Python Kodunun Açıklamaları

import cv2
import numpy as np

cv2: OpenCV kütüphanesini içe aktarıyoruz; bu, görüntü işleme fonksiyonlarını sağlar.
numpy: OpenCV ile beraber kullanmak için numpy kütüphanesini içe aktarıyoruz. numpy, sayısal işlemler ve matris işlemleri için kullanılır.

cap = cv2.VideoCapture(0)

cv2.VideoCapture(0): Bilgisayara bağlı olan kamerayı başlatır. (0), varsayılan kamerayı (genellikle dahili kamera) belirtir. Eğer harici bir kamera varsa, (1) gibi başka bir indeks ile çağırılabilir.

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
```

while True: Sonsuz döngü başlatılır. Kamera sürekli olarak çalışır ve yeni görüntüler yakalar.
ret, frame = cap.read(): Kameradan bir kare (frame) yakalar. ret değeri, kameranın düzgün çalışıp çalışmadığını kontrol eden bir bayraktır (True veya False döner). frame, kameradan alınan görüntü verisidir.
if not ret: break: Eğer ret değeri False ise döngüyü kırarak programı sonlandırır. Bu, kameranın düzgün çalışmadığını belirtir.

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY): frame içindeki renkli görüntüyü gri tonlara dönüştürür. Bu, kenar algılama gibi işlemler için gereklidir çünkü renkli görüntülerde işlem yapmak yerine gri tonlu görüntüde çalışmak daha hızlıdır.

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

cv2.GaussianBlur(gray, (5, 5), 0): gray görüntüsüne Gauss bulanıklaştırma filtresi uygular. (5, 5), bulanıklaştırma çekirdeğinin boyutunu belirler. Bu işlem, görüntüdeki gürültüyü azaltır ve kenar tespitini iyileştirir.

edges = cv2.Canny(blurred, 200, 450)

cv2.Canny(blurred, 200, 450): Canny kenar algılama işlemi yapar. 200 ve 450, kenar tespitinde kullanılan alt ve üst eşik değerleridir. Yüksek değerler, sadece en belirgin kenarları bulmaya yarar.

contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE): Kenarlarda bulunan konturları (sınırları) bulur. edges görüntüsünde algılanan şekillerin konturlarını çıkartır.
cv2.RETR_EXTERNAL: Sadece dış konturları bulur, iç konturları göz ardı eder.
cv2.CHAIN_APPROX_SIMPLE: Gereksiz noktaları kaldırarak konturları basitleştirir. Sadece gerekli köşe noktalarını döndürür.

shape_count = 0

shape_count = 0: Bulunan ve gösterilecek şekil sayısını tutmak için sayaç değişkeni. Bu sayaç, maksimum 5 şekil sınırını korumak için kullanılır.

for contour in contours:

if shape_count >= 5:

break

for contour in contours: Bulunan her bir konturu (şekil sınırlarını) döngüyle işler.
if shape_count >= 5: break: shape_count 5 veya daha fazlaysa döngüyü kırarak yeni şekillerin çizilmesini engeller.

epsilon = 0.02 * cv2.arcLength(contour, True)

cv2.arcLength(contour, True): contour uzunluğunu (çevresini) hesaplar. True, konturun kapalı bir şekil olduğunu belirtir.
epsilon = 0.02 * cv2.arcLength(contour, True): Şeklin çevresinin %2'si kadar bir değer alır. Bu, köşeleri basitleştirmek için epsilon adı verilen hata payıdır.

approx = cv2.approxPolyDP(contour, epsilon, True)

cv2.approxPolyDP(contour, epsilon, True): Konturu, verilen epsilon değeri ile yaklaştırır ve şeklin köşe sayısını basitleştirir. Böylece şeklin kaç köşeli olduğu belirlenebilir.

num_vertices = len(approx)

len(approx): approx içinde bulunan köşe noktalarının sayısını hesaplar. Bu, şeklin köşe sayısını ifade eder.

```
shape = 'Bilinmeyen'
if num_vertices == 3:
    shape = 'Triangle'
elif num_vertices == 4:
    shape = 'Rectangle'
elif num_vertices > 8:
    shape = 'Circle'
```

shape = 'Bilinmeyen': Şeklin adını tutacak bir değişken, varsayılan olarak "Bilinmeyen" olarak ayarlanır.
if num_vertices == 3: Eğer köşe sayısı 3 ise, şekil bir 'Triangle' (üçgen) olarak değerlendirilir.
elif num_vertices == 4: Eğer köşe sayısı 4 ise, şekil bir 'Rectangle' (dikdörtgen) olarak değerlendirilir.
elif num_vertices > 8: Eğer köşe sayısı 8'den fazla ise, şekil bir 'Circle' (daire) olarak değerlendirilir.

cv2.drawContours(frame, [approx], 0, (0, 255, 0), 2)

cv2.drawContours(frame, [approx], 0, (0, 255, 0), 2): frame üzerinde şekli çizer. [approx], çizilecek konturu temsil eder.
(0, 255, 0): Çizim rengi (yeşil).
2: Çizgi kalınlığı.

x, y = approx.ravel()[0], approx.ravel()[1] - 10

approx.ravel(): approx içinde bulunan koordinatları tek bir diziye çevirir.
[0] ve [1] - 10: Metin konumunun x ve y koordinatlarını belirler. Y değeri, metnin şeklin hemen üstünde görünmesi için 10 piksel yukarı kaydırılır.

cv2.putText(frame, shape, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.putText(...): frame üzerinde şeklin ismini çizer.
shape: Yazılacak metin (şeklin ismi).
(x, y): Metnin pozisyonu.
cv2.FONT_HERSHEY_SIMPLEX: Yazı tipi.
0.5: Yazı boyutu.
(0, 255, 0): Yazı rengi (yeşil).
2: Yazı kalınlığı.

shape_count += 1

shape_count += 1: Tespit edilen ve çizilen şekil sayısını bir artırır.

cv2.imshow('Kamera', frame)

cv2.imshow('Kamera', frame): frame görüntüsünü ekranda 'Kamera' başlığıyla gösterir.

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

cv2.waitKey(1): 1 milisaniye bekler. Herhangi bir tuşa basılırsa ASCII değerini döner.
& 0xFF == ord('q'): Basılan tuş 'q' ise döngüyü kırarak programı sonlandırır.

```
cap.release()
cv2.destroyAllWindows()
```

cap.release(): Kamerayı serbest bırakır.
cv2.destroyAllWindows(): Açık olan tüm OpenCV pencerelerini kapatır.