Paula Osés Noguero
NIA: 207310

P1: JPEG

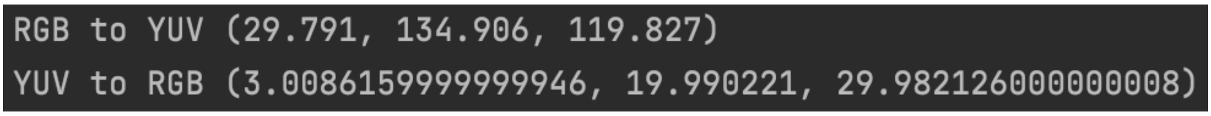
In this Lab we had to resolve 5 exercises using python and ffmpeg. For this I have created a Python project using PyCharm.

Exercise 1:

For the first exercise we were asked to 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.

In my case I have created the file *rgb_yuv.py* with two functions, *rgb_to_yuv()* that transforms RGB values to YUV values and *yuv_to_rgb()* that does the inverse operation. Then I call these functions from the main having as arguments the RGB and YUV values.

In the next image you can see the result of each function,



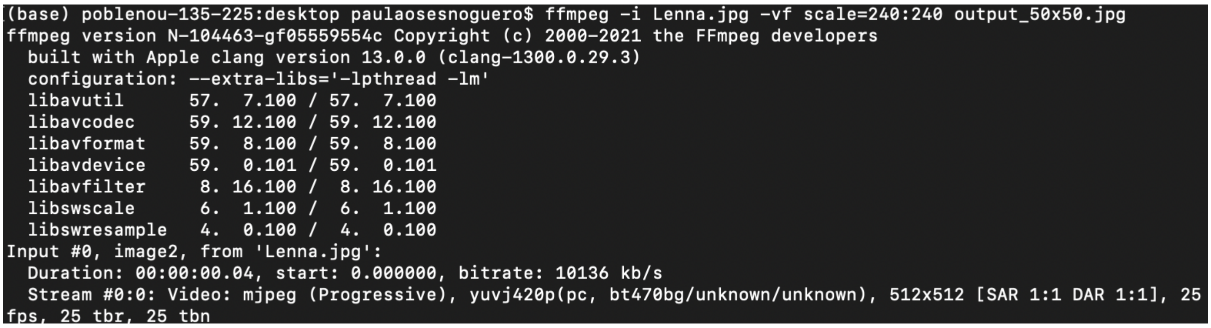
```
RGB to YUV (29.791, 134.906, 119.827)
YUV to RGB (3.0086159999999946, 19.990221, 29.982126000000008)
```

Exercise 2:

In this exercise we were asked to use ffmpeg to resize images into lower quality. To do so we run from our terminal *ffmpeg* and enter the following command.

```
ffmpeg -i Lenna.jpg -vf scale=240:240 output_50x50.jpg
```

Here we see the process after running the command.



```
(base) poblenou-135-225:desktop paulaosesnoguero$ ffmpeg -i Lenna.jpg -vf scale=240:240 output_50x50.jpg
ffmpeg version N-104463-gf05559554c Copyright (c) 2000-2021 the FFmpeg developers
  built with Apple clang version 13.0.0 (clang-1300.0.29.3)
  configuration: --extra-libs='-lpthread -lm'
  libavutil      57.  7.100 / 57.  7.100
  libavcodec     59. 12.100 / 59. 12.100
  libavformat    59.  8.100 / 59.  8.100
  libavdevice    59.  0.101 / 59.  0.101
  libavfilter     8. 16.100 /  8. 16.100
  libswscale      6.  1.100 /  6.  1.100
  libswresample   4.  0.100 /  4.  0.100
Input #0, image2, from 'Lenna.jpg':
  Duration: 00:00:00.04, start: 0.000000, bitrate: 10136 kb/s
  Stream #0:0: Video: mjpeg (Progressive), yuvj420p(pc, bt470bg/unknown/unknown), 512x512 [SAR 1:1 DAR 1:1], 25
fps, 25 tbr, 25 tbn
```

