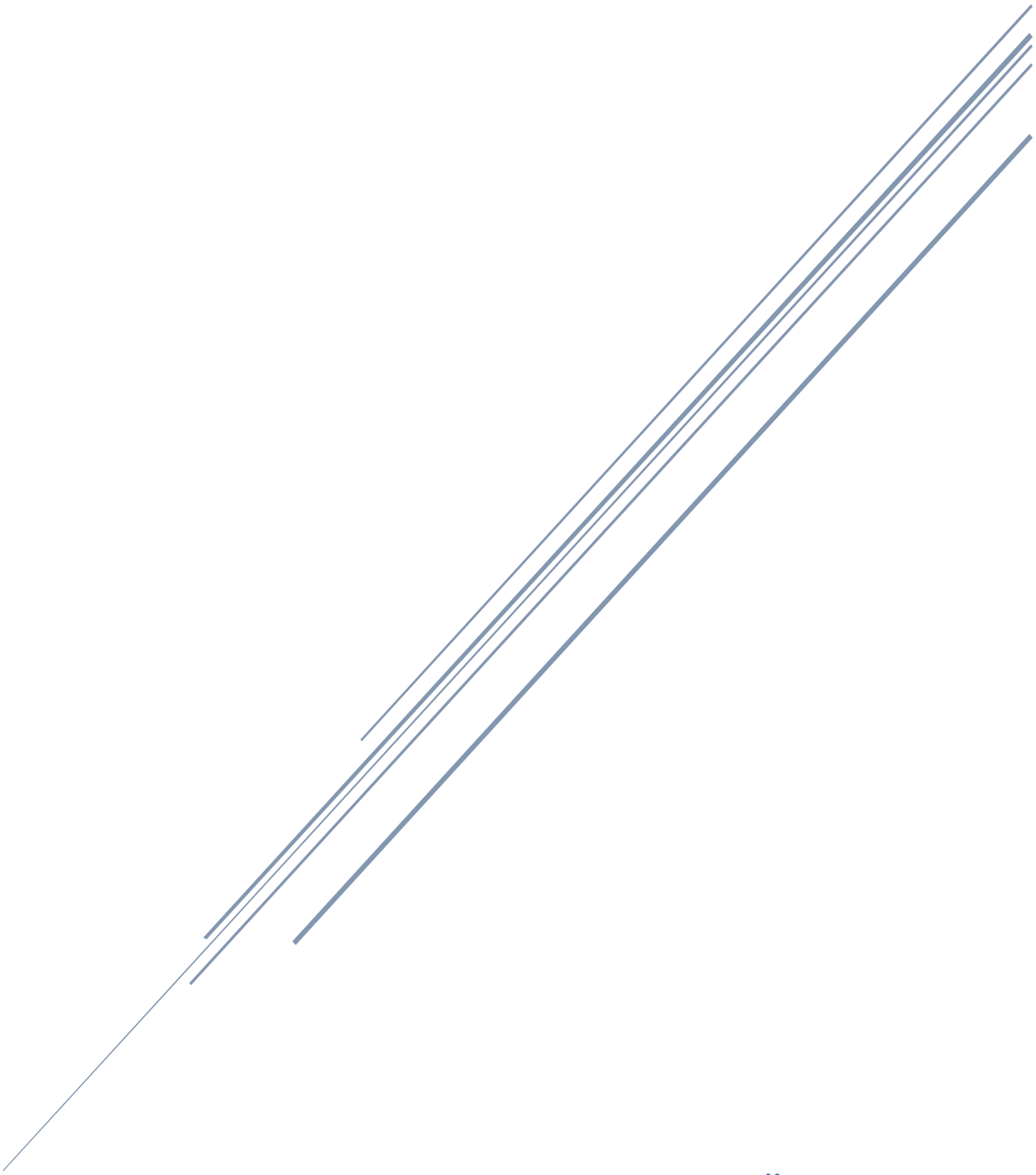
IMAGE PROCESSING -HW3

Rıdvan Demirci 141044070



Gebze Teknik Üniversitesi

Implement the jpeg compression algorithm

- Ödev için Python kullanıldı.
- Resim açıldı, orijinal görüntüyü kayıp etmemek için kopyalandı ve YUV katmanına çevrildi.

```
img = cv2.imread('horse.jpg') # read image
nimg = deepcopy(img) # copy image to calculateing error
YUVimg = convertYUC(img) # convert RGB to YUV
z1,z2,z3 = compress(YUVimg, Q) # compress return 3 array
```

- YUV renk katmanına çevrilen görüntü compress metoduna verildi.
- Compress methodu geriye 3 katman döndürür.
- Compress methodunun içinde ise;
  - Gelen görüntünün katmanları ayırılır.
  - Her katman için zig-zag tablosu oluşturulur.
  - Paralel olarak hesaplanması için threadler oluşturulur.

```
zigzagTable1 = np.zeros((math.ceil(rowN/BLOCK),math.ceil((colN/BLOCK))),dtype=object)
    zigzagTable2 = np.zeros((math.ceil(rowN/BLOCK),math.ceil((colN/BLOCK))),dtype=object)
    zigzagTable3 = np.zeros((math.ceil(rowN/BLOCK),math.ceil((colN/BLOCK))),dtype=object)
    thread1 = threading.Thread(target=calcuateParelel,args=(Y,zigzagTable1))
    thread2 = threading.Thread(target=calcuateParelel,args=(U,zigzagTable2))
    thread3 = threading.Thread(target=calcuateParelel,args=(V,zigzagTable3))
```

- **Threadler içinde ise;**

  Gelen katman 8X8 block olarak gezilir.

52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Her block ‘dan 128 çıkarılır ve DCT değeri alınır.

−415.38	−30.19	−61.20	27.24	56.12	−20.10	−20.10	0.46
4.47	−21.86	−60.76	10.25	13.15	−7.09	−8.54	4.88
−46.83	7.37	77.13	−24.56	−28.91	9.93	5.42	−5.65
−48.53	12.07	34.10	−14.76	−10.24	6.30	1.83	1.95
12.12	−6.55	−13.20	−3.95	−1.87	1.75	−2.79	3.14
−7.73	2.91	2.38	−5.94	−2.38	0.94	4.30	1.85
−1.03	0.18	0.42	−2.42	−0.88	−3.02	4.12	−0.66
−0.17	0.14	−1.07	−4.19	−1.17	−0.1	0.50	1.68

Quantization’a bölünür.

Quantization matrix için slaytlardaki matrix kullanılmıştır.

−416	−33	−60	32	48	−40	0	0
0	−24	−56	19	26	0	0	0
−42	13	80	−24	−40	0	0	0
−42	17	44	−29	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization’a bölündükten sonra ise ortaya çıkan tablo zigzag olarak gezildi ve listenin listesi gibi bir yapıya atıldı. Komple sıfır olan listeler ise silindi.

[[-416],[-33,0],[-42,-24,-60],[32,-56,13,-42],[18,19,80,19,48],[-40,26,-24,44,0,0],[0,0,0,-29,-40,0,0]]

Listenin geri kalan yapısı alınmaz atılır. Çünkü hep sıfır olduğu için…

- **Listenin listesi olan yapı bir tabloda tutulup sıkıştırma işlemi tamamlandı.**

Decode işleminde ise yine paralel olarak yapılmaktadır.

- **Zig-zag tablosundan değerler 8X8 lik Blocklara yerleştirilir.**
- **Quntization matrix tablosu ile çarpılır**
- **IDCT alınır**
- **128 ile toplanır.**
- **Ve oluşturulacak resimde yerine yazılır.**
- **Oluşturulan resim ise renk düzeyi YUV dan RGB ye geri alınır.**

Hata hesabı;

Hata hesabı orijinal resimlerinin hem ayrı ayrı katmanları ile hem de tüm katmanları toplamı ile ayrı ayrı hesaplanmıştır.

Programın çalışması için main de istenilen görüntü yolu verilmelidir.

