 19 -qmin 9 -chromaoffset ◯ -aq-mode 3 -aq-strength ◯ -trellis 2 -deblockalpha 0 -deblockbeta -1 -psy-rd ◯:◯ -nr 4 -flags2 +bpyramid -fps_mode passthrough -c:a copy ".\输出.mp4"

剪辑素材存档

加强无损压缩，降低有损压缩，增加 IDR 帧数量. 建议 YUV4:2:2 或 4:4:4 8~10bit

分块-前瞻 --partitions all --bframes 12 --b-adapt 2

动态-帧内搜索 --me umh --subme <电影 10~11，动漫 9 (11 仅 7mod)> --merange <快速 40，高压 48> --no-fast-pskip --direct auto --weightb

帧控-参考 --keyint <5~8×帧率> --min-keyint 1 --ref 3 --sliced-threads

量化-主控 --crf 16 --tune grain

自适-RDO --trellis 2 --fgo 15

划区压制 --zones 0,<片头/OP 结束帧>,crf=30 --zones<片尾/ED 开始帧>,<片尾/ED 结束帧>,crf=30

压制范围 --seek 从第<>帧开始压 --frame 压制<>帧后停止 --fps 元数据没写多少时手动指定帧数

α——x264 CLI 命令

x264.exe --partitions all --bframes 12 --b-adapt 2 --me esa --subme ◯ --merange ◯ --no-fast-pskip --direct auto --weightb --keyint ◯ --min-keyint 1 --ref 3 --crf 16 --tune grain --trellis 2 --fgo 15 --output ".\输出.mp4" ".\导入.mp4"

β——libx264 私有 CLI，兼容 libav，不支持 fgo

ffmpeg.exe-y -i ".\导入.mp4" -c:v libx264 -x264-params "partitions=all:me=umh:subme=◯:merange=◯:fast-pskip=0:direct=auto:weightb=1:keyint=◯:min-keyint=1:bframes=12:b-adapt=2:ref=3:crf=16:rellis=2" -

fps_mode passthrough −c:a copy ".\输出.mp4"

γ——libx264 私有 CLI，不支持 fgo

上面的 `-x264-params` 改成 `-x264opts`，功能大致相同

δ——libx264 ffmpeg CLI 命令

ffmpeg.exe−y −i ".\导入.mp4" −c:v libx264 −bf 12 −b_strategy 2 −me_method esa −subq ◯ −me_range ◯ −flags2 −fastpskip −directpred 3 −flags2 +wpred −g ◯ −keyint_min 1 −refs 3 −crf 17 −tune grain −trellis 2 −flags2 +bpyramid −fps_mode passthrough −c:a copy ".\输出.mp4"

OBS 录屏

由于消费端算力增加，因此除少数吃满 CPU 的场景外，使用 CPU 录屏的性价比，无论画质还是成本都已优于要额外购置的硬件编码方案（采集卡、最新代际显卡）。OBS 是一款开源免费的专业直播软件，附带有高度自定义的录屏功能。以下是 OBS 的配置与参数。

高级

一般除文件名外和图中保持一致即可。其它选项建议在正式录制前测试兼容性。

OBS 使用 FOURCC 代码表示色彩空间与位深。兼容性最高的是 4:2:0 8bit 格式，如果需要保存精细的色度信息，且已验证兼容，则可以尝试 10bit、4:4:4 等高位深/高精度格式。

ffmpeg 空间	OBS-FOURCC	位深	平面布置（数量）
yuv420p	I420 / YV12	8	planar（3）
nv12	NV12	8	semi-planar（2）

yuv420p10le	I010	10	planar（3）
p010le	P010	10	semi-planar（2）
yuv422p	Y42B	8	planar（3）
yuv444p	YV24	8	planar（3）

热键

开始/停止录制可以使用快捷键实现，看个人习惯，但应注意测试其它软件、系统、输入法中可能存在的冲突。例如图中的 Ctrl+Shift+X 尽管非常少见，但在浏览器的网址框中其实是切换左~右对齐的按键。

