5C1 Video Processing : Assignment II

Cormac Madden

19 April 2023

---

1. To create each of the videos at different resolutions and bit-rates I used a python script, which performed a series of operations on each of the source video files. The resolutions and bitrates used can be seen in Figure 1. The full process is shown in the block diagram in Figure 2. The algorithm first converts the raw YUV file to an MP4 file using lossly encoding, at a given constant bit-rate. It uses the H.264 video compression standard and uses ffmpeg's "slow" preset. The MP4 file is then converted back to a YUV file using lossless encoding. The third step in the algorithm is to upscale the resolution to 720p. This is done so that each of the files can be evenly compared with the original 720p video file. The compressed, encoded, up-scaled file is then compared with the original file using the python FFMPEG Quality Metrics package. The package calculates the average SSIM and PSNR of each video and appends it to an array so it can be displayed in a graph later. PSNR is calculated by computing the Mean Square Error (MSE) between the original frames and the compressed frames. The MSE is the sum of the squared differences between each pixel in the two frames, divided by the total number of pixels. The MSE is then used in the PSNR formula (1) with the Max value which is typically 255 for a 8-bit video.
$$PSNR = 10 * log10(Max^2/MSE) \tag{1}$$

2. The R/D plot in Figure 5, shows the RD curves for each of the representations. The points highlighted in red are the estimations for the crossover bitrates i.e. the maximum bit-rate where the quality of representations at 138p and 274p are greater than 274p and 548p respectively.

3.
   - At 180p I would chose to represent the video with bit-rate of 96Kbs and which would result in a PSNR value of 28.5dB. Despite the 360p file having a higher PSNR value at the same bit-rate, the artifacts in the 360p file are too blocky and distracting at that bit-rate (See Figure 3). I also qualitatively compared the videos at 256Kbps and at that stage 360p is definitely clearer. 96 Kbs probably wouldn't be the result if you used a convex hull approach to find the best bit-rate, however I think a lower bit-rate of 64Kbps, would just be unwatchable in terms of quality.
   - For the 360p representation I would choose the 612Kbps bit-rate which would result in a PSNR of 39dB. I decided on this bit-rate for several reasons. First the quality appears

Size		Suggested Bitrates for RD curve (Kbps)						
720p	1280×548	512	1024	2048	3072			
360p	640×274	96	128	256	384	512	1024	2048
180p	320×138	64	96	128	256	512	1024	

Figure 1: Bitrates and Resolutions used for creating RD curve.

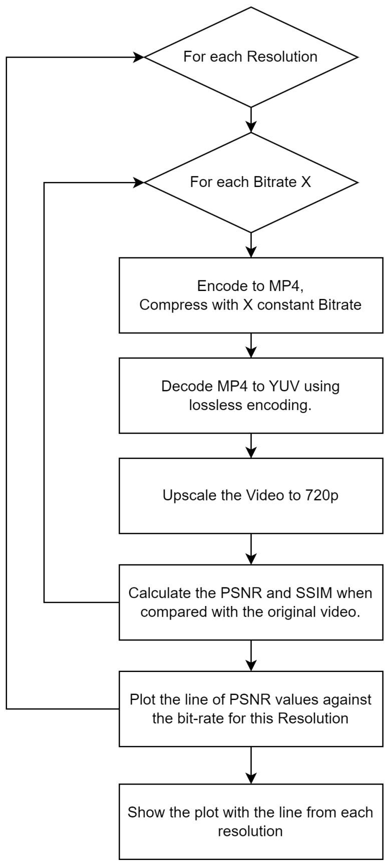

For each Resolution
For each Bitrate X
Encode to MP4, Compress with X constant Bitrate
Decode MP4 to YUV using lossless encoding.
Upscale the Video to 720p
Calculate the PSNR and SSIM when compared with the original video.
Plot the line of PSNR values against the bit-rate for this Resolution
Show the plot with the line from each resolution

Figure 2: Comparison of 138p (left) and 274p (right) both at 96Kbps.

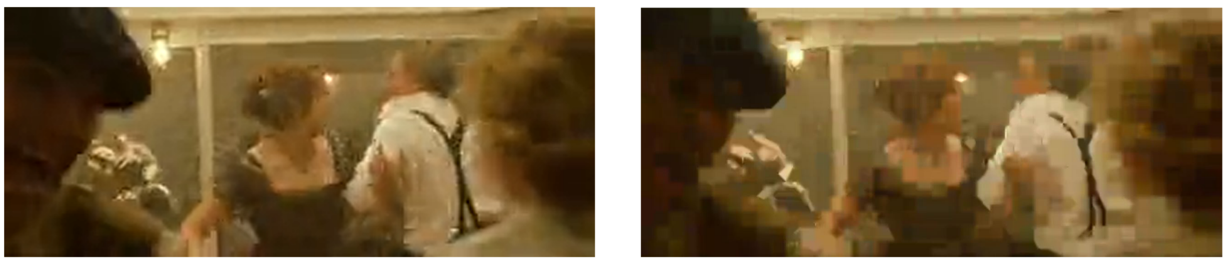


Figure 3: Comparison of 138p (left) and 274p (right) both at 96Kbps.

