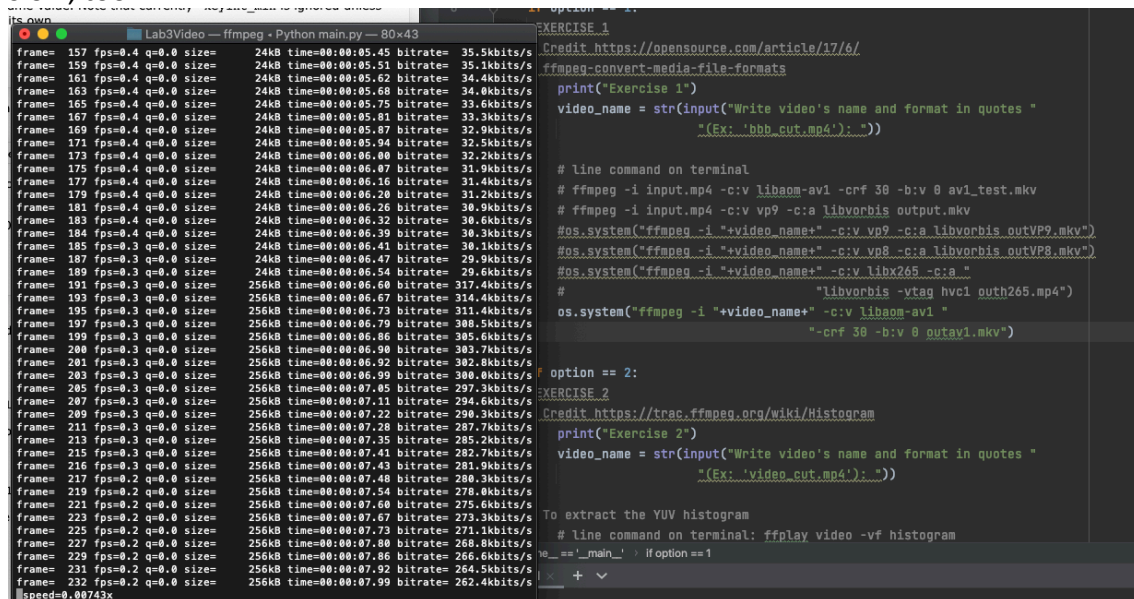


Lab 3 Video Encoding

Part 1: Converting the video

I struggled when converting the video to the AV1 format. The processing speed was very slow. For the 360-frames video, each frame was taking more than 7 seconds to be computed (and increasing approx. 1 second each 15 frames), which is extremely long compared to the time taken to convert to VP9, VP8 and h265 (10-12 seconds for the whole video). My computer was consuming many resources and it had been computing for more than 20 minutes and still 100 frames were remaining, so I decided to stop the execution. I tried reducing the quality of the output, but the computations were very slow, too.