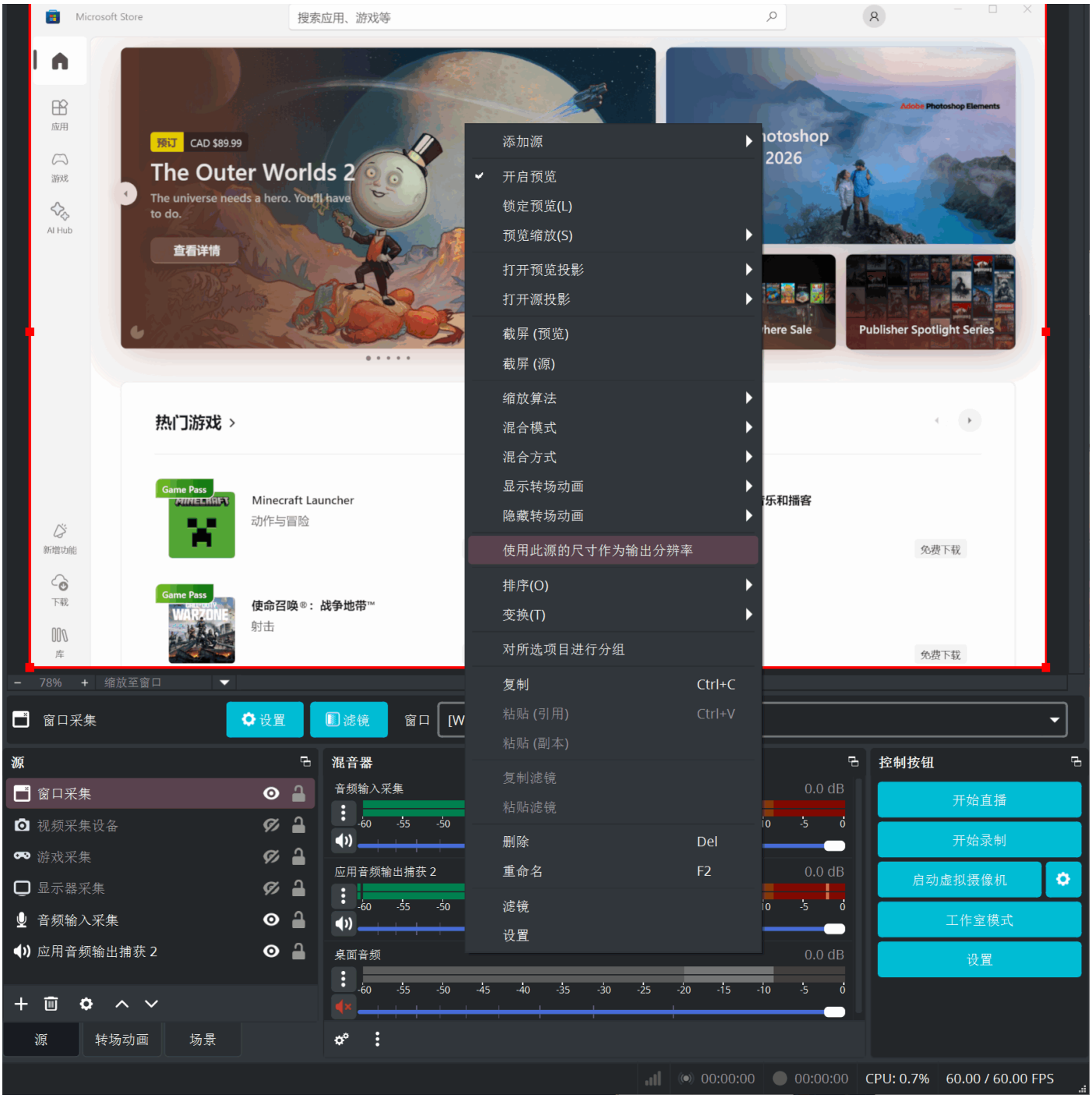
视频

帧率

设为 60；背景高速移动则建议 90 或 120，更低帧率需在正式录制前测试和规定所有元素的移动速度，确保视频流畅度后可行。尽管高帧率会增加文件体积，但录制步骤在媒体处理链路之先，因此必须预留信息余量给后续步骤（包括平台转码），以避免画质与流畅度受到二次损失。