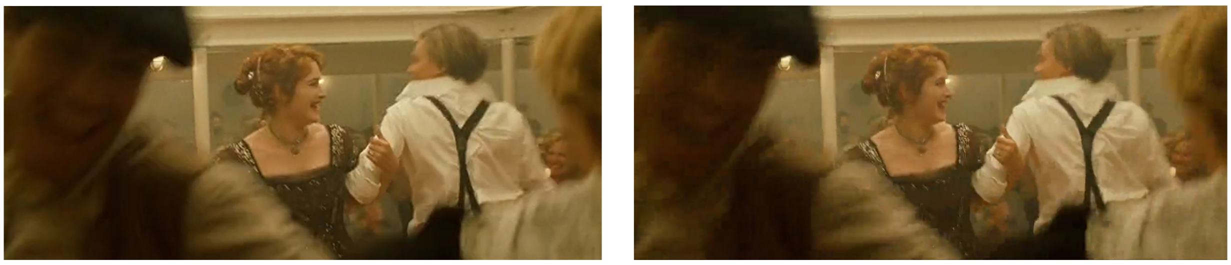


Figure 4: Comparison of 274p (left) and 720p (Right) both at 1024Kbps.

comparable to the 720p version at the PSNR crossover point. 612Kbps is closer to the higher crossover point than the lower, but in this case that seems appropriate as the next jump in quality and bit-rate is so large.

- For the 720p representation I would chose the 3Mbps bit-rate which would result in a PSNR of around 44dB. My justification for this is that first, there is a noticeable quality difference between the 2048Kbps and the 3MBps version, but you need to look very closely at paused frames to see it. I also thought that because this Movie is a high value production blockbuster film, the expectation is that it could be streamed at a very high quality, and because of that the extra bandwidth for the marginal increase in quality seems worth it.

4. I would estimate that the threshold at which 180p is better than 360p is somewhere between 128Kbps and 256Kbps. At 128Kbps the 180p version looks better, whereas at 256Kbps the 360p version looks better despite the PSNR graph favoring the the 360p version at both resolutions. I would estimate 200Kbps is the appropriate threshold bit-rate form 180p to 360p. I would estimate that the threshold between the 360p and 720p is around the 1024Kbps mark. This can be seen again in Figure 4 with the blocking on the male in the foreground of the 720p image. This threshold is difficult to establish because the PSNR gaph cannot be fully trusted. It gives a guide of what values to check but if it was followed strictly you would end up wasting bandwidth on higher resolutions despite the lower resolutions performing better for a while before the graphs shows they should.

5. A target bit-rate that is halfway between the thresholds estimated above is around 400Kbps (B_l). Assuming that a video streaming company set their default target bit-rate for their 360p representation (V_l) at 400Kbps, their PSNR would be 31.564dB.

6. This representation would be lower in picture quality and have a lower bit-rate than my 612Kbps, 39dB PSNR representation. I do think that the streaming company's representation is reasonable and doesn't waste quality by using the wrong resolution at their specified bit-rate. However, I think the streaming service should incorporate multi-resolution adaptive bit-rate streaming (ABR). This would allow them to have more options with regards to choosing the right bit-rate for their end user.

