GÖRÜNTÜ İŞLEME

ÖDEV - 2

AYŞE HİLAL DOĞAN

17011907

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

a. YÖNTEM

```
def main():

    tests = []  # test imagelerinin oldugu liste
    images = []  # train imagelerinin oldugu liste
    rgb_histograms = []  # rgb histogramlarinin tutuldugu liste
    h_histograms = []   # hue histogramlarinin tutuldugu liste

    tests, images = take_random_images(tests, images)  # test ve train image'lerini random olarak sec

    # rgb ve h icin histogramlari hesapla ve ilgili listeye kaydet
    rgb_histograms, h_histograms = save_histograms(images, rgb_histograms, h_histograms)

    rgb(tests, images, rgb_histograms)   # RGB icin distance'i en az olan 5 resmi bul
    hue(tests, images, h_histograms)    # Hue icin distance'i en az olan 5 resmi bul
```

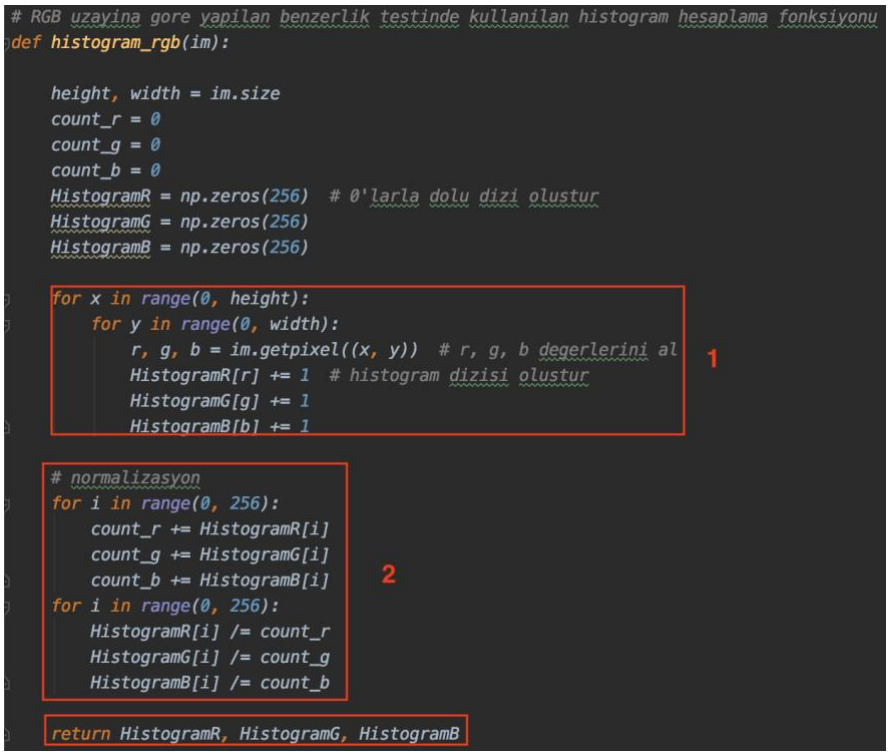
- **Main()** fonksiyonunda gerekli fonksiyonlar sırasıyla çağırılır.
  1- Dosyadan resimler random olarak alınır. Test için olan resimler **tests** listesine, train için olan resimler **images** listesine kaydedilir. (take_random_images())
  2- Her resim için rgb ve hue histogramlarının hesaplanması ve ilgili listeye kaydedilmesi yapılır. (save_histograms())
  3- RGB histogramlarına göre hesaplanmış, test resmine en yakın 5 resim bulunur. (rgb())
  4- Hue histogramlarına göre hesaplanmış, test resmine en yakın 5 resim bulunur. (hue())

```
def take_random_images(tests, images):
    # test ve train image'lerini her bir class'tan take_images() ile random secerek test veya images listesine kaydet
    # Her class'tan 5 test, 25 eğitim resmi al
    path = "/Users/hilaldogan/Desktop/GoruntuOdev2/028.camel"
    tests = take_images(tests, images, path, 5, 0)
    images = take_images(tests, images, path, 25, 1)
    path = "/Users/hilaldogan/Desktop/GoruntuOdev2/056.dog"
    tests = take_images(tests, images, path, 5, 0)
    images = take_images(tests, images, path, 25, 1)
    path = "/Users/hilaldogan/Desktop/GoruntuOdev2/057.dolphin"
    tests = take_images(tests, images, path, 5, 0)
    images = take_images(tests, images, path, 25, 1)
    path = "/Users/hilaldogan/Desktop/GoruntuOdev2/084.giraffe"
    tests = take_images(tests, images, path, 5, 0)
    images = take_images(tests, images, path, 25, 1)
    path = "/Users/hilaldogan/Desktop/GoruntuOdev2/089.goose"
    tests = take_images(tests, images, path, 5, 0)
    images = take_images(tests, images, path, 25, 1)
    path = "/Users/hilaldogan/Desktop/GoruntuOdev2/105.horse"
    tests = take_images(tests, images, path, 5, 0)
    images = take_images(tests, images, path, 25, 1)
    return tests, images
```

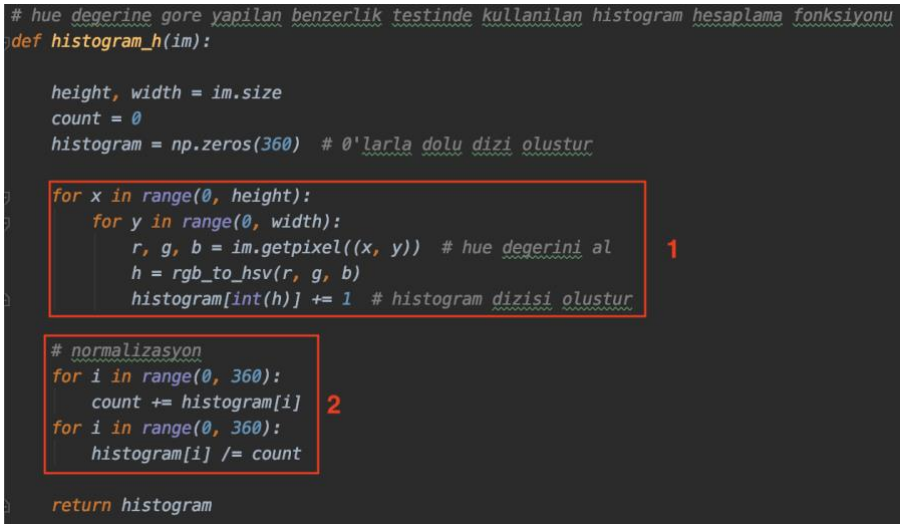
- **take_random_images()** fonksiyonu; parametre olarak tests ve images listelerini alır.
  Path'leri değiştirerek take_images() fonksiyonuyla toplamda 6 class'tan test için **random** 5 resim çekilerek 'tests' listesine, eğitim için 25 resim çekilerek 'images' listesine kaydedilir.
  Geriye dönüşte tests ve images listeleri return edilir.

```
# dosyadan random image alip, verilen flag degerine gore test veya train icin listeye kaydet
def take_images(tests, images, path, i, flag):
    count = 0
    while count < i:
        random_filename = random.choice([
            x for x in os.listdir(path)
            if os.path.isfile(os.path.join(path, x)) and x.endswith('.jpg')
        ])
        if flag == 0:
            if search(tests, random_filename):
                count += 1
                tests.append(random_filename)  # tests'e her classten 5 tane (6*5) test resmi kaydet
        if flag == 1:
            if search(tests, random_filename) and search(images, random_filename):
                count += 1
                images.append(random_filename)  # images e her siniftan 25 tane egitim resimlerini kayd
    if flag == 0:
        return tests
    else:
        return images
```

take_ images() fonksiyonu; parametre olarak tests, images, path, i, flag alır.
path'i verilen dosyadan, verilen i değerince random resim çeker. Flag=0 ise 'tests' listesi için seçim yapar. Flag=1 ise train resimlerini seçerek 'images' listesine kaydeder. Dönüşte hangi listeye kayıt yapılmışsa o liste return edilir.

- 
```
# RGB uzayina gore yapilan benzerlik testinde kullanilan histogram hesaplama fonksiyonu
def histogram_rgb(im):

    height, width = im.size
    count_r = 0
    count_g = 0
    count_b = 0
    HistogramR = np.zeros(256)  # 0'larla dolu dizi olustur
    HistogramG = np.zeros(256)
    HistogramB = np.zeros(256)

    for x in range(0, height):
        for y in range(0, width):
            r, g, b = im.getpixel((x, y))  # r, g, b degerlerini al
            HistogramR[r] += 1  # histogram dizisi olustur
            HistogramG[g] += 1
            HistogramB[b] += 1

    # normalizasyon
    for i in range(0, 256):
        count_r += HistogramR[i]
        count_g += HistogramG[i]
        count_b += HistogramB[i]
    for i in range(0, 256):
        HistogramR[i] /= count_r
        HistogramG[i] /= count_g
        HistogramB[i] /= count_b

    return HistogramR, HistogramG, HistogramB
```
1
2

histogram_rgb() fonksiyonu parametre olarak im (image) alır.
1. kısımda resmin R G B değerleriyle ayrı ayrı histogram oluşturulur.
2. kısımda oluşturulan histogramları normalize edilir.
Normalize edilen histogramlar (HistogramR, HistogramG, HistogramB) return edilir.

- 
```
# hue degerine gore yapilan benzerlik testinde kullanilan histogram hesaplama fonksiyonu
def histogram_h(im):

    height, width = im.size
    count = 0
    histogram = np.zeros(360)  # 0'larla dolu dizi olustur

    for x in range(0, height):
        for y in range(0, width):
            r, g, b = im.getpixel((x, y))  # hue degerini al
            h = rgb_to_hsv(r, g, b)
            histogram[int(h)] += 1  # histogram dizisi olustur

    # normalizasyon
    for i in range(0, 360):
        count += histogram[i]
    for i in range(0, 360):
        histogram[i] /= count

    return histogram
```
1
2

histogram_h() fonksiyonu parametre olarak im (image) alır.
1. kısımda resmin hue değeri **rgb_to_hsv()** fonksiyonuyla rgb değerinden elde edilir. Daha sonra bulunan hue değeri ile histogram oluşturulur.

2. kısımda oluşturulan histogramdaki her değer, toplam pixel sayısına bölünerek normalize edilir.
Normalize edilen histogram (histogram) return edilir.

```
def rgb_to_hsv(r, g, b):
    r, g, b = r/255.0, g/255.0, b/255.0
    mx = max(r, g, b)
    mn = min(r, g, b)
    df = mx-mn
    if mx == mn:
        h = 0
    elif mx == r:
        h = (60 * ((g-b)/df) + 360) % 360
    elif mx == g:
        h = (60 * ((b-r)/df) + 120) % 360
    elif mx == b:
        h = (60 * ((r-g)/df) + 240) % 360

    return h
```

- **rgb_to_hsv()** fonksiyonu parametre olarak r,g,b değerlerini alır. Bu değerlerden resmin hue değerini bulur ve h olarak return eder. Bu değer **histogram_h()** fonksiyonunda kullanılır.

```
# rgb ve h için histogramları hesapla ve ilgili listeye kaydet
def save_histograms(images, rgb_histograms, h_histograms):

    # 150 resim için r, g, b ,h histogramlarini diziye kaydet
    for i in range(0, 150):
        image2 = Image.open(images[i]).convert('RGB')   # images listesindeki train image'lerini ac
        histogramr2, histogramg2, histogramb2 = histogram_rgb(image2)  # rgb histogramlarini hesapla

        # hesaplanan histogramları rgb_histograms listesine sırayla kaydet
        rgb_histograms.append(histogramr2)
        rgb_histograms.append(histogramg2)
        rgb_histograms.append(histogramb2)

        histogramh = histogram_h(image2)  # hue için histogram hesapla
        # hesaplanan hue histogramini h_histograms listesine kaydet
        h_histograms.append(histogramh)

    return rgb_histograms, h_histograms
```

- **save_histograms()** fonksiyonu; parametre olarak images, rgb_histograms, h_histograms alır. Histogramların program boyunca **bir defa** hesaplanması için histogramların kaydedilmesine yarar.
rgb_histograms: resmin rgb histogramlarının saklandığı listedir.
h_histograms: resmin hue histogramlarının saklandığı listedir.
For döngüsü içinde train image'leri açılır. Rgb ve hue histogramları hesaplanır ve listelerde saklanır.
histogram_rgb() fonksiyonuyla rgb histogramı hesaplanır, histogram_h() fonksiyonuyla hue histogramı hesaplanır.
Dönüşte rgb_histograms ve h_histograms olarak kaydedilen histogramlar döndürülür.

```python
def rgb(tests, images, rgb_histograms):
    print("-------------------------------")
    print("-------------RGB---------------")
    print("-------------------------------")
    # 25 test resmiyle r, g, b ye gore benzerlik testi yap
    for count in range(0, 30):
        distances = []  #
        image1 = Image.open(tests[count]).convert('RGB')
        print([count+1], ".Test Resmi: ", tests[count])
        histogramr1, histogramg1, histogramb1 = histogram_rgb(image1)

        # daha onceden kaydedilen 150 tane resmin rgb histogramlariyla islem yap
        j = 0
        while j < 450:
            histogramr2 = rgb_histograms[j]
            j += 1
            histogramg2 = rgb_histograms[j]
            j += 1
            histogramb2 = rgb_histograms[j]
            distance = calc_distance_rgb(histogramr1, histogramr2, histogramg1, histogramg2, histogramb1, histogramb2)
            distances.append(distance)  # distance'ları diziye kaydet
            j += 1

        # en yakin 5 resmi bul
        find_similar_images(distances, images)
```

- **rgb()** fonksiyonu; parametre olarak tests, images, rgb_histograms alır. 30 test resmiyle, 150 train testini karşılaştırarak 'distance' hesaplar. **(calc_distance_rgb())** En küçük 5 distance'ı bulabilmek için bütün uzaklıkları 'distances' listesinde toplar.
  Burada histogramları önceden hesaplanılan 'rgb_histograms[]' listesinden alır.
  Bütün test resimleriyle train resimlerini karşılaştırdıktan sonra **find_similar_images()** fonksiyonu ile distance'ı en küçük 5 resmi bulup ekrana yazdırır.

```python
# rgb'de  histogramlar arasinda distance hesapla
def calc_distance_rgb(histogramr1, histogramr2, histogramg1, histogramg2, histogramb1, histogramb2):

    distance = 0
    for x in range(0, 256):
        rd = histogramr1[x] - histogramr2[x]
        gd = histogramg1[x] - histogramg2[x]
        bd = histogramb1[x] - histogramb2[x]
        distance += math.sqrt(rd * rd + gd * gd + bd * bd)

    return distance
```

- **calc_distance_rgb()** fonksiyonu parametre olarak histogramr1, histogramr2, histogramg1, histogramg2, histogramb1, histogramb2 alır.
  Euclidian Distance ile RGB değerleri için iki resmin histogramları arasındaki benzerliği ölçer ve sonucu 'distance' değerinde return eder.

```python
def hue(tests, images, h_histograms):
    print("-------------------------------")
    print("-------------HUE---------------")
    print("-------------------------------")
    # HUE
    # 25 test resmiyle hue ya gore benzerlik testi yap
    for count in range(0, 30):
        distances = []
        image1 = Image.open(tests[count]).convert('RGB')
        print([count+1], ". Test Resmi: ", tests[count])
        histogramh1 = histogram_h(image1)

        j = 0
        while j < 150:
            histogramh2 = h_histograms[j]
            distance = calc_distance_h(histogramh1, histogramh2)
            distances.append(distance)  # distance'ları diziye kaydet
            j += 1

        # en yakin 5 resmi bul
        find_similar_images(distances, images)
```

- **hue()** fonksiyonu; parametre olarak tests, images, h_histograms alır.
  rgb() ile aynı adımları hue ile hesaplanmış histogramlar için uygular. Test histogramları burada ilk defa hesaplanır [histogram1 = histogram_h(image1) , **histogram_h()** hue değeri için histogram hesaplaması yapar] ve daha önceden hesaplanmış histogramlarla arasındaki

mesafe ölçülür (**calc_distance_h()**). Ölçülen distance'lar distances[] listesine kaydedilir ve **find_similar_images()** ile en yakın 5 tanesi ekrana yazdırılır.

```
# hue degeri icin  histogramlar arasinda distance hesapla
def calc_distance_h(histogramh1, histogramh2):
    distance = 0
    for x in range(0, 360):
        d = histogramh1[x] - histogramh2[x]
        distance += math.sqrt(d * d)

    return distance
```

- **calc_distance_h ()** fonksiyonu parametre olarak histogramh1, histogramh2 alır. Euclidian Distance ile iki resmin hue histogramları arasındaki mesafeyi ölçer ve sonucu 'distance' değerinde return eder.

```
# distances 'da tutulan distance'lar arasindaki en kucuk 5 tanesini bul
def find_similar_images(distances, images):
    for x in range(0, 5):
        i = distances.index(min(distances))
        image = Image.open(images[i])
        print(images[i])
        distances.remove(min(distances))
    distances.clear()
```

- **find_similar_images()** fonksiyonu; parametre olarak distances, images alır. distances[] listesindeki en küçük 5 distance'ı bulur ve o distance'a ait resmi ekrana yazdırır.

```
# Verilen listede verilen image isminin olup olmadigina bak
def search(list, name):
    for i in range(len(list)):
        if list[i] == name:
            return False
    return True
```

- **Search()** fonksiyonu verilen resim listesinde verilen resmin olup olmadığını kontrol etmek için kullanılır. Dosyadan random resim seçilirken aynı resmin birden fazla gelmemesi amacıyla kontrol için kullanılan fonksiyondur.

Böylece rgb değerlerine göre ve hue değerine göre 6 classtan alınmış 25 test resmiyle 150 train resmi karşılaştırılarak benzerlikleri ölçülür ve her test resmi için rgb ve hue değerlerine göre en benzer 5 resim bulunur.

Örnek ekran çıktısı:

```
------------------------------
------------RGB---------------
------------------------------
[1] .Test Resmi:  028_0047.jpg
089_0017.jpg
105_0073.jpg
028_0052.jpg
056_0004.jpg
056_0076.jpg
[2] .Test Resmi:  028_0091.jpg
105_0029.jpg
089_0017.jpg
056_0005.jpg
057_0014.jpg
056_0040.jpg
```

```
------------------------------
------------HUE---------------
------------------------------
[1] . Test Resmi:  028_0047.jpg
089_0070.jpg
089_0109.jpg
028_0026.jpg
089_0059.jpg
056_0059.jpg
[2] . Test Resmi:  028_0091.jpg
056_0040.jpg
105_0197.jpg
056_0045.jpg
057_0052.jpg
089_0038.jpg
```

b. UYGULAMA

RGB uzayı için başarılı örnekler:

1.
```
[10] .Test Resmi:  056_0015.jpg
056_0053.jpg
056_0099.jpg
105_0181.jpg
105_0252.jpg
089_0054.jpg
```

Test resmi:

2.
```
[12] .Test Resmi:  057_0101.jpg
057_0033.jpg
028_0059.jpg
057_0049.jpg
089_0079.jpg
105_0252.jpg
```

Test resmi:

RGB uzayı için başarısız örnekler:

1.
```
[3] .Test Resmi:  028_0018.jpg
089_0095.jpg
105_0042.jpg
089_0090.jpg
089_0089.jpg
105_0230.jpg
```

Test resmi:

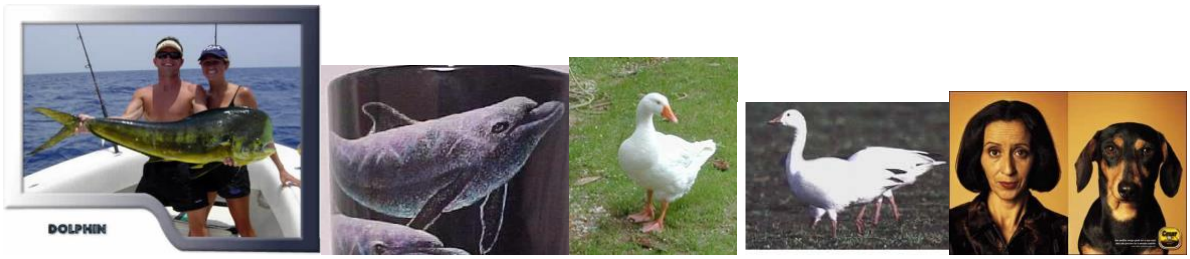
2.
```
[23] .Test Resmi:  089_0019.jpg
056_0053.jpg
056_0012.jpg
105_0039.jpg
057_0002.jpg
028_0038.jpg
```

Test resmi:

HSV uzayı için başarılı örnekler:

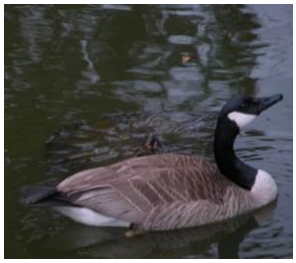
1.
```
[15] . Test Resmi:  057_0007.jpg
057_0039.jpg
057_0086.jpg
089_0032.jpg
089_0055.jpg
056_0061.jpg
```

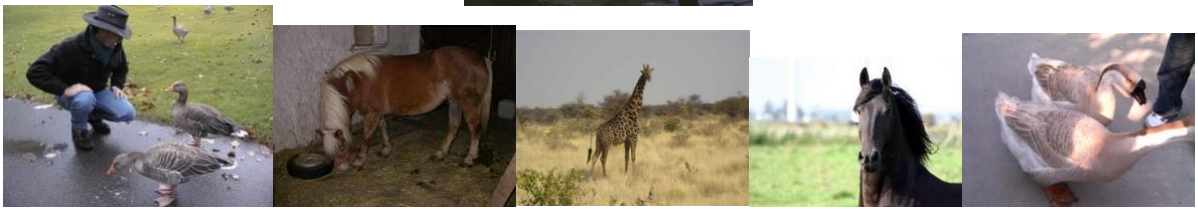
Test resmi:



2.
```
[24] . Test Resmi:  089_0022.jpg
089_0074.jpg
105_0223.jpg
084_0074.jpg
105_0140.jpg
089_0040.jpg
```

Test resmi:





HSV uzayı için başarısız örnekler:

1.
```
[25] . Test Resmi:  089_0071.jpg
084_0069.jpg
105_0141.jpg
057_0002.jpg
105_0011.jpg
105_0270.jpg
```

Test resmi:

2.
```
[2] . Test Resmi:  028_0073.jpg
056_0006.jpg
105_0151.jpg
105_0172.jpg
089_0024.jpg
056_0098.jpg
```

Test resmi:

- **Tanıma başarıları**

RGB
028.camel : %80
056.dog : %80
057.dolphin : %60
084.giraffe : %60
089.goose : %80
105.horse : %40
Toplam başarı yüzdesi: %66

HSV
028.camel : %80
056.dog : %80
057.dolphin : %60
084.giraffe : %60
089.goose : %100
105.horse : %80
Toplam başarı yüzdesi: %76

Yukarıda verilen tanıma başarıları programın bir kez çalıştırılması sonucu elde edilen sonuçlardır.
Toplam başarı yüzdesi hesaplanırken,6 sınıfın başarı ortalaması alınarak sonuç elde edilmiştir.

c. SONUÇ

RGB ve HSV uzaylarına göre yapılan benzerlik testlerinde, random seçilen resimlere göre tanıma başarıları b.Uygulama bölümünde verilmiştir.

RGB uzayındaki toplam başarı yüzdesi ortalama %66 çıkmıştır. Bu sonuçlar random seçilen resimlere göre farklılık göstermektedir. Seçilen resmin arka planının karışık olması ve red, green, blue oranlarına ve random seçilen eğitim resimlerine göre her denemede sonuçlar değişebilmektedir. Basit bir yöntem olan RGB uzayına göre benzer resim bulma çok olmasa da genel anlamda başarılı sonuçlar vermektedir.
Programın 5 ayrı çalışmasında 028 grubu için sonuçlar:
028.camel (RGB): %100 - %60 - %60 - %60 - %80

HSV uzayındaki hue değerine göre yapılan benzerlik testinde toplam başarı yüzdesi ortalama %76 çıkmıştır. Seçilen resmin arka plan karışıklığına göre, renk tonuna (hue değerine) göre ve random seçilen eğitim resimlerine göre her program çalıştırılmasında sonuçlar değişebilmektedir. Basit bir yöntem olan HSV uzayına göre benzer resim bulma çok olmasa da genel anlamda başarılı sonuçlar vermektedir.
Programın 5 ayrı çalışmasında 028 grubu için sonuçlar:
028.camel (HSV): %60 - %100 - %100 - %40 - %80

Genel değerlendirme: RGB ve HSV uzaylarına göre yapılan benzerlik testlerinde RGB'de ortalama %66 başarı, HSV'de ortalama %76 başarı elde edilmiştir. Başarı oranları birbirine yakındır, HSV'de %10 daha başarılı çıkmıştır. Fakat bu sonuç her çalıştırmada farklılık göstermektedir. Yapılan 5 ayrı çalıştırmanın sonuçlarının ortalamasında HSV uzayında yapılan testlerin ortalaması %5 daha başarılı çıkmıştır.
Fakat test ve eğitim resimleri her seferinde random seçildiği için bu başarı oranı kimi zaman hsv'de kimi zaman rgb'de yüksek çıkmaktadır. Yukarıda verilen örnekte 028.camel için hsv'de başarı oranı 2 testte rgb'den daha yüksek, 2 testte daha düşük, 1 testte eşittir.
b.Uygulama kısmında yapılan 5 testte her sınıf için başarı aralıkları verilmiştir.