分辨率

窗口录制：主页中新建“窗口采集”并选中需录制的窗口，画面中右键 → 使用此源的输出作为分辨率。

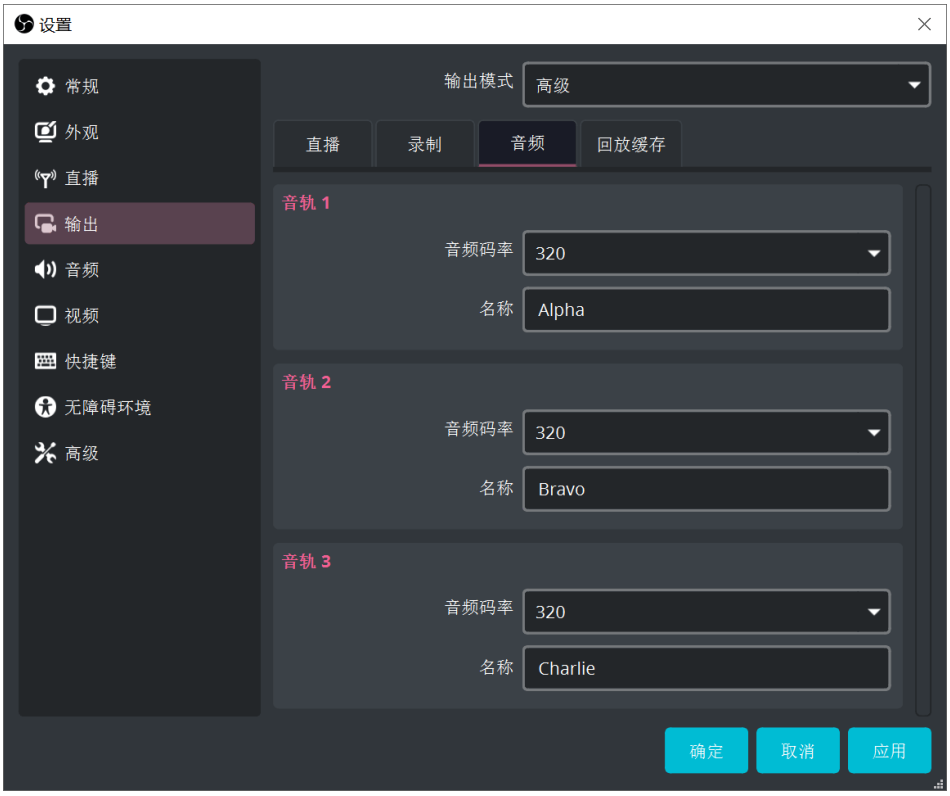


全屏录制：根据是否要录制桌面底部的任务栏来决定视频高，如分辨率为 1920x1080 时，避开任务栏使用 1920x1040；3840x2160 则使用 3840x2080。

音频

输出→音频→码率

建议选择 320kbps（不要低于 256kbps）



主页→源

建议安装 win-capture-audio 插件录制应用程序的音轨（降低延迟），原理是录制的对象从 Windows 音频输出改为应用程序进程，但可能会与部分程序不兼容。只要 win-capture-audio 能用，就不建议使用延迟更高的音频输出采集（添加源列表中）或桌面音频（默认开启）。开启多个音源时，注意这些音源采集的目标不同。

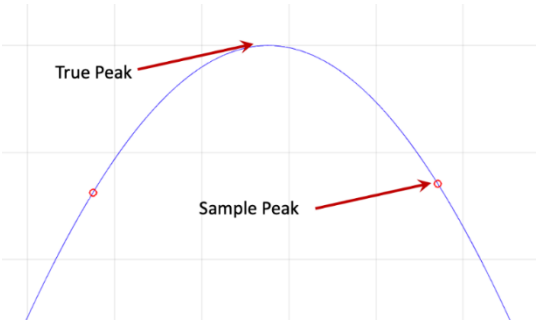
主页→混音器→音源 [⋮]→音频插件

不建议在录制时添加任何滤镜，而是之后使用编辑软件处理（如 Adobe Audition 的杂音降噪器滤镜、清理旁白电平效果组），从而避免录制结果作废，以及额外的 CPU 占用延迟。

音频→采样率

不高于声卡 DAC 默认设置（除非有问题，否则不要更改声卡默认），OBS 中最高 48kHz（48000），这个值代表了最高音高（采样率÷2，以避免混叠失真 aliasing），也就是齿音中最尖锐部分的信号、和“擦”这种乐器所能发出的音高上限。当然，44.1kHz（44100）也是完全足够的。

音频→电平表



涉及专业混音监听显示，但 OBS 只负责录制，分工上不应搀和混音操作，因此选择低占用（采样峰值）的电平表显示方法即可。真峰值 True Peak 的作用是通过超采样（二倍放大）得出每个采样点之间的真实音频响度，这在混音，尤其是多轨混音中起到避免响度超标触发平台/标准惩罚机制的作用。然而，OBS 具备多音轨导出能力，因此多轨混音仍是应该对症下药地交给混音软件去处理（如 Adobe Audition、FL Studio 的多轨混音）。图片来源：docs.dolby.io

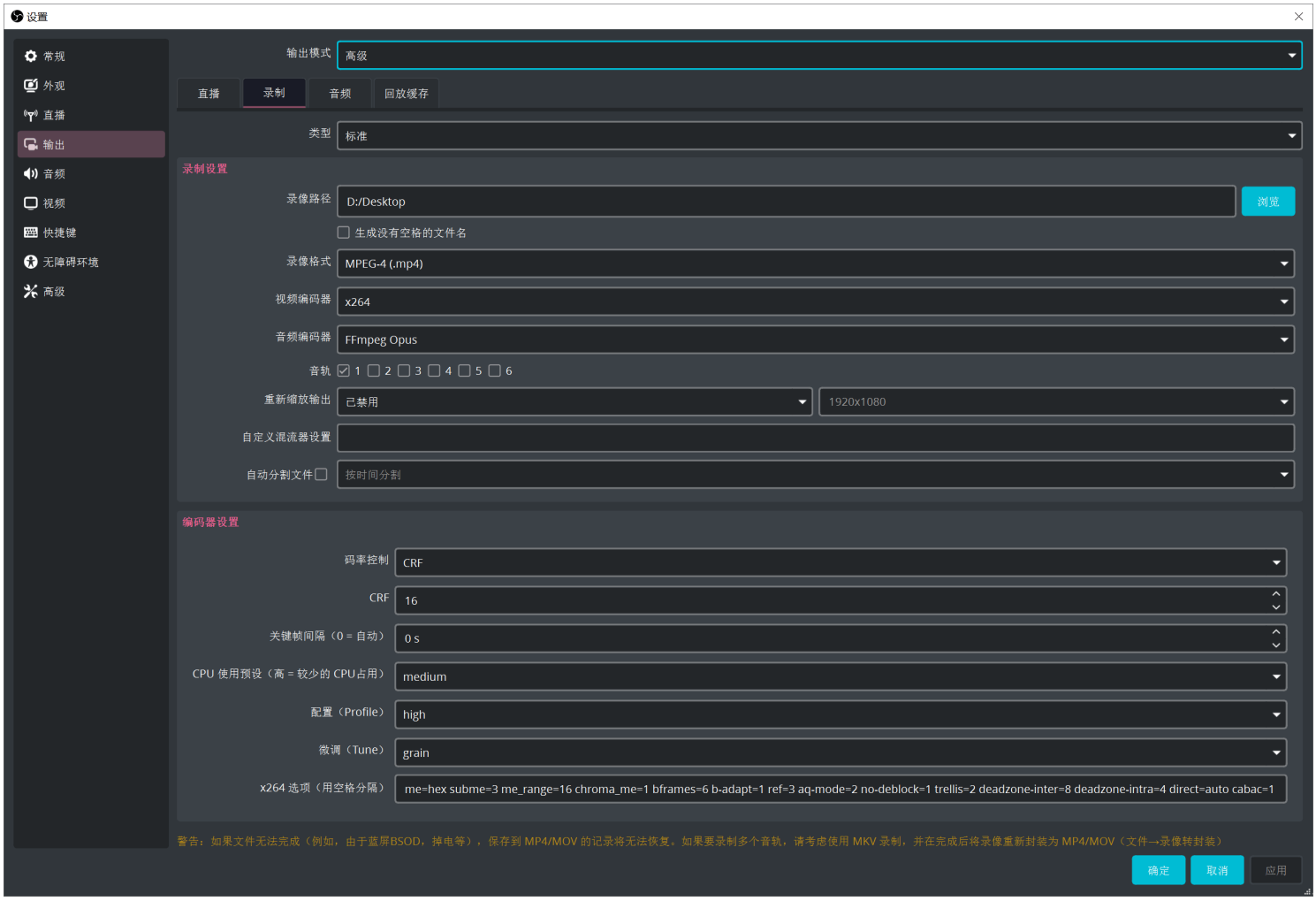
输出→录制→音频编码器

音频格式\封装格式	.flv	.mkv	.mp4	.mov	混合 MP4	分片 MP4	分片 MOV	.ts	.m3u8+.ts
FFmpeg AAC 有损，音质最差，高兼容	✔	✔	✔	✔	⚠	⚠	⚠	✔	✔
FFmpeg ALAC 无损，24bit	✘	✔	✔	✔	⚠	⚠	⚠	✘	✘
FFmpeg FLAC 无损，16bit	✘	✔	⚠	⚠	⚠	✘	✘	✘	✘
FFmpeg OPUS 有损，音质好，兼容一般	✘	✔	✔	⚠	⚠	⚠	✘	✘	✘
FFmpeg PCM 无损，16/24/32bit	✘	✔	⚠	✔	⚠	⚠	⚠	✔	✔

根据视频封装格式、剪辑软件及视频平台的限制选择。如果既要高压缩，又要兼容工作流，可以考虑录制为 OPUS（.ogg），之后用如 `ffmpeg -i "视频.mp4" -vn -c:a pcm_s16le "导出波形.wav"` 的命令无损转换。

编码

输出→录制→编码器设置



参数\预设 preset	ultrafast	superfast	veryfast	faster	fast	medium	slow	slower	veryslow	placebo
b-adapt	0	默认 1					2			
bframes	0	默认 3							8	16
direct	默认 spatio						auto			
me	dia		默认 hex				umh			tesa
merange	默认 16								24	

参数\预设 preset	ultrafast	superfast	veryfast	faster	fast	medium	slow	slower	veryslow	placebo
partitions	none	i8x8,i4x4	默认 p8x8,b8x8,i8x8,i4x4					all		
rc-lookahead	0		10	20	30	默认 40	50	60		
ref	1			2		默认 3	5	8	16	
subme	0	1	2	4	6	默认 7	8	9	10	11
trellis	0			默认 1			2			
weightp	0	1				默认 2				
no-weightb	1	默认 0								
no-8x8dct	1	默认 0								
no-cabac	1	默认 0								
no-deblock	1	默认 0								
no-mbtree	1		默认 0							
no-mixed-refs	1				默认 0					
scenecut	0	默认 40								
no-fast-pskip	默认 0									1
slow-first-pass	默认 0									1

其中的 `bframes` 建议在自定义参数中手动覆盖，设置在 7~10 的范围

输出→录制（x264 编码器）→x264 选项

本地录屏		内网推流	
码率控制：	CRF	**码率控制：**	ABR
CPU 使用预设：	medium	**CPU 使用预设：**	fast

CRF：	16	**比特率（据内网上限设置）：**	60000kbps
		缓冲大小：	3000

关键帧间隔： 0（自动）

Profile： high

Tune： grain

x264 编码器参数

cabac=1 ref=4 deblock=0:−1 me=hex subme=4 psy=0 psy_rd=0:0 me_range=12 chroma_me=0 deadzone−inter=8 deadzone−intra=4 chroma_qp_offset=−2 dct−decimate=0 bframes=9 b−pyramid=2 b−adapt=1 open_gop=0 scenecut=35 mbtree=1 qcomp=0.7 aq−mode=2 aq−strength=0.8

性能测量

根据分辨率、帧率等参数计算 CPU 算力余量，只要录屏剩下的余量足够运行要录制的程序即可保证流畅

- 3840x2160@60fps，Cinebench R23 多核减 4000~10000 分
- 2560x1440@60fps，Cinebench R23 多核减 2000~5000 分

额外考量

- 使用弱降噪限制码率（`nr=100`，整数范围 0~1000）
- 推流也使用码率分配更均衡的 CRF 模式
- 在快速预设中添加慢速预设才有的 `partitions=all`，但需要预先测试性能损失是否合理
- 性能不够时，进一步通过 `me=dia me_range=8` 降低占用

说明

使用快速预设 `trellis=1` 时，`deadzone-inter=8 deadzone-intra=4` 会起效，否则无效（`trellis=2` 实际上更好）