Program bittikten sonra verilen görüntünün decode edilmiş hali ekrana gelir console da ise en son hesaplanan hata hesabı vardır.

```
Layer of Error R: 31166.20184798808
Layer of Error G: 45794.33822155986
Layer of Error B: 9452.239920516642
Total error: 28804.259996688193
```

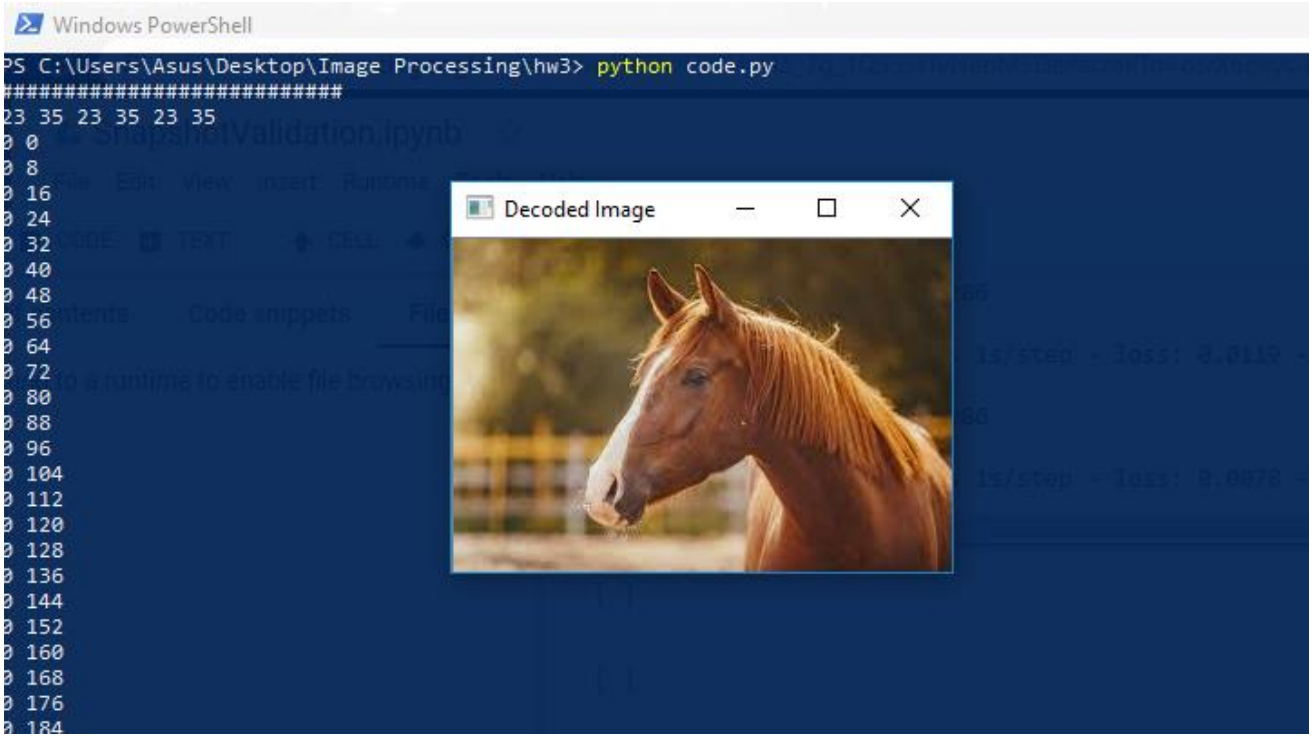
Programın çalışması uzun sürdüğü için hesaplamalar console da gösterilir.

Programın hızlanması için paralel yazılsa bile programın tamamen sonuçlanması için ortalama 1.20 dk. gerekir. (Görüntünün boyutuna göre değişmektedir.)

ÖRNEK GÖRÜNTÜLER

Before

After



Kaynakça:

- **Jpeg Compress ,wikipedia**
  http://www.wikizeroo.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvSlBFRw