GÖRÜNTÜ İŞLEME ÖDEV 2

HAZIRLAYAN: YUNUS KARATEPE 17011051

KONU : İçerik Tabanlı Görüntü Erişimi

Yöntem:

Genel mantık olarak rgb ve lbp histogramları çıkarılmış, normalize edilmiş ve bu histogramlar üzerinden distance değerleri hesaplanarak minimum distance değerine sahip olan resimlerin birbirlerine en çok benzediği kabul edilmiştir.

Kullanılan Fonksiyonlar:

1-) def image_infos(file_name):

Bu fonksiyon ile dosya uzantısı alınmış resim üzerinde yapılan işlemler sonucunda geri dönüş değeri olarak normalize edilmiş bir biçimde (0-1 arasında, pixellerin bulunma ihtimallerini verecek şekilde)

[blue, green, red, lbp] veri yapısı döndürülmüştür. 0. indiste mavi, 1. indiste yeşil, 2. indiste kırmızı ve son olarak 3. indiste lbp (doku) histogramlarını içerir.

2-) def create_data(num_of_images):

Bu fonksiyon sayesinde “training” klasörü içinde bulunan 70 adet resim için ayrı ayrı image_infos fonksiyonu çağrılarak hepsi için 4 adet histogram elde edilmiş ve bunlar bir “data.txt” dosyasında saklanmıştır. Dosyada tutulma biçimi olarak ilk 1024 satırlık veri 1 resim dosyasına ait histogramların bilgisini içerir.

3-) def find_similar(img_name):

Bu fonksiyon adından da anlaşılabileceği üzere gelen test resmi için rgb histogramları açısından en yakın 5 resmi ve lbp (doku) histogamı için 5 resmi göstermektedir.

Uygulama:

```
find_similar("test/1.jpg")
find_similar("test/13.jpg")
find_similar("test/22.jpg")
find_similar("test/35.jpg")
find_similar("test/42.jpg")
find_similar("test/51.jpg")
find_similar("test/66.jpg")
```

input olarak verilen 1-13-22-35-42-51-66 dosyaları için (her 10 luk küme aynı türden resimleri içermektedir) yapılan işlemler sonucu elde edilenler şu şekildedir:

Resim 1 için : (Akordeon)

```
Resme en yakinlar :
rgb ye gore
['1.jpg', '9.jpg', '6.jpg', '48.jpg', '10.jpg']
lbp ye gore
['6.jpg', '2.jpg', '5.jpg', '8.jpg', '10.jpg']
```

Burada görüldüğü üzere;

Rgb doğruluk oranı 4/5, lbp doğruluk oranı 5/5 tir.

Resim 2 için : (Dalmaçyalı)

```
Resme en yakinlar :
rgb ye gore
['66.jpg', '67.jpg', '3.jpg', '37.jpg', '19.jpg']
lbp ye gore
['30.jpg', '12.jpg', '43.jpg', '13.jpg', '28.jpg']
```

rgb doğruluk oranı 1/5,

lbp doğruluk oranı 2/5

Resim 3 için : (Çiçek)

```
Resme en yakinlar :
rgb ye gore
['13.jpg', '42.jpg', '17.jpg', '44.jpg', '23.jpg']
lbp ye gore
['21.jpg', '69.jpg', '43.jpg', '65.jpg', '26.jpg']
```

rgb doğruluk oranı 1/5

lbp doğruluk oranı 2/5

Resim 4 için : (Yunus)

```
Resme en yakinlar :
rgb ye gore
['32.jpg', '62.jpg', '20.jpg', '70.jpg', '36.jpg']
lbp ye gore
['18.jpg', '50.jpg', '42.jpg', '12.jpg', '30.jpg']
```

rgb için doğruluk oranı 2/5,

lbp için doğruluk oranı 1/5

Resim 5 için : (Vahşi Kedi)

```
Resme en yakinlar :
rgb ye gore
['18.jpg', '46.jpg', '42.jpg', '3.jpg', '9.jpg']
lbp ye gore
['42.jpg', '43.jpg', '26.jpg', '65.jpg', '25.jpg']
```

rgb için doğruluk oranı 2/5

lbp için doğruluk oranı 2/5

Resim 6 için: (Çita)

```
Resme en yakinlar :
rgb ye gore
['56.jpg', '57.jpg', '58.jpg', '51.jpg', '60.jpg']
lbp ye gore
['56.jpg', '59.jpg', '67.jpg', '24.jpg', '57.jpg']
```

rgb için doğruluk oranı 5/5

lbp için doğruluk oranı 3/5

Resim 7 için : (Yelkenli)

```
Resme en yakinlar :
rgb ye gore
['37.jpg', '25.jpg', '33.jpg', '50.jpg', '49.jpg']
lbp ye gore
['69.jpg', '30.jpg', '68.jpg', '64.jpg', '28.jpg']
```

rgb için doğruluk oranı 0/5

lbp için doğruluk oranı 3/5

Tüm değerleri hesaba katarsak;

rgb için toplam 15/35 = 0.42857142857

lbp için toplam 18/35 = 0.51428571429

değerlerini elde ettik.

Sonuç:

Tamamen doğru istatistiksel veri alabilmemiz için çok fazla sayıda test resmi için denememiz gerekmektedir. Şu an denediğimiz 35 er adet resim için lbp matris (doku) özelliklerini kullanmak daha mantıklı gözüküyor olabilir.

Bu işlemi istatistiksel olarak analiz eden kodu şu şekilde açıklayabiliriz: Tüm 70 test resmi için aynı işlemleri tekrar edip her resim için fonksiyondan başarı oranını veren bir dizi döndüren (dizi[0] => rgb başarı oranı, dizi[1] lbp başarı oranı) kodu her işlemi yaptığımız find_similar fonksiyonuna ekliyoruz.

Daha sonrasında tüm 70 resim için gelen 0-1 arası başarı oranlarını topladıktan sonra bu sonucu 70 e bölmemiz gerekiyor. Bu aşama ise şu şekilde

```
successRateRgb = 0
successRateLbp = 0
a = [0.0] * 2
for i in range(70):
    a = find_similar(i+1)
    successRateRgb += a[0]
    successRateLbp += a[1]

successRateRgb /= 70
successRateLbp /= 70
print("Rgb Başarı Oranı = " + str(successRateRgb))
print("Lbp Başarı Oranı = " + str(successRateLbp))
```

Tüm bu işlemler sonucunda aldığımız başarı oranı ise bizi yine lbpnin daha doğru bir seçim olacağına yönlendiriyor.

```
Rgb Başarı Oranı = 0.3857142857142856
Lbp Başarı Oranı = 0.40285714285714427
```