KARADENİZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERSİN ADI : AĞ VE VERİ GÜVENLİĞİ

PROJE RAPORU

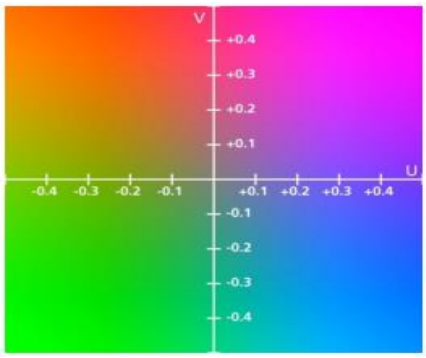
I.ÖĞRETİM ☐ II.ÖĞRETİM [X]

NUMARA	AD SOYAD
330152	Onur BUDAK
330188	Batuhan EKİCİ
330192	Hakan AKSOY

PROJE RAPORU

Görüntüleme teknolojilerindeki gelişmeler ile dijital görüntüler somut bir bilgi kaynağı haline gelmektedir. Bu süreçte, birçok resim düzenleme araçları, dijital fotoğrafçılığın orijinalliğini inceler.

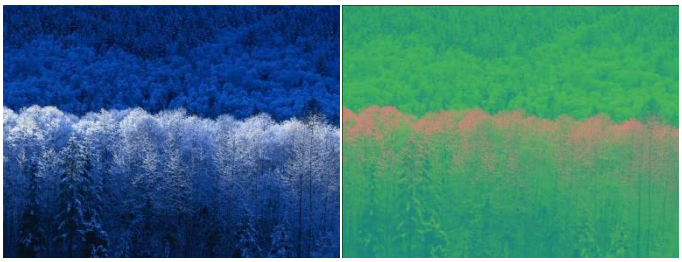
RGB uzayından YUV'a geçiş sırasında yaptığımız araştırmalarımızda; Y: bir görüntüdeki siyah-beyaz bilgisini ve U ve V ise renk bilgisini ifade etmektedir. Aşağıda U ve V eksenli sistemde bu iki bileşen arasındaki ilişki söz konusudur.



R, G, B değerleri 0 ile 255 arasındaki değerleri belirtir.
Bu değerleri kullanarak;
Y= (0.257*R)+(0.504*G)+(0.098*B)+16
U = -(0.148*R)-(0.291*G)+(0.439*B)+128
V = (0.439*R)-(0.368*G)-(0.071 *B)+128
formülleri yardımıyla YUV değerlerini hesaplayabiliriz.

Resmimizin her pikseli için bu pikseli ifade eden RGB değerine bu işlem uygulandığında o pikselin YUV değeri elde edilmiş olur.

Bu renk sisteminde Y bileşeni 0-255 arasında, U bileşeni -112 ile +112 arasında, V ise -157 ile +157 arasında değerler almaktadır. Bu sistem videolarla ilgili işlemlerde yaygın olarak kullanılabilir.



Fotoğraflar üzerindeki sahtecilik tespiti yaparken 2 farklı resim kullanıldı. Bu resimlerden ilki olan forged1.png ile forged1_maske.png resimlerini kullanarak gerekli testler yapıldı. Bu testleri yaparken yapılan karşılaştırmalar; bakılması gereken vektör sayısı, eşik değeri ve yakınlık eşiğidir.

Resim üzerindeki quantalama değeri başta bir dizi şeklinde belirlenerek düzenlendi. Daha sonra bu değer verilen kaynaktaki değer olan 16 ile değiştirilip, işlemler tekrarlandı. Zigzag vektörlerinin konumlarını bulmak için 16+2 (x-y koordinat) değerli dizi tasarlanmıştır.

```
im.shape
```
```
(256, 256)
```

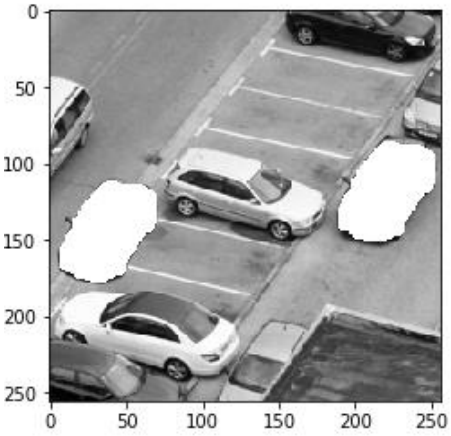
```
bloklar.shape
```
```
(256, 256, 8, 8)
```

```
dct_blogu.shape
```
```
(256, 256, 8, 8)
```

```
quantalama_tablosu = np.array([1, 1, 1, 2, 1, 2, 4, 2, 2, 4, 8, 4, 2, 4, 8, 16])
quantalama_tablosu = 16
```

