

P1: JPEG and MPEG

Q1: What do you remember from JPG, MPEG and MPEG2?

JPEG is a commonly used image compression standard. It is a lossy compression method designed to reduce the file size of digital images while preserving visual quality to an acceptable level. It achieves compression by analyzing and removing redundant image data, primarily in areas where the human eye is less sensitive to small details. The file extension for JPEG images is ".jpg."

MPEG is a group of standards developed by the Moving Picture Experts Group, focusing on the compression and encoding of audio and video data. Its standards are widely used for digital video and audio compression, transmission, and storage. The most well-known MPEG standard is MPEG-2, which is commonly used in DVD video compression and broadcasting.

MPEG-2 is a video compression standard developed by the Moving Picture Experts Group and is the successor to MPEG-1. It provides high-quality video compression with relatively low data rates, making it suitable for broadcast and distribution of digital video content.

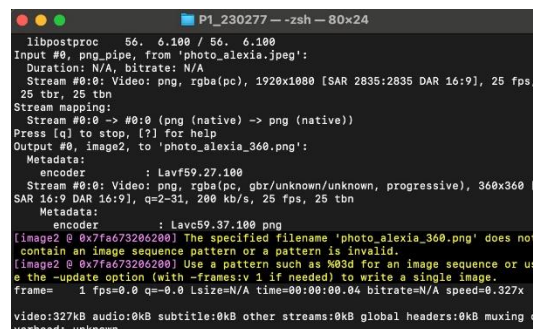
1. Start a script called `rgb_yuv.py` and create a translator from 3 values in RGB into the 3 YUV values, plus the opposite operation. You can choose the 3 values, or open them from a text file, receive it from command line... feel free. Put it in a method

Done in the file: `rgb_yuv.py`

2. Use `ffmpeg` to resize images into lower quality. Use any image you like

I have chosen an image of Alexia Putellas celebrating a goal. The command used to convert the image into a 360x360 is:

`ffmpeg-i photo_ex2.jpeg-vf scale=360:360 photo_ex2_360x360.png`



```
P1_230277 -- zsh -- 80x24
libpostproc 56.6.100 / 56.6.100
Input #0, png_pipe, from 'photo_ex2.jpeg':
  Duration: N/A, bitrate: N/A
  Stream #0:0: Video: png, rgb(pc), 1920x1080 [SAR 2835:2835 DAR 16:9], 25 fps,
  25 tbr, 25 tbn
Stream mapping:
  Stream #0:0 -> #0:0 (png (native) -> png (native))
Press [q] to stop, [?] for help
Output #0, image2, to 'photo_ex2_360.png':
  Metadata:
    encoder         : Lavf59.27.100
  Stream #0:0: Video: png, rgb(pc, gbr/unknown/unknown, progressive), 360x360 [
  SAR 16:9 DAR 16:9], q=2-31, 200 kb/s, 25 fps, 25 tbn
  Metadata:
    encoder         : Lavc59.37.100 png
[image2 @ 0x7fa673206200] The specified filename 'photo_ex2_360.png' does not
contain an image sequence pattern or a pattern is invalid.
[image2 @ 0x7fa673206200] Use a pattern such as %03d for an image sequence or us
e the -update option (with -frames:v 1 if needed) to write a single image.
frame= 1 fps=0.0 q=0.0 Lsize=N/A time=00:00:00.04 bitrate=N/A speed=0.327x
video:327kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing o
verhead: unknown
```

Finally I have converted the image to a smaller format and by doing so, it has lost quality as expected since the pixel density is smaller.

Here we see the difference between the original and the small one:



360x360



Original Image

Now, create a method in previous script to automatise this order

I created the method in the same file as before: `rgb_yuv.py` (#Ex2)

3. Create a method called serpentine which should be able to read the bytes of a JPEG file in the serpentine way we saw.

I created the method in the file: E3.py

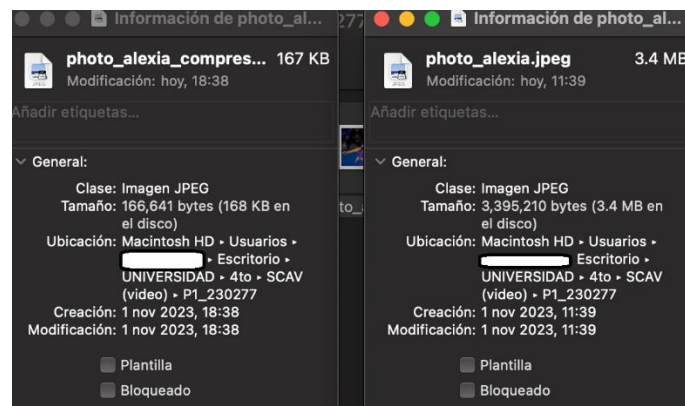
4. Use FFMPEG to transform the previous image into b/w. Do the hardest compression you can. Add everything into a new method and comment the results

I created the method in the file: E4.py

First of all, I compressed Alexia's image using the terminal with the command as follows:

```
P1_230277 -- -zsh -- 80x24
[swscaler @ 0x7ff208de4000] [swscaler @ 0x7ff208f1e000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x7ff208de4000] [swscaler @ 0x7ff208f0f000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x7ff208de4000] [swscaler @ 0x7ff208f00000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x7ff208de4000] [swscaler @ 0x7ff208f00000] deprecated pixel format used, make sure you did set range correctly
Output #0: image2, to 'photo_alexia_compressed_ex3.jpeg':
Metadata:
  encoder      : Lavf59.27.100
  Stream #0:0: Video: mjpeg, yuvj444p(pc, progressive), 1920x1080 [SAR 1:1 DAR 16:9], q=2-31, 200 kb/s, 25 fps, 25 tbn
Metadata:
  encoder      : Lavc59.37.100 mjpeg
Side data:
  cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: N/A
[image2 @ 0x7ff20800140] The specified filename 'photo_alexia_compressed_ex3.jpg' does not contain an image sequence pattern or a pattern is invalid.
[image2 @ 0x7ff20800140] Use a pattern such as %03d for an image sequence or use the -update option (with -frames:v 1 if needed) to write a single image.
frame= 1 fps=0.0 q=7.2 Lsize=N/A time=00:00:00.04 bitrate=N/A speed=0.331x
video:163kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
```

As we can see, the new image occupies less than the original image:



Then finally, using the script: E4.py we get the image in bw:



5. Create a method which applies a run-length encoding from a series of bytes given

I created the method in the file: E5.py

6. Create a class which can convert, can decode (or both) an input using the DCT. Not necessary a JPG encoder or decoder. A class only about DCT is OK too

I created the method in the file: E6.py