GÖRÜNTÜLERDE RENGE GÖRE NESNE TESPİTİ VE KUTU İÇİNE ALINMASI

Bir görüntüde renkleri tespit etmek, görüntü işlemedeki önemli bir konudur. Bu işlem, görüntülerdeki nesneleri tanıma, konumlandırma ve sınıflandırma gibi işlemleri kolaylaştırır.

Renk, bir görüntünün en belirgin özelliklerinden biridir. Renk tespiti, görüntülerdeki nesneleri tanımak için yaygın olarak kullanılır. Örneğin, yüz tanıma sistemleri, yüzlerdeki renk özelliklerini kullanarak yüzleri tanır. Malzeme tanıma sistemleri, malzemelerdeki renk özelliklerini kullanarak malzemeleri tanımlar.

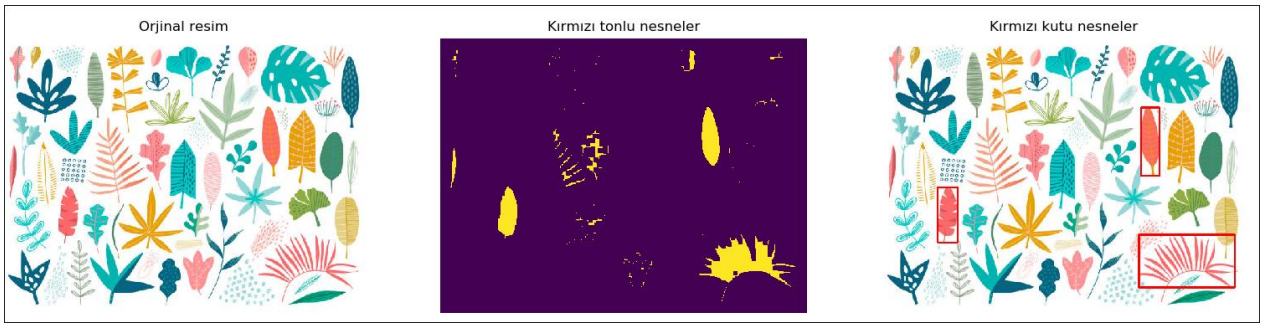
Renk tespiti, endüstriyel süreçlerin otomasyonunu artırabilir, kalite kontrolü sağlayabilir ve ürünlerin estetik ve teknik özelliklerini optimize edebilir. Bu nedenle, bu tür teknikler, birçok endüstriyel sektörde yaygın olarak kullanılmaktadır.

- Otomatik üretim: Ürünlerin kalite kontrolünde, renk ve doku özelliklerinin analiz edilmesi kullanılır. Örneğin, bir otomobil üretim hattında, boyanmış otomobillerin renkleri ve dokuları kontrol edilir.
- Malzeme tanıma: Malzemelerin özelliklerinin belirlenmesinde, renk ve doku özelliklerinin analizi kullanılır. Örneğin, bir madencilik tesisinde, kayaların özellikleri, renk ve doku özelliklerinin analiz edilmesiyle belirlenir.
- Tıp: Hastalıkların teşhisinde, renk ve doku özelliklerinin analizi kullanılır. Örneğin, cilt kanseri tespitinde, ciltteki renk ve doku özellikleri analiz edilir.

BGR görüntülerde ana renkler ve ara renkler farklı yöntemlerle tespit edilebilir. Aşağıda kırmızı renkli nesneleri tespit edip kapsayıcı kutuya alan bir program verilmiştir.

```
import cv2
from matplotlib import pyplot as plt

x = cv2.imread('desenler.jpg')
y=x.copy()

xGri=cv2.cvtColor(x,cv2.COLOR_BGR2GRAY)
xred=x[:,:,2]
kirmiziRenk=cv2.subtract(xred,xGri)
z,xIkili=cv2.threshold(kirmiziRenk,80,255,cv2.THRESH_BINARY)

strel=cv2.getStructuringElement(cv2.MORPH_RECT, (7,7))
# Birbirine yakın ancak ayrık kalan nesneler birleştiriliyor.
x_kirmizi=cv2.morphologyEx(xIkili, cv2.MORPH_CLOSE, strel)

(toplamBlob, etiket_id, degerler, centroid)=cv2.connectedComponentsWithStats(x_kirmizi, 4, cv2.CV_32S)
```

```
# Çok fazla kırmızı tonlu nesne var. Alanı 1000 pikselden büyük olanları seçelim.
for i in range (1, toplamBlob):
    if degerler[i, cv2.CC_STAT_AREA]>1000:

        # Her bir blobun sol üst köşe, en ve boy koordinatları alınıyor.
        x1 = degerler[i, cv2.CC_STAT_LEFT]
        y1 = degerler[i, cv2.CC_STAT_TOP]
        w = degerler[i, cv2.CC_STAT_WIDTH]
        h = degerler[i, cv2.CC_STAT_HEIGHT]

        # Kapsayıcı kutunun (Bounding box) koordinatları hesaplanıyor.
        solUst = (x1, y1)
        sagAlt = (x1+ w, y1+ h)

        # Herbir blob'a kutu çiziliyor.
        cv2.rectangle(y, solUst, sagAlt, (0, 0, 255), 2)

plt.subplot(131)
plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB)), plt.title('Orjinal resim'),plt.axis('off')
plt.subplot(132)
plt.imshow(cv2.cvtColor(x_kirmizi, cv2.COLOR_BGR2RGB)), plt.title('Kırmızı tonlu nesneler'),plt.axis('off')
plt.subplot(133)
plt.imshow(cv2.cvtColor(y, cv2.COLOR_BGR2RGB)), plt.title('Kırmızı kutu nesneler'),plt.axis('off')
```



GÖRÜNTÜLERDE NESNE KONTUR'LARININ BULUNMASI

Konturlar nesne dış sınırlarıdır. Yani basitçe, aynı renk veya yoğunluğa sahip tüm sürekli noktaları (sınır boyunca) birleştiren bir eğri olarak açıklanabilir. Konturlar, şekil analizi ve nesne algılama ve tanıma için kullanışlı bir araçtır. Daha iyi doğruluk için ikili görüntüler kullanılabilir. Bu nedenle bir görüntüdeki nesne konturlarını bulmadan önce, eşik veya canny kenar algılama kullanmak uygun olabilir.

OpenCV'de konturları bulmak, siyah arka plandan beyaz nesneyi bulmak gibidir. Bu nedenle, bulunacak nesnenin beyaz ve arka planın siyah olması gerektiği unutulmamalıdır.

Örnek:

```
import numpy as np
import cv2 as cv
im = cv.imread('test.jpg')
imgray = cv.cvtColor(im, cv.COLOR_BGR2GRAY)
ret, thresh = cv.threshold(imgray, 127, 255, 0)
contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

cv.findContours() fonksiyonunda üç argüman vardır,

- birincisi kaynak görüntü,
- ikincisi kontur alma modu,
- üçüncüsü kontur yaklaştırma yöntemidir.

Ayrıca, **konturları** ve **hiyerarşiyi** çıktı olarak verir. Konturlar, görüntüdeki tüm konturların bir Python listesidir. Her bir kontur, nesnenin sınır noktalarının (x,y) koordinatlarından oluşan bir dizi/matris'dir.

print(contours[0]) komutu ile içeriğini görebiliriz. CHAIN_APPROX_NONE kesinlikle tüm kontur noktalarını saklar. CHAIN_APPROX_SIMPLE olarak ayarlanırsa yatay, dikey ve diyagonal segmentleri sıkıştırır ve yalnızca uç noktalarını bırakır.

```
import cv2
img = cv2.imread('blobs1.jpg')
cv2.imshow("orjinal", img)
xred=img[:,:,2]
# Gri seviye dönüşümü
gray_img = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)

kirmiziRenk=cv2.subtract(xred,gray_img)
cv2.imshow("gri", kirmiziRenk)
z,xIkili=cv2.threshold(kirmiziRenk,70,255,cv2.THRESH_BINARY)
cv2.imshow("Binary", xIkili)

contours, hierarchy = cv2.findContours(xIkili,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
for i in range(0, len(contours)):
    img = cv2.drawContours(img, contours[i], -1, (0,255,0), 3)
cv2.imshow("Frame", img)
# cv2.waitKey()
# cv2.destroyAllWindows()
```

Yukarıdaki örnekte, imgede bulunan kırmızı renkli tüm nesnelerin konturunu döngü içinde çizen bir program bulunmaktadır.

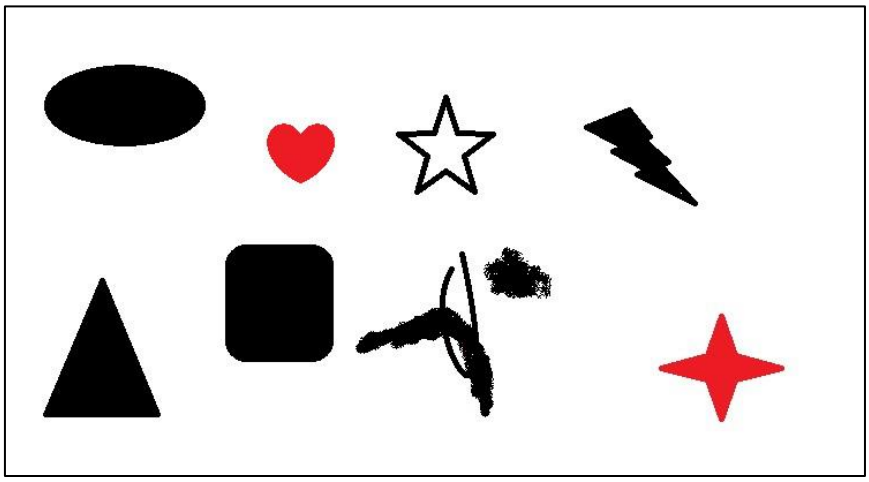
Tüm konturlar döngü içinde çiziliyor. Her konturun bir indisi var. "i" indis numarasını temsil ediyor.

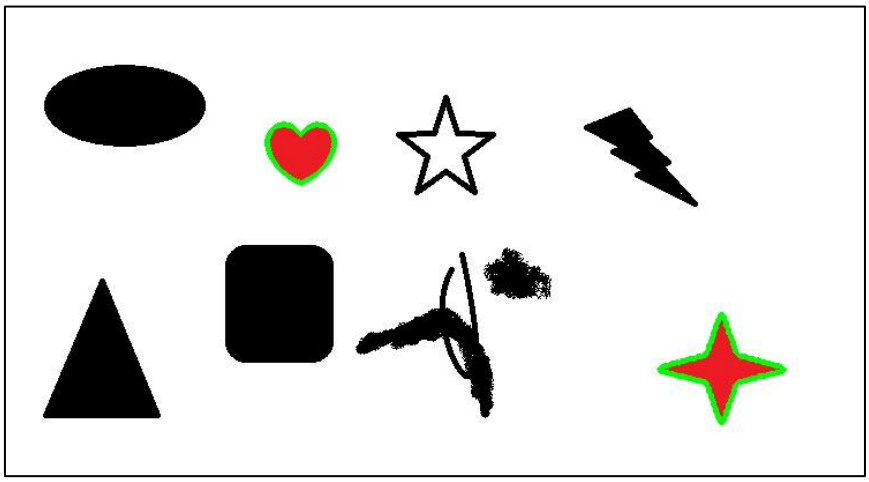
```
for i in range(0, len(contours)):
    img = cv2.drawContours(img, contours[i], -1, (0,255,0), 3)
```

Kontur kalınlığı "3" olacak

Konturun tamamını çizeceğimizi gösterir.

Kontur yeşil renkte olacak.





Görüntülerdeki Dairesel nesnelerin tespiti ve kutu içine alınması:

Dairesel nesnelerin tespiti için BLOB'lara bir dairesellik formülü uygulanır.

$$Dairesellik = \frac{4 * \pi * Alan}{Çevre^2}$$

Bu formüle göre sonuç 0.7'den büyükse bu nesneye daireseldir denilebilmektedir.

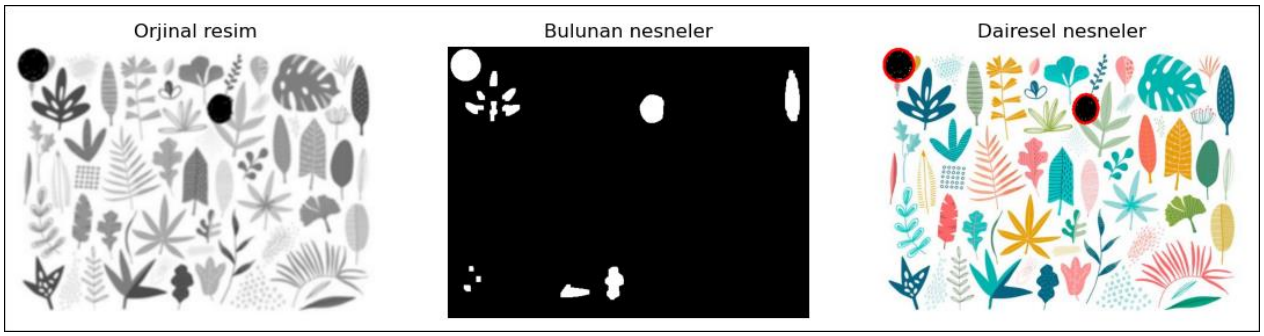
Buna göre aşağıdaki programı inceleyiniz:

```
import cv2
from matplotlib import pyplot as plt
import numpy as np

x = cv2.imread('desenler-2.jpg')
y=x.copy()
x=cv2.cvtColor(x, cv2.COLOR_BGR2GRAY)
x=cv2.blur(x, (5,5))

z,xIkili=cv2.threshold(x,90,255,cv2.THRESH_BINARY_INV)

strel=cv2.getStructuringElement(cv2.MORPH_RECT, (7,7))
# Birbirine yakın ancak ayrık kalan nesneler birleştiriliyor.
xFiltreli=cv2.morphologyEx(xIkili, cv2.MORPH_CLOSE, strel)

# Gürültüler temizleniyor..
xFiltreli=cv2.morphologyEx(xFiltreli, cv2.MORPH_OPEN, strel)
# Görüntüdeki tüm nesnelerin konturları bulunuyor.
contours, hierarchy = cv2.findContours(xFiltreli, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
# Tüm konturlar için tek tek işlem yapılıyor..
for contour in contours:
    alan = cv2.contourArea(contour)
    cevre = cv2.arcLength(contour, True)
    dairesellik = 4 * np.pi * alan / (cevre ** 2)
    if (alan>500) and (0.7 < dairesellik < 1):
        # Kapsayıcı elipsler bulunuyor ve çizdiriliyor.
        ellipse = cv2.fitEllipse(contour)
        cv2.ellipse(y, ellipse, (0, 255, 0), 4)

plt.subplot(131)
plt.imshow(x, cmap='gray'), plt.title('Orjinal resim'),plt.axis('off')
plt.subplot(132)
```

#Kapsayıcı kutu çizdirmek istersek aşağıdaki kodları kullanabiliriz.

```
x1, y1, width, height = cv2.boundingRect(contour)
cv2.rectangle(y, (x1, y1), (x1 + width, y1 + height), (0,255,0), 4)
```

plt.imshow(xFiltreli, cmap='gray'), plt.title('Bulunan nesneler'),plt.axis('off')
plt.subplot(133)
plt.imshow(cv2.cvtColor(y, cv2.COLOR_BGR2RGB)), plt.title('Dairesel nesneler'),plt.axis('off')



Ödev:

cv2.HoughCircles() işlevi ile dairesel nesne bulma ödevi

cv2.HoughLines() işlevi ile çizgisel nesne bulma ödevi