Daniel Barrero

NIA: 205446

Processament de Vídeo: Pràctica 1

En aquesta pràctica he reutilitzat d'una altra assignatura la maquina virtual d'Ubuntu, y he programat amb Atom amb aquesta guia per tal de tenir la compilació amb python i el format PEP8:

https://platzi.com/contributions/configurando-mi-editor-atom-para-python/

També he instal·lat FFmpeg amb la guia de https://trac.ffmpeg.org/

Exercici 1:

En aquest exercici he implementat la funció anomenada "rgb_to_yuv" per tal de pasar un vector de color RGB a format YUV. A més hi ha una altra funció per fer la inversa amb "yuv_to_rgb". He utilitzat les slides de teoria per tal de implementar-la.

En el main del programa provem que funciona amb un valor d'exemple i obtenim:

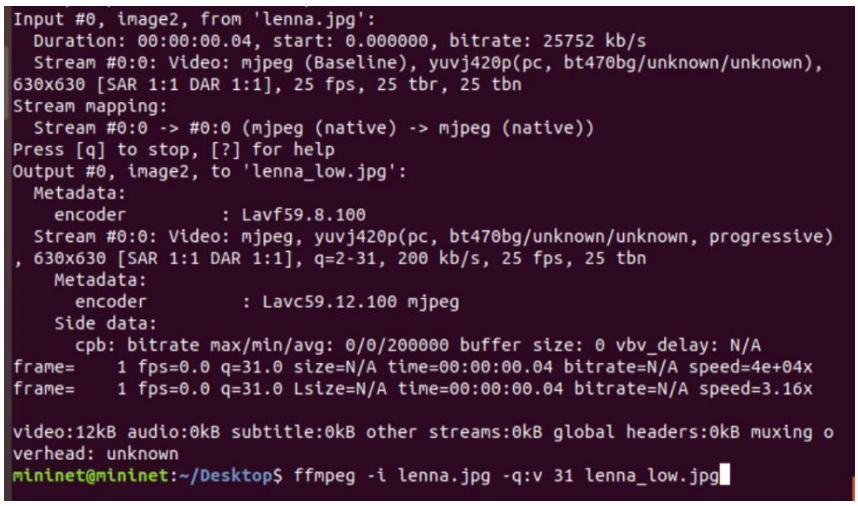
```
#RGB to YUV
RGB:[1.0, 1.0, 1.0]
YUV:[16.859, 128.0, 128.0]
RGB:[1.0, 1.0, 1.0]
```

Exercici 2:

En aquest exercici he utilitzat la llibreria de FFmpeg per tal de fer un resize de la image de Lenna per tal de tenirla en inferior qualitat. He utilitzat la següent comanda:

"ffmpeg - i lenna.jpg - q: v 31 lenna_low.jpg"

On lenna.jpg es la image input, 31 el valor modificable de la qualitat, y lenna_low.jpg la imatge resultant.



Obtenim el següent resultat on es pot veure com la qualitat de la imatge de la dreta, que es la nova, te inferior qualitat, però mantenint la mateixa resolució:

Exercici 3:

Al igual que en l'exercici anterior utilitzem FFmpeg per convertir la imatge en blanc i negre, he trobat dos mètodes:

- "ffmpeg - i lenna.jpg - vf format = gray lenna_bw.jpg"
- "ffmpeg - i lenna.jpg - vf hue = s = 0 lenna_bw2.jpg"

En el primer cas li indiquem que volem cambiar el format a blanc i negre i el segon utilitzem que una imateg en blanc i negre te el hue i la saturació igual a 0. Amb el segon obtenim una imatge amb un tamany més petit.

```
Input #0, image2, from 'lenna.jpg':
  Duration: 00:00:00.04, start: 0.000000, bitrate: 25752 kb/s
  Stream #0:0: Video: mjpeg (Baseline), yuvj420p(pc, bt470bg/unknown/unknown),
630x630 [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 @ 0x5648462e7b00] deprecated pixel format used, make sure you did set
 range correctly
[swscaler @ 0x564846305b40] deprecated pixel format used, make sure you did set
 range correctly
Output #0, image2, to 'lenna_bw.jpg':
  Metadata:
    encoder         : Lavf59.8.100
  Stream #0:0: Video: mjpeg, yuvj444p(pc, bt470bg/unknown/unknown, progressive)
, 630x630 [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=5.6 size=N/A time=00:00:00.04 bitrate=N/A speed=4e+04x
frame=    1 fps=0.0 q=5.6 Lsize=N/A time=00:00:00.04 bitrate=N/A speed=1.89x

video:27kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing o
verhead: unknown
mininet@mininet:~/Desktop$ ffmpeg -i lenna.jpg -vf format=gray  lenna_bw.jpg
```

Comparant el resultat dels dos casos veiem que el resultat és identic:

Exercici 4:

En aquest exercici implementem la codificació run-lenght per una sèrie de bytes. La funció "run_lenght" itera entre els bits fins trobar un de diferent als anteriors, quan ho fa guarda el numero de vegades que ha aparegut aquest bit i el seu valor, i comença un altre cop la iteracció pel nou valor del bit trobat. Al main provem amb un exemple i que funciona correctament:

```
#Run-Length encoding
Bytes:[1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0]
Encoded Bytes:[[1, 3], [0, 1], [1, 2], [0, 2], [1, 2], [0, 3], [1, 2], [0, 1]
```

Exercici 5:

Utilitzo les funcions "dct" i "idct" de la llibreria de "scipy":

https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.dct.html#scipy.fftpack.dct

Provem que funciona fent la dct de un array i després la idct del resultat, i veiem que el resultat es el mateix:

```
#DCT i IDCT
x:[4, 3, 5, 3, 4]
X_dct:[  8.49705831e+00  -1.40433339e-16  -2.41576517e-01   3.51083347e-17
   1.65579008e+00]
x_idct:[ 4.  3.  5.  3.  4.]
```