The screenshot shows a terminal window with two main sections. The left section displays the output of an ffmpeg command, showing progress for frames 157 to 232. The right section contains a Python script for 'EXERCISE 1' that prompts the user for a video name and format, then executes an ffmpeg command to convert the video to AV1 format. The script includes comments for the terminal command and the ffmpeg command.

```
frames 157 fps=0.4 q=0.0 size= 24kB time=00:00:05.45 bitrate= 35.5kbits/s
frames 159 fps=0.4 q=0.0 size= 24kB time=00:00:05.51 bitrate= 35.1kbits/s
frames 161 fps=0.4 q=0.0 size= 24kB time=00:00:05.62 bitrate= 34.4kbits/s
frames 163 fps=0.4 q=0.0 size= 24kB time=00:00:05.68 bitrate= 34.0kbits/s
frames 165 fps=0.4 q=0.0 size= 24kB time=00:00:05.75 bitrate= 33.6kbits/s
frames 167 fps=0.4 q=0.0 size= 24kB time=00:00:05.81 bitrate= 33.3kbits/s
frames 169 fps=0.4 q=0.0 size= 24kB time=00:00:05.87 bitrate= 32.9kbits/s
frames 171 fps=0.4 q=0.0 size= 24kB time=00:00:05.94 bitrate= 32.5kbits/s
frames 173 fps=0.4 q=0.0 size= 24kB time=00:00:06.00 bitrate= 32.2kbits/s
frames 175 fps=0.4 q=0.0 size= 24kB time=00:00:06.07 bitrate= 31.9kbits/s
frames 177 fps=0.4 q=0.0 size= 24kB time=00:00:06.16 bitrate= 31.4kbits/s
frames 179 fps=0.4 q=0.0 size= 24kB time=00:00:06.20 bitrate= 31.2kbits/s
frames 181 fps=0.4 q=0.0 size= 24kB time=00:00:06.26 bitrate= 30.9kbits/s
frames 183 fps=0.4 q=0.0 size= 24kB time=00:00:06.32 bitrate= 30.6kbits/s
frames 184 fps=0.4 q=0.0 size= 24kB time=00:00:06.39 bitrate= 30.3kbits/s
frames 185 fps=0.3 q=0.0 size= 24kB time=00:00:06.41 bitrate= 30.1kbits/s
frames 187 fps=0.3 q=0.0 size= 24kB time=00:00:06.47 bitrate= 29.9kbits/s
frames 189 fps=0.3 q=0.0 size= 24kB time=00:00:06.54 bitrate= 29.6kbits/s
frames 191 fps=0.3 q=0.0 size= 25kB time=00:00:06.60 bitrate= 317.4kbits/s
frames 193 fps=0.3 q=0.0 size= 25kB time=00:00:06.67 bitrate= 314.4kbits/s
frames 195 fps=0.3 q=0.0 size= 25kB time=00:00:06.73 bitrate= 311.4kbits/s
frames 197 fps=0.3 q=0.0 size= 25kB time=00:00:06.79 bitrate= 308.5kbits/s
frames 199 fps=0.3 q=0.0 size= 25kB time=00:00:06.86 bitrate= 305.6kbits/s
frames 200 fps=0.3 q=0.0 size= 25kB time=00:00:06.90 bitrate= 303.7kbits/s
frames 201 fps=0.3 q=0.0 size= 25kB time=00:00:06.92 bitrate= 302.8kbits/s
frames 203 fps=0.3 q=0.0 size= 25kB time=00:00:06.99 bitrate= 300.8kbits/s
frames 205 fps=0.3 q=0.0 size= 25kB time=00:00:07.05 bitrate= 297.3kbits/s
frames 207 fps=0.3 q=0.0 size= 25kB time=00:00:07.11 bitrate= 294.6kbits/s
frames 209 fps=0.3 q=0.0 size= 25kB time=00:00:07.22 bitrate= 290.3kbits/s
frames 211 fps=0.3 q=0.0 size= 25kB time=00:00:07.28 bitrate= 287.7kbits/s
frames 213 fps=0.3 q=0.0 size= 25kB time=00:00:07.35 bitrate= 285.2kbits/s
frames 215 fps=0.3 q=0.0 size= 25kB time=00:00:07.41 bitrate= 282.7kbits/s
frames 216 fps=0.3 q=0.0 size= 25kB time=00:00:07.43 bitrate= 281.9kbits/s
frames 217 fps=0.2 q=0.0 size= 25kB time=00:00:07.48 bitrate= 280.3kbits/s
frames 219 fps=0.2 q=0.0 size= 25kB time=00:00:07.54 bitrate= 278.8kbits/s
frames 221 fps=0.2 q=0.0 size= 25kB time=00:00:07.60 bitrate= 275.6kbits/s
frames 223 fps=0.2 q=0.0 size= 25kB time=00:00:07.67 bitrate= 273.3kbits/s
frames 225 fps=0.2 q=0.0 size= 25kB time=00:00:07.73 bitrate= 271.1kbits/s
frames 227 fps=0.2 q=0.0 size= 25kB time=00:00:07.80 bitrate= 268.8kbits/s
frames 229 fps=0.2 q=0.0 size= 25kB time=00:00:07.86 bitrate= 266.6kbits/s
frames 231 fps=0.2 q=0.0 size= 25kB time=00:00:07.92 bitrate= 264.5kbits/s
frames 232 fps=0.2 q=0.0 size= 25kB time=00:00:07.99 bitrate= 262.4kbits/s
speed=0.00743x
```

```
EXERCISE 1
Credit https://opensource.com/article/17/6/
ffmpeg-convert-media-file-formats
print("Exercise 1")
video_name = str(input("Write video's name and format in quotes "
"(Ex: 'bbb_cut.mp4'): "))

# Line command on terminal
# ffmpeg -i input.mp4 -c:v libaom-av1 -crf 30 -b:v 0 av1_test.mkv
# ffmpeg -i input.mp4 -c:v vp9 -c:a libvorbis output.mkv
#os.system("ffmpeg -i "+video_name+" -c:v vp9 -c:a libvorbis outVP9.mkv")
#os.system("ffmpeg -i "+video_name+" -c:v vp8 -c:a libvorbis outVP8.mkv")
#os.system("ffmpeg -i "+video_name+" -c:v libx265 -c:a "
"libvorbis -vtag hvc1 outh265.mkv")
#os.system("ffmpeg -i "+video_name+" -c:v libaom-av1 "
"-crf 30 -b:v 0 outav1.mkv")

option == 2:
EXERCISE 2
Credit https://trac.ffmpeg.org/wiki/Histogram
print("Exercise 2")
video_name = str(input("Write video's name and format in quotes "
"(Ex: 'video_cut.mp4'): "))

To extract the YUV histogram
# Line command on terminal: ffmpeg video -vf histogram
if __name__ == '__main__':
    if option == 1:
```

Terminal computations using the uncommented command on the right ("ffmpeg -i "+video_name+" -c:v libaom-av1 -crf 63 -b:v 0 outav1.mkv")

Part 2: Comparison between VP8 and VP9

When both videos are played simultaneously with the ffmpeg command, we cannot see visual differences between them. However, opening them separately show that the VP9 video has a greater image quality. This is explained by the fact that the size of the VP8 video is 1.1 MB, while the size of the VP9 is 3.5 MB (the original bbb_cut size is 4.7 MB). VP9 also reduces the bitrate because, having the same bitrate, obtains better quality than the VP8.

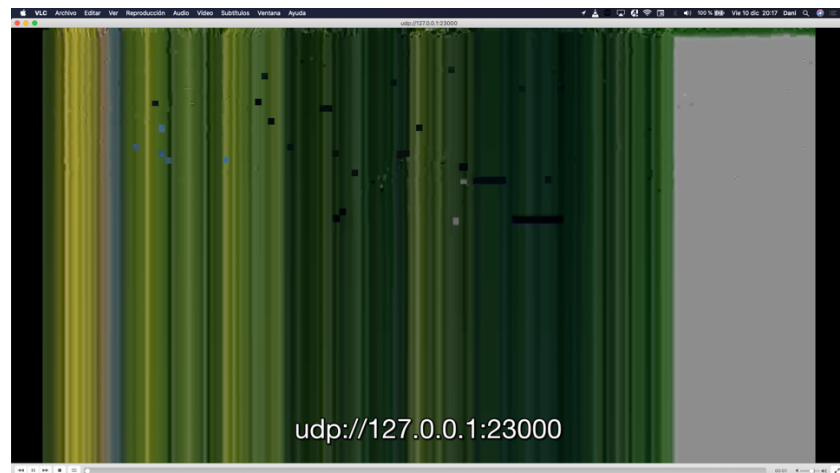
Part 3: Streaming with FFMPEG

After starting the streaming and opening the IP address on the VLC player, the result is not the desired one: The video starts playing but gets frozen. It looks like the player cannot read the bitrate from the IP fast enough. Nonetheless, when asking my

classmates about this result, they tell me that they have the same issue. I guess it is a VLC problem, and not a problem related to my implementation.



Opening the stream in the VLC player



VLC playing the stream

Despite the bad result on the VLC player, when playing the video with ffmpeg, I obtain a correct visualization. This makes me think that the stream is correctly set.



Stream played on the terminal with ffmpeg