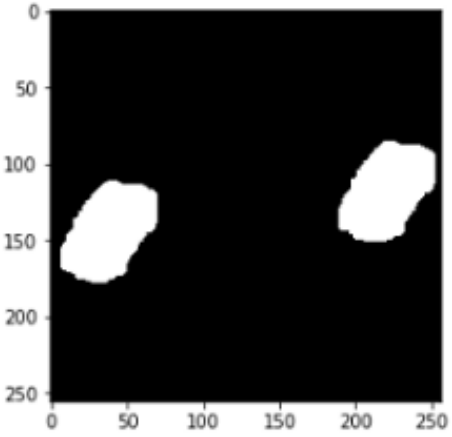
```
zigzag_vektorleri[i * genislik + j][:16] = np.floor(zig_zag(dct_blogu[i, j], blok_boyutu) / quantalama_tablosu)
zigzag_vektorleri[i * genislik + j][16] = i # X konum bilgisi
zigzag_vektorleri[i * genislik + j][17] = j # Y konum bilgisi
```

UYGULAMA ÜZERİNDEKİ TEST GÖRÜNTÜLERİ



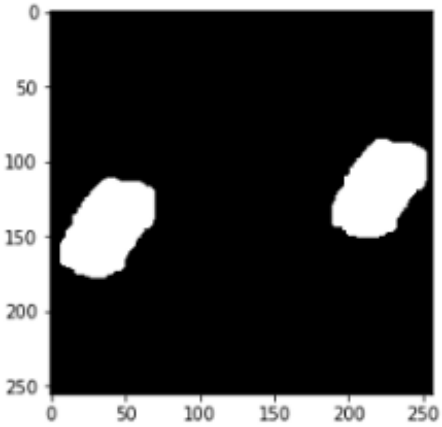
```
bakilacak_vektor = 25
benzerlik_esigi = 0.3
yakinlik_esigi = 5
```

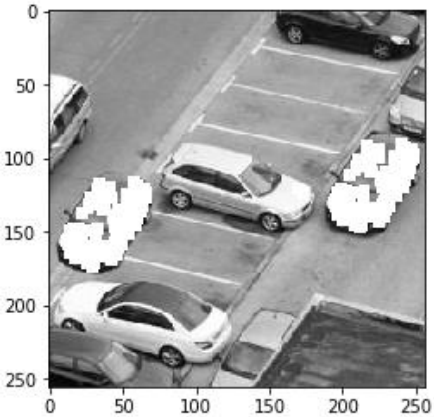
```
6738 degerde 5798 tanesi bulundu
Dogruluk = 0.8604927278124073
```



```
bakilacak_vektor = 25
benzerlik_esigi = 20
yakinlik_esigi = 175
```

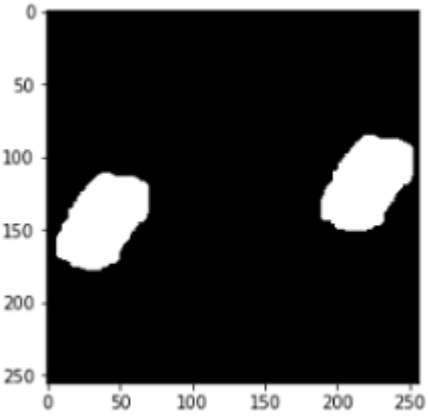
```
6738 degerde 5512 tanesi bulundu
Dogruluk = 0.8180468981893737
```

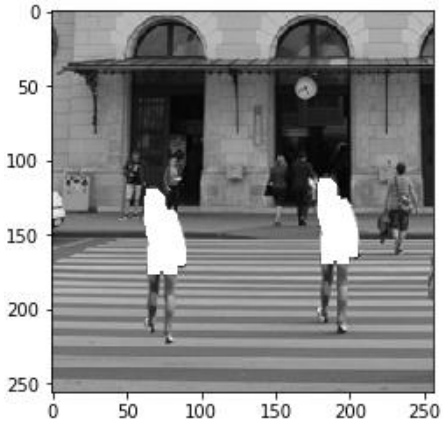




```
# DEGER SETLEMELERİ
bakilacak_vektor = 15
benzerlik_esigi = 20
yakinlik_esigi = 175
```

```
6738 degerde 4864 tanesi bulundu
Dogruluk = 0.721875927574948
```





```
# DEGER SETLEMELERİ
bakilacak_vektor = 10
benzerlik_esigi = 0.7
yakinlik_esigi = 5
```

```
4244 degerde 2346 tanesi bulund
Dogruluk = 0.5527803958529689
```