The result we get is the following,

Original Image (51 KB)

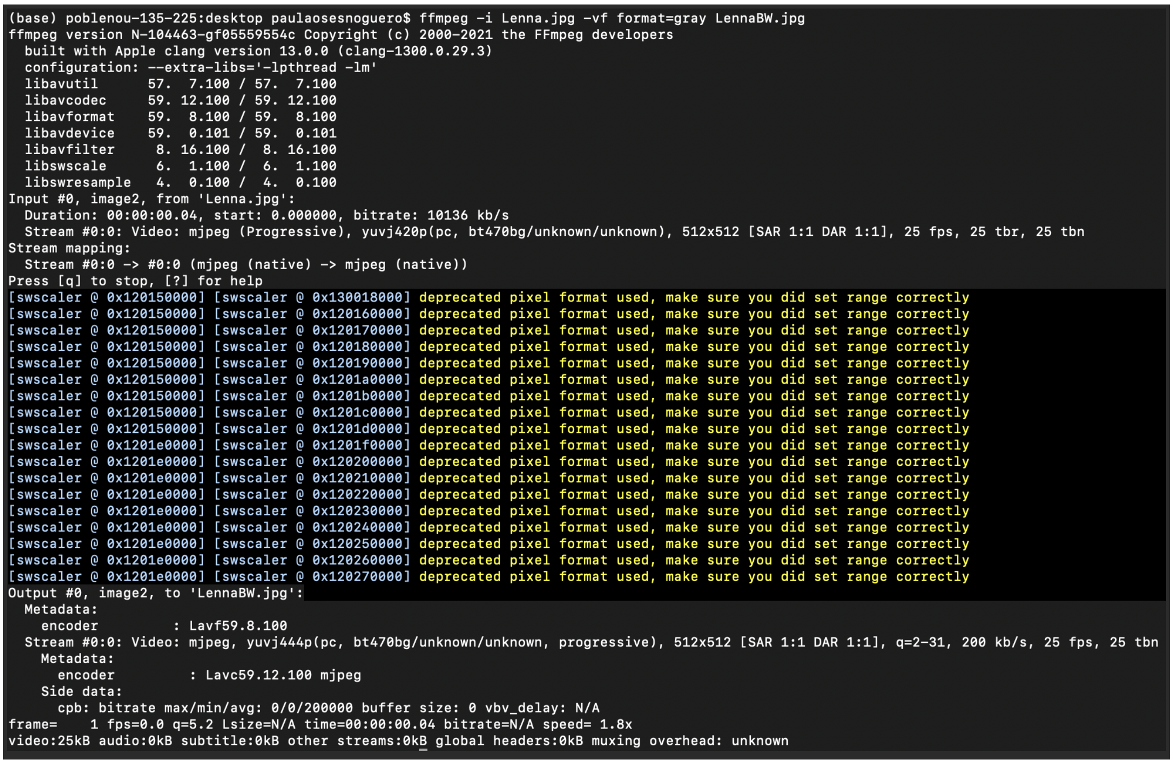
Resized Image (11 KB)

Exercise 3:

Now we use FFMPEG to transform the Lenna image into b/w. For this we use the following command,

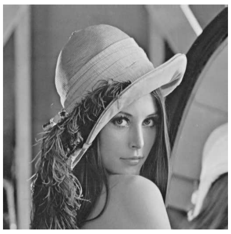
```
ffmpeg -i Lenna.jpg -vf format=gray LennaBW.jpg
```

Here we see the image of the terminal.



```
(base) poblenou-135-225:desktop paulaosesnoguero$ ffmpeg -i Lenna.jpg -vf format=gray LennaBW.jpg
ffmpeg version N-104463-gf05559554c Copyright (c) 2000-2021 the FFmpeg developers
  built with Apple clang version 13.0.0 (clang-1300.0.29.3)
  configuration: --extra-libs='-lpthread -lm'
  libavutil      57.  7.100 / 57.  7.100
  libavcodec     59. 12.100 / 59. 12.100
  libavformat    59.  8.100 / 59.  8.100
  libavdevice    59.  0.101 / 59.  0.101
  libavfilter     8. 16.100 /  8. 16.100
  libswscale      6.  1.100 /  6.  1.100
  libswresample   4.  0.100 /  4.  0.100
Input #0, image2, from 'Lenna.jpg':
  Duration: 00:00:00.04, start: 0.000000, bitrate: 10136 kb/s
  Stream #0:0: Video: mjpeg (Progressive), yuvj420p(pc, bt470bg/unknown/unknown), 512x512 [SAR 1:1 DAR 1:1], 25 fps, 25 tbr, 25 tbn
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> mjpeg (native))
Press [q] to stop, [?] for help
[swscaler @ 0x120150000] [swscaler @ 0x130018000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x120160000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x120170000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x120180000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x120190000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x1201a0000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x1201b0000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x1201c0000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x120150000] [swscaler @ 0x1201d0000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x1201f0000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120200000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120210000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120220000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120230000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120240000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120250000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120260000] deprecated pixel format used, make sure you did set range correctly
[swscaler @ 0x1201e0000] [swscaler @ 0x120270000] deprecated pixel format used, make sure you did set range correctly
Output #0, image2, to 'LennaBW.jpg':
  Metadata:
    encoder         : Lavf59.8.100
  Stream #0:0: Video: mjpeg, yuvj444p(pc, bt470bg/unknown/unknown, progressive), 512x512 [SAR 1:1 DAR 1:1], q=2-31, 200 kb/s, 25 fps, 25 tbn
    Metadata:
      encoder         : Lavc59.12.100 mjpeg
    Side data:
      cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: N/A
frame=    1 fps=0.0 q=5.2 Lsize=N/A time=00:00:00.04 bitrate=N/A speed= 1.8x
video:25kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
```

The resulting image we get,

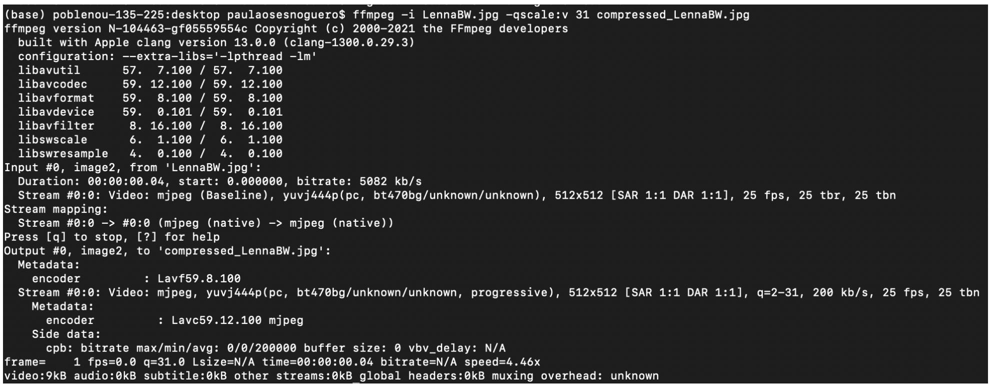


Gray scale Image 25 KB

Then we are also asked to do the hardest compression you can and comment on the results. To do so I have used the following command, as we are asked to do the hardest compression i put the value 31 which is the maximum.

```
ffmpeg -i LennaBW.jpg -qscale:v 31 compressed_LennaBW.jpg
```

The terminal,



```
(base) poblenou-135-225:desktop paulaosesnoguero$ ffmpeg -i LennaBW.jpg -qscale:v 31 compressed_LennaBW.jpg
ffmpeg version N-104463-gf05559554c Copyright (c) 2000-2021 the FFmpeg developers
  built with Apple clang version 13.0.0 (clang-1300.0.29.3)
  configuration: --extra-libs='-lpthread -lm'
  libavutil      57.  7.100 / 57.  7.100
  libavcodec     59. 12.100 / 59. 12.100
  libavformat    59.  8.100 / 59.  8.100
  libavdevice    59.  0.101 / 59.  0.101
  libavfilter     8. 16.100 /  8. 16.100
  libswscale      6.  1.100 /  6.  1.100
  libswresample   4.  0.100 /  4.  0.100
Input #0, image2, from 'LennaBW.jpg':
  Duration: 00:00:00.04, start: 0.000000, bitrate: 5082 kb/s
  Stream #0:0: Video: mjpeg (Baseline), yuvj444p(pc, bt470bg/unknown/unknown), 512x512 [SAR 1:1 DAR 1:1], 25 fps, 25 tbr, 25 tbn
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> mjpeg (native))
Press [q] to stop, [?] for help
Output #0, image2, to 'compressed_LennaBW.jpg':
  Metadata:
    encoder         : Lavf59.8.100
  Stream #0:0: Video: mjpeg, yuvj444p(pc, bt470bg/unknown/unknown, progressive), 512x512 [SAR 1:1 DAR 1:1], q=2-31, 200 kb/s, 25 fps, 25 tbn
    Metadata:
      encoder         : Lavc59.12.100 mjpeg
    Side data:
      cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: N/A
frame=    1 fps=0.0 q=31.0 Lsize=N/A time=00:00:00.04 bitrate=N/A speed=4.46x
video:9kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
```

The final result we get is the following image that occupies 9KB so we can see that it is actually compressed. We also see that we have lost a lot of quality.

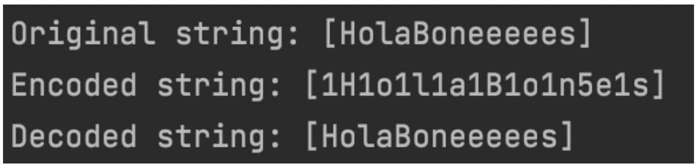
Compressed and gray scale image 9 KB

Also notice that when we are transforming the original image to gray scale we are already compressing it as we can see the original image was 51 KB, the gray scale one is 25 KB and finally the compressed one is 9 KB.

Exercise 4:

For this exercise we have to create a script which contains a function which applies a run-lenght encoding from a series of bytes given. For this I have created the file *run_lenth_alg.py.* Inside we can find the function *encode_message()* where we look at the initial string and see how many letters are the same. Then we have the function *decode_message()*, in this case we have as argument the encoded sequence and we decode it to the original.

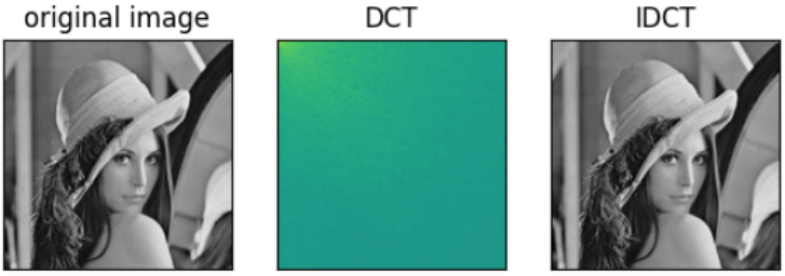
In this image you can see the outputs of the program.



```
Original string: [HolaBoneeeees]
Encoded string: [1H1o1l1a1B1o1n5e1s]
Decoded string: [HolaBoneeeees]
```

Exercise 5:

Finally we are asked to create a script which can convert, can decode (or both) an input using the DCT. For this I have created the file *dct.py* that has the function *dct_idct()*. In this function we receive a gray scale image and then apply the DCT using the formula and then we do the inverse IDCT to get the original image. Here we can see the results.


original image
DCT
IDCT