Image Processıng -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.**

**Her block ‘dan 128 çıkarılır ve DCT değeri alınır.**

**Quantization’a bölünür.**

**Quantization matrix için slaytlardaki matrix kullanılmıştır.**

**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.)**

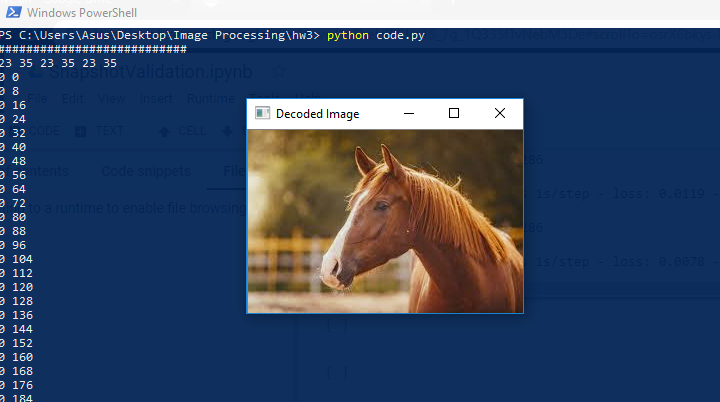
**ÖRNEK GÖRÜNTÜLER**

**After**

**Before**



****

**Kaynakça:**

* **Jpeg Compress ,wikipedia**

**http://www.wikizeroo.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvSlBFRw**