ÇERÇEVELERİN SIRALANMASI VE MERKEZ NOKTALARINA GÖRE GRAFİK ÇİZİMİ

Önceki dokümanda yapılan şekil tanıma ve merkez noktası belirlemeye ek olarak çerçeveler sıralandı ve çerçevelerin merkez noktaları grafik üzerinde gösterildi.

1- Gerekli kütüphaneleri dahil ediyoruz.

import numpy as np  
import matplotlib.pyplot as plt  
import argparse  
import imutils  
import cv2

2- Konturları sıralamak için fonksiyon oluşturuyoruz.

# reverse

False ise soldan sağa veya yukarıdan aşağıya,

True ise sağdan sola veya aşağıdan yukarıya doğru sıralama yapılır.

# i

0 ise x eksenine,

1 ise y eksenine göre sıralama yapılır.

def sort\_contours(cnts, method="left-to-right"):  
 # tersleme değişkenini ve sıralama dizinini başlat  
 reverse = False  
 i = 0  
 # ters sıralama yapmak istersek,  
 if method == "right-to-left" or method == "bottom-to-top":  
 reverse = True  
 # x koordinatı yerine y koordinatına göre sıralama yapıyorsak  
 if method == "top-to-bottom" or method == "bottom-to-top":  
 i = 1  
   
 # sınırlayıcı kutuların listesini oluşturun ve sıralayın  
 boundingBoxes = [cv2.boundingRect(c) for c in cnts]  
 (cnts, boundingBoxes) = zip(\*sorted(zip(cnts, boundingBoxes),  
 key=lambda b:b[1][i], reverse=reverse))  
 # sıralanan konturlar ve sınırlayıcı kutular listesini döndürür  
 return (cnts, boundingBoxes)

3- Konturları çizmek için fonksiyon oluşturuyoruz.

c -> Konturların koordinatları(x,y)

i -> Kontur sıralama numaraları

def draw\_contour(image, c, i):  
 # kontur alanının merkezini hesaplayın.  
  
 M = cv2.moments(c)  
 cx = int(M["m10"] / M["m00"])  
 cy = int(M["m01"] / M["m00"])  
 # Her çerçevenin merkez noktalarını (x -> m, y -> n) dizilere kaydet.  
 m.append(cx)  
 n.append(cy)  
 # Konturun çevresini hesaplıyoruz.  
 peri = cv2.arcLength(c, True)  
 # Köşe sayısını (kordinatlarını) elde ediyoruz  
 approx = cv2.approxPolyDP(c, 0.04 \* peri, True)  
 print(approx)  
 if len(approx) == 3:  
 shape = "Ucgen"  
 elif len(approx) == 4:  
 (x, y, w, h) = cv2.boundingRect(approx)  
 oran = w / float(h)  
  
 if oran == 1:  
 shape = "Kare"  
 else:  
 shape = "Dikdortgen"  
 elif len(approx) == 5:  
 shape = "Besgen"  
 else:  
 shape = "Daire"  
 # kontur sıra numaralarını yazdırın (1,2,3…).  
 cv2.putText(image, "{}".format(i + 1), (cx - 20, cy), cv2.FONT\_HERSHEY\_SIMPLEX,  
 1.0, (255, 255, 255), 2)

# merkez noktasına siyah daire çizin.  
 cv2.circle(image, (cx, cy), 3, (0, 0, 0), -1)

# konturları (her bir çerçevenin çevresini) çizin.  
 cv2.drawContours(image, [c], -1, (255, 100, 255), 2)

# çerçevelerin şekil isimlerini yazın.  
 cv2.putText(image, shape, (cx-20, cy + 30), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2)

return image

4- Bağımsız değişkenleri ayrıştırma işlemi, kod dosyasında değişiklik yapmadan komut satırı üzerinde hangi resmin okunacağını ve sıralama metodunu kolayca değiştirebilmeyi sağladı.

# bağımsız değişken ayrıştırıcısını oluşturma ve bağımsız değişkenleri ayrıştırma  
ap = argparse.ArgumentParser()  
ap.add\_argument("-i", "--image", required=True, help="Path to the input image")  
ap.add\_argument("-m", "--method", required=True, help="Sorting method")  
args = vars(ap.parse\_args())

5- Resmi okuma, gri formata çevirme, filtreleme ve eşikleme işlemleri yapıldı.

image = cv2.imread(args["image"])  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
blurred = cv2.medianBlur(gray, 11)  
thresh = cv2.threshold(blurred, 60, 255, cv2.THRESH\_BINARY)[1]

6- Konturları buluyoruz ve farklı Opencv versiyonları için uygun hale getiriyoruz.

cnts = cv2.findContours(thresh, cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)  
cnts = imutils.grab\_contours(cnts)

7- Konturları ve sıralama metodunu sıralama fonksiyonuna gönderiyoruz.

# konturları sağlanan yönteme göre sıralayın  
(cnts, boundingBoxes) = sort\_contours(cnts, method=args["method"])

8- Bulunan konturlar üzerinde döngü oluşturarak kontur çizme fonksiyonunu çağırıyoruz.

for (i, c) in enumerate(cnts):  
 draw\_contour(image, c, i)

9- Resmi ekranda gösterme, herhangi bir tuşa basılana kadar bekleme ve bütün sekmeleri kapatma işlemlerini yapıyoruz.

cv2.imshow("Sıralı", image)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

10- Figür oluşturuyoruz.

fig = plt.figure()

11- Grafiğin boyutunu ve figür üzerindeki konumunu belirliyoruz.

axes = fig.add\_axes([0.1,0.2,0.8,0.5])

12- Grafiği çizdiriyoruz.

axes.plot(m,n,"red",marker = "o",markerfacecolor="blue",markeredgecolor="yellow")

13- Başlığın ve x, y eksenlerinin isimlerini belirliyoruz.

axes.set\_xlabel("Y")  
axes.set\_ylabel("X")  
axes.set\_title("Çerçevelerin Merkez Noktaları")

14- Grafiği ekranda gösteriyoruz.

plt.show()

15- Komut satırında aşağıdaki komutu çalıştırıyoruz.

NOT: Resim ve dosya konumlarına dikkat edilmelidir.

python frame\_sort.py --image im.png --method "right-to-left"

NOT: Sağdan sola doğru sıralama yapılmıştır. Metodu değiştirmek için aşağıdaki komutları kullabilirsiniz.

“left-to-right “ -> Solsan sağa

“top-to-bottom” -> Yukarıdan aşağıya

“bottom-to-top” -> Aşağıdan yukarıya