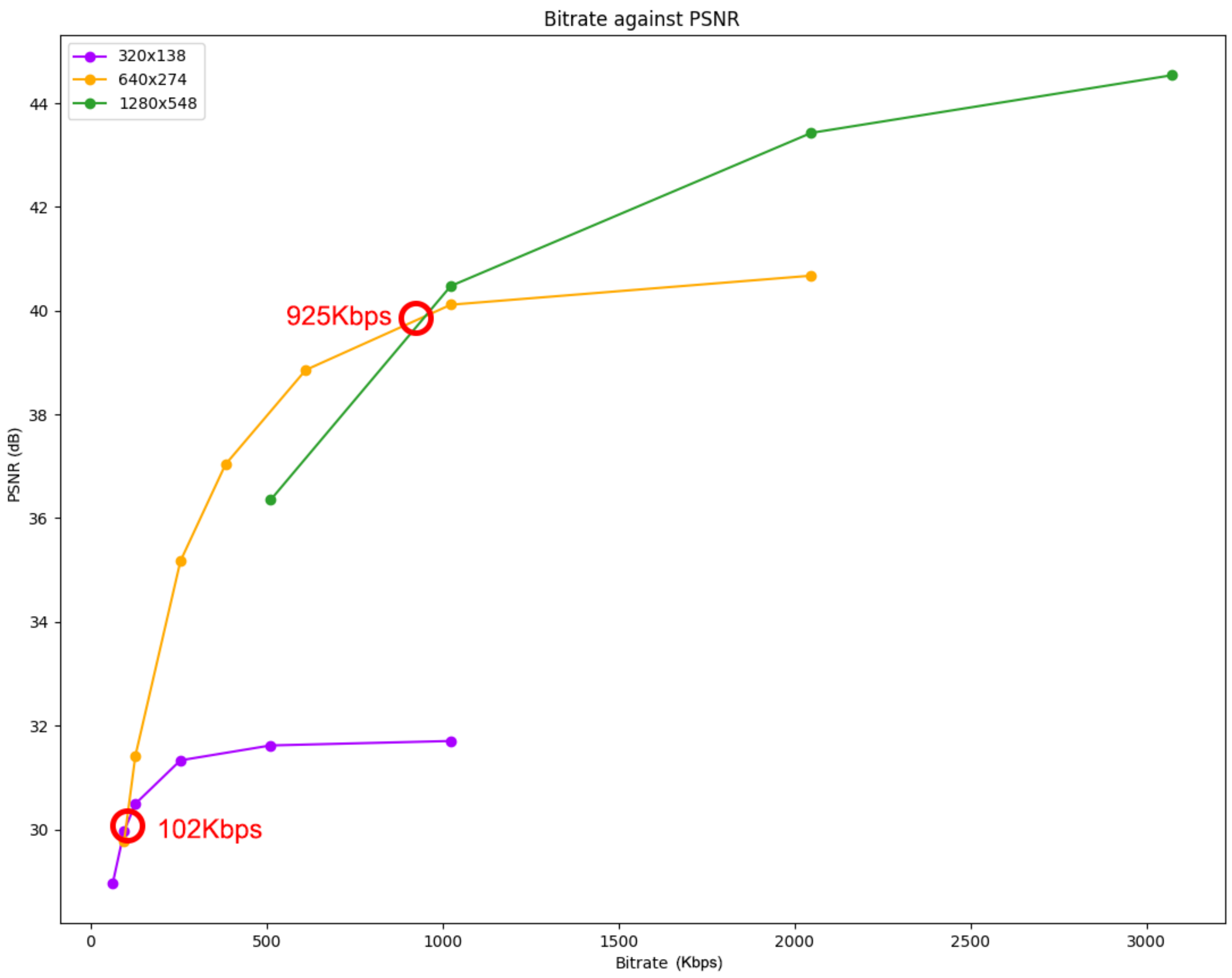
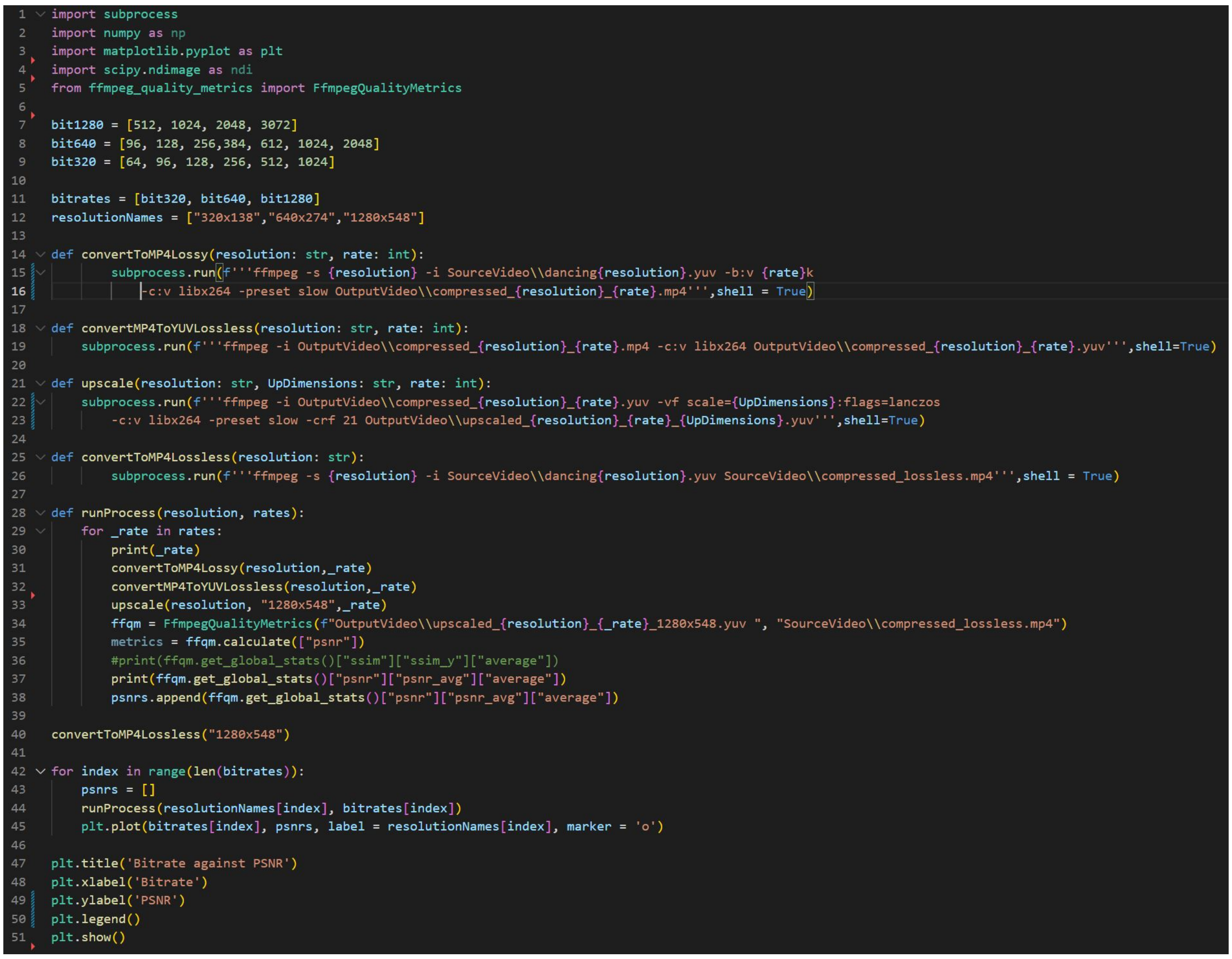


Figure 5: The RD Curves for three representations are shown as well as crossover bitrates.



```python
import subprocess
import numpy as np
import matplotlib.pyplot as plt
import scipy.ndimage as ndi
from ffmpeg_quality_metrics import FfmpegQualityMetrics

bit1280 = [512, 1024, 2048, 3072]
bit640 = [96, 128, 256,384, 612, 1024, 2048]
bit320 = [64, 96, 128, 256, 512, 1024]

bitrates = [bit320, bit640, bit1280]
resolutionNames = ["320x138","640x274","1280x548"]

def convertToMP4Lossy(resolution: str, rate: int):
        subprocess.run(f'''ffmpeg -s {resolution} -i SourceVideo\\dancing{resolution}.yuv -b:v {rate}k
            -c:v libx264 -preset slow OutputVideo\\compressed_{resolution}_{rate}.mp4''',shell = True)

def convertMP4ToYUVLossless(resolution: str, rate: int):
    subprocess.run(f'''ffmpeg -i OutputVideo\\compressed_{resolution}_{rate}.mp4 -c:v libx264 OutputVideo\\compressed_{resolution}_{rate}.yuv''',shell=True)

def upscale(resolution: str, UpDimensions: str, rate: int):
    subprocess.run(f'''ffmpeg -i OutputVideo\\compressed_{resolution}_{rate}.yuv -vf scale={UpDimensions}:flags=lanczos
        -c:v libx264 -preset slow -crf 21 OutputVideo\\upscaled_{resolution}_{rate}_{UpDimensions}.yuv''',shell=True)

def convertToMP4Lossless(resolution: str):
        subprocess.run(f'''ffmpeg -s {resolution} -i SourceVideo\\dancing{resolution}.yuv SourceVideo\\compressed_lossless.mp4''',shell = True)

def runProcess(resolution, rates):
    for _rate in rates:
        print(_rate)
        convertToMP4Lossy(resolution,_rate)
        convertMP4ToYUVLossless(resolution,_rate)
        upscale(resolution, "1280x548",_rate)
        ffqm = FfmpegQualityMetrics(f"OutputVideo\\upscaled_{resolution}_{_rate}_1280x548.yuv ", "SourceVideo\\compressed_lossless.mp4")
        metrics = ffqm.calculate(["psnr"])
        #print(ffqm.get_global_stats()["ssim"]["ssim_y"]["average"])
        print(ffqm.get_global_stats()["psnr"]["psnr_avg"]["average"])
        psnrs.append(ffqm.get_global_stats()["psnr"]["psnr_avg"]["average"])

convertToMP4Lossless("1280x548")

for index in range(len(bitrates)):
    psnrs = []
    runProcess(resolutionNames[index], bitrates[index])
    plt.plot(bitrates[index], psnrs, label = resolutionNames[index], marker = 'o')

plt.title('Bitrate against PSNR')
plt.xlabel('Bitrate')
plt.ylabel('PSNR')
plt.legend()
plt.show()
```

Figure 6: The code used to generate the outputs.