GÖRÜNTÜ İŞLEME DERSİ ÖDEV2

Öğrenci İsmi ve Numarası: İlker Bedir - 16011036

Ders Sorumlusu: Doç.Dr. M. Elif Karslıgil

Teslim Tarihi: 22.12.2020

Ödev Konusu: İçerik Tabanlı Görüntü Erişimi (Content Based Image Retrieval) Uygulaması : Bir resmin renk ve doku bilgisine göre benzerlerinin bulunması

1-)YÖNTEM BÖLÜMÜ:

Yöntem bölümünde önce resmi Python'nun CV kütüphanesinin hazır fonksiyonu kullandım.Bu fonksiyon ile resim renkli olarak bir 3 boyutlu olarak matriste rgb bilgileri saklanmaktadır.

```
def resim_okuma(path):  # resim dosyasını okuma fonksiyonu
    image = cv2.imread(path, cv2.IMREAD_COLOR)
    return image
```

Okunan resimleri eğitimleri histogramını hesaplamak için önce resimde kaç adet olduğunu hesaplıyorum daha sonra ise adetleri resmin toplam adetine bölüyorum(Normalizasyon için).

```
def rgb_hist(image):  # girilen resmi red,blue,green histogramlarına ayırma ve normalizasyon yapma fonksiyonu
    [rows, cols, bit] = image.shape
    size = rows*cols
    red = np.zeros(256)
    green = np.zeros(256)
    blue = np.zeros(256)
    for i in range(rows):
        for j in range(cols):
            red[image[i][j][0]] += 1
            green[image[i][j][1]] += 1
            blue[image[i][j][2]] += 1
    # resmin histogramı resmin boyutuna bölünerek (pixellerin bulunma olasılıkları eldesi) normalizasyon yapılıyor.
    for i in range(0, 256):
        red[i] = red[i] / size
        green[i] = green[i] / size
        blue[i] = blue[i] / size
    return red, blue, green
```

Daha sonra rgb resimleri hsv uzayında sadece Hue(ton) formül ile sınıfına çeviriyorum.

```
# red,blue,green listesinden hsv uzayın hue ya çevirme
def rgb_to_hsv(r, b, g):
    cmax = findmax(r, g, b)     # r, g, b arasında en bi
    cmin = findmin(r, g, b)     # r, g, b arasında en ki
    diff = cmax-cmin        # en büyük ile en küçüğü ara
    # fark 0 ise hue=0'dır
    if cmax == cmin:
        h = 0
    # cmax=r ise hue formulü
    elif cmax == r:
        h = (60 * ((g - b) / diff) + 360) % 360
    # cmax=g ise hue formulü
    elif cmax == g:
        h = (60 * ((b - r) / diff) + 120) % 360
    # cmax=b ise hue formulü
    elif cmax == b:
        h = (60 * ((r - g) / diff) + 240) % 360
    return h
```

Bu fonksiyonları önce 150 tane eğitim resmi için daha sonra ise her test resmi için kullanıyorum.

Test resmi için ise bu rgb resmi için rgb hesaplaması yapıyorum. Daha sonra ise önceden oluşturduğum eğitim listesinin her bir resmiyle olan uzaklığı buluyorum. Uzaklıkları ayrı bir listede tutuyorum.

```
    path0 = "test" + str(t) + "/"+str(i+1)+".jpg"
    image1 = resim_okuma(path0)
    red, blue, green = rgb_hist(image1)
    dtlist = np.zeros(150)
    for k in range(0, 150):
        # öklid distance yaptığım yer.Öncelikle test resmi için oluşan r,g,b histogramları
        for j in range(0, 256):
            dtlist[k] += math.sqrt(((red[j]-rgb_list[k, 0, j])**2)+(
                (blue[j]-rgb_list[k, 1, j])**2)+((green[j]-rgb_list[k, 2, j])**2))
```

Uzaklıkları tuttuğum listeyi ise bir BUBBLE sıralama fonksiyonunu çağırıyorum burada hem sayıları hem de başta tutulan sayı indislerini değiştiriyorum. Sıralama işlemi bittikten sonra indis dizisinin ilk 5 elemanını alıyorum.

```
min = bubblesort(dtlist)
```

```
# uzaklık listesini sıralama ve enküçük 5 indisi alma fonksiyonu bubble sort kullandım
def bubblesort(distance):
    indis = []
    for i in range(0, 150):
        indis.append(i)
    n = len(distance)
    for i in range(n):
        for j in range(0, n-i-1):
            if distance[j] >= distance[j+1]:
                distance[j], distance[j+1] = distance[j+1], distance[j]
                indis[j], indis[j+1] = indis[j+1], indis[j]
    return indis[:5]
```

Daha sonra aldığım indisleri hangi gruba ait olduğu kontrolü yapıyorum ve her bir test grubu için kaç tane doğru tahmin olduğunu döndürüyor ve yazdırıyor.

```
        for j in range(0, 5):
            if(int(min[j]/25)+1 == t):  # indis gruba uygun mu kontrolü yaptım
                acc.append(True)
            else:
                acc.append(False)
    newarr = array_split(acc, 5)
    p = []
    for i in range(0, 5):  # kaçtane true kaçtane yanlış yakınlık
        for j in range(0, 5):
            if(newarr[i][j] == True):
                c += 1
        p.append(c)
        c = 0
    return p
```

Hue_test fonksiyonunda benzer işlemleri kullanırken uzaklık almayı farklılık olarak sadece hue için yapıyorum ve ayrıca test resmin hue değerleri ile hue_listin hue değerleri yakınlık hesaplamasını yapıyorum.

```
def hue_test(hue_list, t):  # hue test fonksiyonum gidişat şekli rgb ye benzer ama öklid için sadece bir özel
    acc = []
    for i in range(0, 5):
        path0 = "test" + str(t) + "/"+str(i+1)+".jpg"
        image = resim_okuma(path0)
        red, blue, green = rgb_hist(image)
        h = np.zeros(256)
        distance = np.zeros(150)
        for j in range(0, 256):
            h[j] = rgb_to_hsv(red[j], blue[j], green[j])
        for j in range(0, 150):
            for k in range(0, 256):
                distance[j] = math.sqrt((h[k]-hue_list[j, k])**2)
        mint = bubblesort(distance)
        for j in range(0, 5):
            if(math.floor(mint[j]/25)+1 == t):
                acc.append(True)
            else:
                acc.append(False)
    newarr = array_split(acc, 5)
    p = []
    c = 0
    for i in range(0, 5):
        for j in range(0, 5):
            if(newarr[i][j] == True):
                c += 1
        p.append(c)
        c = 0
    return p
```

2)UYGULAMA

RGB TEST:

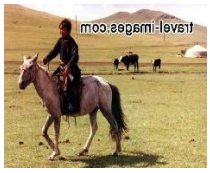
1.GRUP(DOG)

TEST RESMİ **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU** **DOĞRU**

TEST RESMİ **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU** **YANLIŞ**







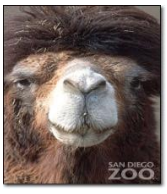
TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ DOĞRU






TEST RESMİ YANLIŞ DOĞRU YANLIŞ YANLIŞ YANLIŞ







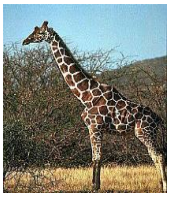
TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ DOĞRU

TEST BAŞARI ORANI 1.GRUP = %100

2.GRUP(DEVE)





TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ



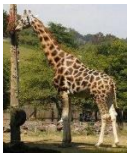



TEST RESMİ YANLIŞ DOĞRU YANLIŞ YANLIŞ YANLIŞ






TEST RESMİ DOĞRU DOĞRU YANLIŞ YANLIŞ YANLIŞ

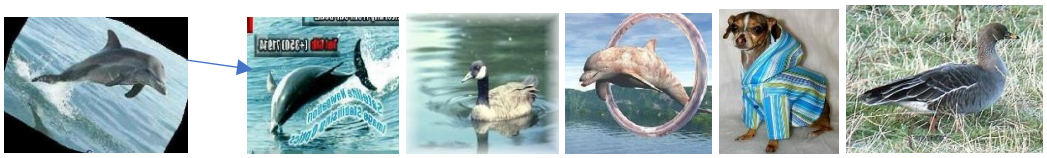



TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ DOĞRU

TEST RESMİ YANLIŞ YANLIŞ DOĞRU DOĞRU YANLIŞ

TEST BAŞARI ORANI 2.GRUP = %100

3.GRUP(YUNUS)



TEST RESMİ DOĞRU YANLIŞ DOĞRU YANLIŞ YANLIŞ

TEST RESMİ DOĞRU DOĞRU DOĞRU YANLIŞ DOĞRU

TEST RESMİ DOĞRU YANLIŞ YANLIŞ DOĞRU YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ DOĞRU YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST BAŞARI ORANI 3.GRUP = %80

4.GRUP(ZÜRAFA)

TEST RESMİ DOĞRU YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ DOĞRU YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ DOĞRU YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ DOĞRU DOĞRU YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ

TEST BAŞARI ORANI 4.GRUP = %100

5.GRUP(KAZ)

TEST RESMİ DOĞRU YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ DOĞRU YANLIŞ YANLIŞ YANLIŞ YANLIŞ

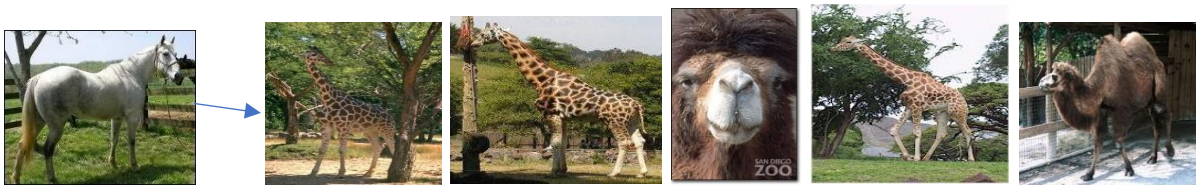
TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST BAŞARI ORANI 5.GRUP = %60

6.GRUP(AT)



TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ DOĞRU

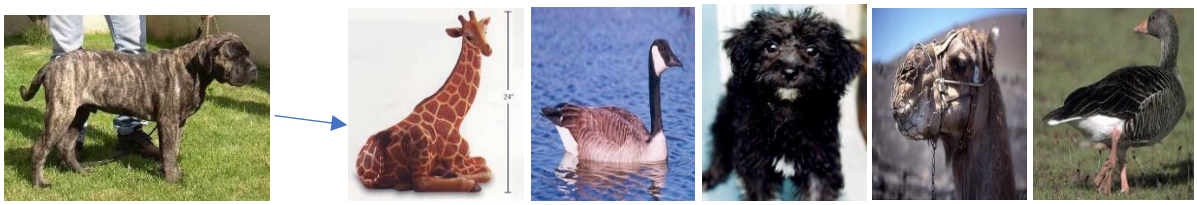
TEST RESMİ YANLIŞ DOĞRU YANLIŞ DOĞRU YANLIŞ

TEST RESMİ YANLIŞ DOĞRU YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST BAŞARI ORANI 6.GRUP = %60

HUE TEST:

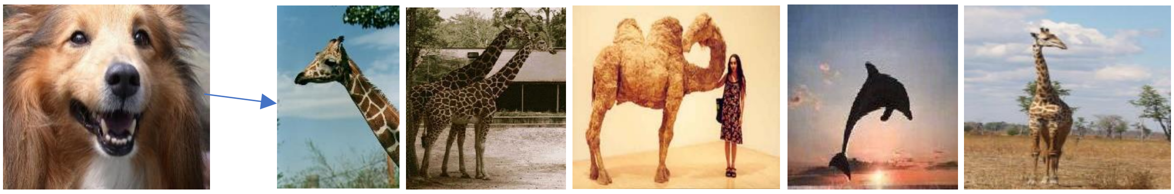


TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ DOĞRU DOĞRU YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ DOĞRU



TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ DOĞRU

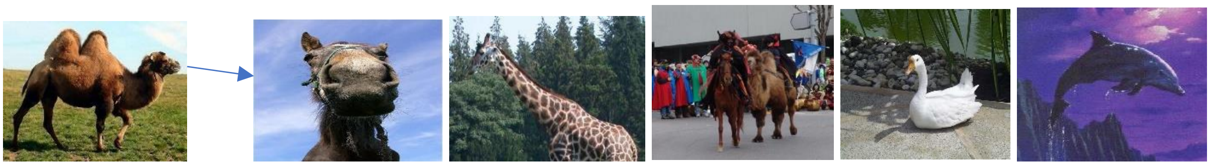
TEST BAŞARI ORANI 1.GRUP = %80

2.GRUP(DEVE)

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ DOĞRU YANLIŞ

TEST RESMİ DOĞRRU YANLIŞ YANLIŞ YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ



TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ

TEST RESMİ YANLIŞ YANLIŞ DOĞRU DOĞRU YANLIŞ

TEST BAŞARI ORANI 2.GRUP = %60

3.GRUP(YUNUS)






TEST RESMİ **YANLIŞ** **DOĞRU** **YANLIŞ** **YANLIŞ** **DOĞRU**






TEST RESMİ **DOĞRU** **YANLIŞ** **YANLIŞ** **YANLIŞ** **YANLIŞ**






TEST RESMİ **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU** **YANLIŞ**






TEST RESMİ **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU** **YANLIŞ**

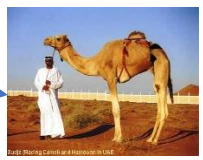


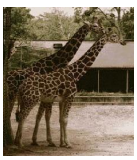



TEST RESMİ **DOĞRU** **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU**

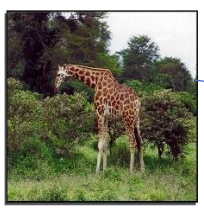
TEST BAŞARI ORANI 3.GRUP = %100

4.GRUP(ZÜRAFA)

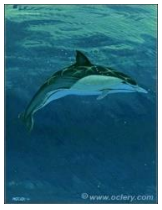




TEST RESMİ **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU** **YANLIŞ**




TEST RESMİ **YANLIŞ** **DOĞRU** **YANLIŞ** **YANLIŞ** **DOĞRU**

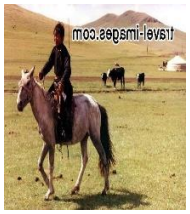





TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ



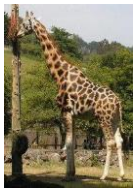
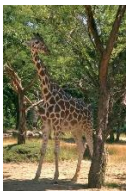


TEST RESMİ YANLIŞ YANLIŞ YANLIŞ DOĞRU YANLIŞ





TEST RESMİ YANLIŞ YANLIŞ YANLIŞ DOĞRU DOĞRU

TEST BAŞARI ORANI 4.GRUP = %100

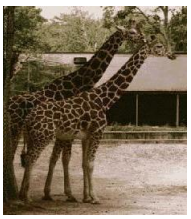
5.GRUP(KAZ)

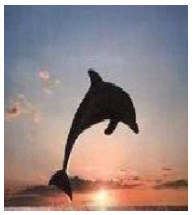





TEST RESMİ YANLIŞ YANLIŞ DOĞRU YANLIŞ YANLIŞ







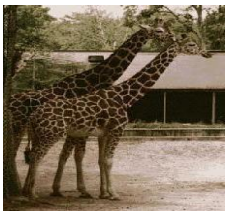
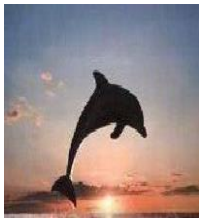
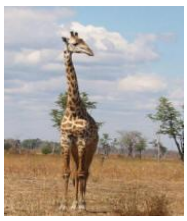

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ



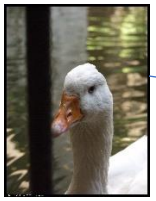



TEST RESMİ YANLIŞ DOĞRU YANLIŞ YANLIŞ YANLIŞ




TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ






TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

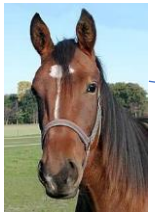
TEST BAŞARI ORANI 5.GRUP = %40

6.GRUP(AT)





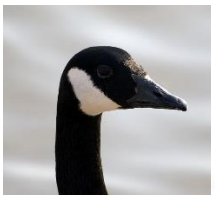

TEST RESMİ YANLIŞ YANLIŞ YANLIŞ DOĞRU YANLIŞ






TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ



TEST RESMİ YANLIŞ YANLIŞ YANLIŞ YANLIŞ YANLIŞ

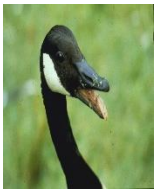





TEST RESMİ DOĞRU YANLIŞ YANLIŞ YANLIŞ YANLIŞ





TEST RESMİ **YANLIŞ** **YANLIŞ** **YANLIŞ** **DOĞRU** **YANLIŞ**

TEST BAŞARI ORANI 6.GRUP = %60

3-)SONUÇ BÖLÜMÜ:

RGB için tüm test başarı oranı %83,3 gibi bir orandır. Hue için bu oran yaklaşık olarak%73,3 genel başarı ortalaması ise %78,3 'dür. Bence RGB 'nin Hue değerine göre başarı oranı daha iyi olmasının temel sebebi RGB 3 değere göre yakınlık hesaplaması, Hsv uzayında sadece tek bir değere göre yapılması onu daha hassas yapar ve bu da daha iyi sonuçlar verir. Ancak HSV uzayında üç değer olsaydı HSV uzayı daha iyi sonuç verirdi.

Ayrıca eğitim yapılan resimler ve test yapılan resimlerin iyi seçilmiş olması doğruluk oranlarını artırır.