OPENCV Video İşlemleri

OpenCV'de video okuma, bilgisayarla görme ve görüntü işlemenin çok önemli bir yönüdür. OpenCV, bilgisayarla görme görevleri için çok çeşitli işlevler ve algoritmalar sağlayan açık kaynaklı bir kütüphanedir.

OpenCV'de video okuma, bir kameradan veya bir dosyadan video yakalama ve analiz ve işleme için videonun karelerini yorumlama işlemidir. Bu, gözetim, nesne izleme ve video düzenleme gibi bilgisayarla görme uygulamalarında yaygın bir görevdir.

OpenCV'de bir videoyu okumak için ilk adım **cv2.VideoCapture()** fonksiyonunu kullanarak bir video yakalama nesnesi oluşturmaktır. Bu fonksiyon, video dosyasının veya video akışının yolunu bir argüman olarak alır.

Video yakalama nesnesi oluşturulduktan sonra, videodan tek tek kareleri okumak için nesnenin read() yöntemini kullanabiliriz. Bu yöntem, okuma işleminin durumunu ve karenin (çerçevenin) kendisini içeren bir veri döndürür.

Burada "ret" mantıksal değişkenini (True or False) kullanarak video okumanın başarılı olup olmadığını test edebiliriz.

Kareleri görüntülemek için, pencere adını ve kareyi argüman olarak alan **cv2.imshow()** fonksiyonunu kullanabiliriz. Çerçevelerin düzgün görüntülenmesi için **imshow()** fonksiyonundan sonra **cv2.waitKey()** fonksiyonunun kullanılması önemlidir.

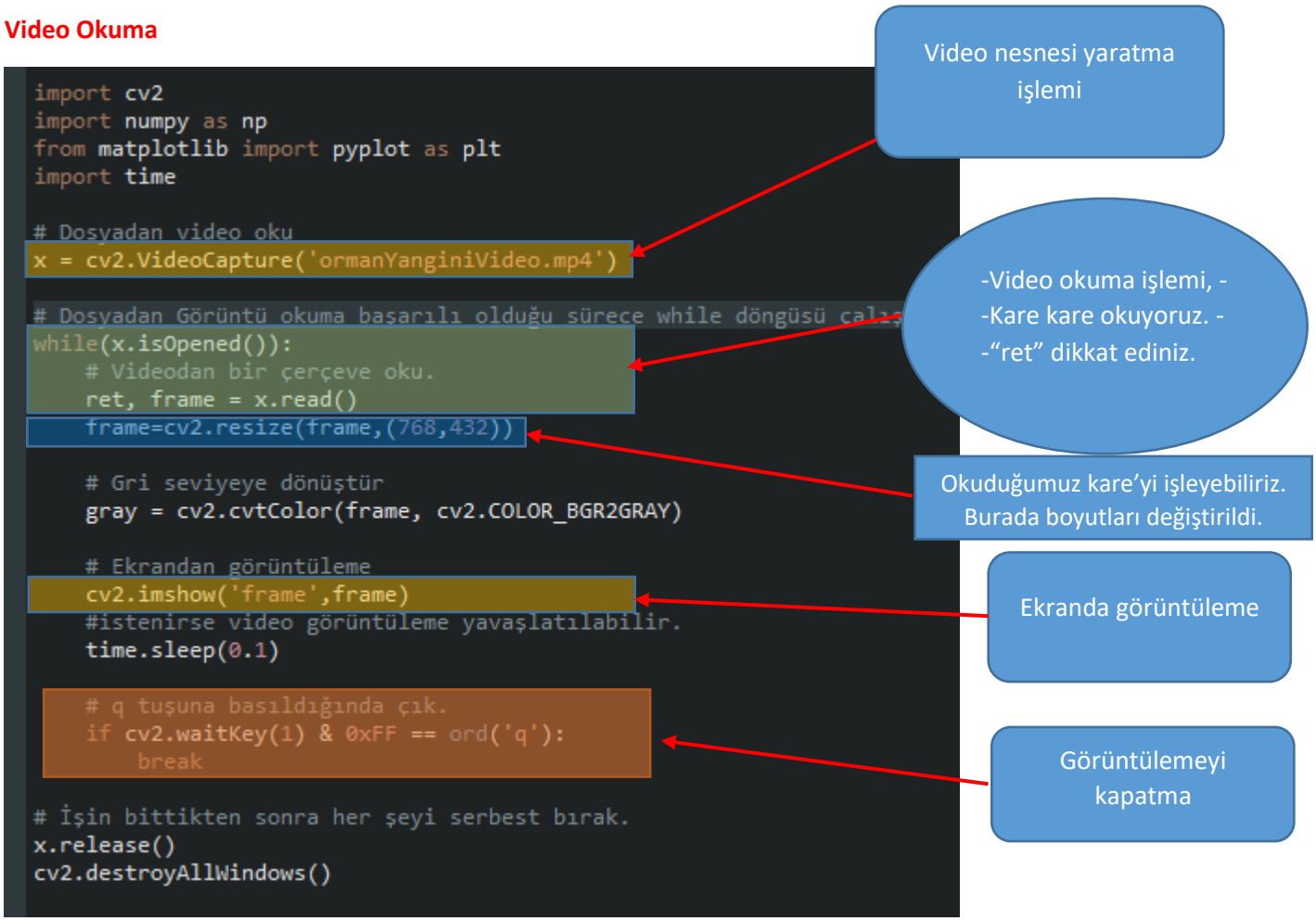
OpenCV, kareleri okuma ve görüntülemeye ek olarak, kareleri yeniden boyutlandırma, döndürme ve çevirme gibi temel video manipülasyonu için işlevler de sağlar. Bu işlevler, daha fazla işlem veya analiz gerçekleştirmeden önce kareleri önceden işlemek için kullanılabilir.

cv2.VideoWriter() fonksiyonunu kullanarak kareleri yeni bir videoya yazmak da mümkündür. Bu fonksiyon çıkış video dosyası adını, fourcc kodunu, kare hızını ve kare boyutunu argüman olarak alır. Fourcc kodu, video codec bileşenini belirtmek için kullanılan 4 baytlık bir koddur.

Genel olarak, OpenCV'de video okuma, bilgisayarla görme uygulamaları için güçlü ve çok yönlü bir araçtır. Video verilerinin analizine ve manipülasyonuna izin vererek gözetim, video düzenleme ve nesne izleme gibi alanlarda çok çeşitli uygulamalara olanak tanır.

İlk Örnek:

Video Okuma



Video Yazma

OpenCV'deki cv2.VideoWriter() işlevi, video karelerini bir dosyaya veya kameraya yazmak için kullanılabilen bir VideoWriter nesnesi oluşturmak için kullanılır. Bu, video düzenleme, nesne izleme ve gözetim gibi bilgisayarla görme uygulamalarında yaygın bir görevdir.

Bir VideoWriter nesnesi oluşturmak için aşağıdaki parametreleri belirtmeniz gerekir:

- Çıktı dosyası yolu ve adı (bir dosyaya yazılıyorsa)
- Video codec bileşenini belirtmek için kullanılan 4 baytlık bir kod olan FourCC kodu
- Videonun kare hızı (saniye başına kare olarak)
- Video karelerinin boyutu (piksel cinsinden)

Örneğin, saniyede 80 kare hızı ve 640x480 piksel kare boyutuyla "output.mp4" adlı bir dosyaya yazmak üzere bir VideoWriter nesnesi oluşturmak için aşağıdaki kodu kullanırsınız:

```
fourCC=cv2.VideoWriter_fourcc(*'XVID')
yaz = cv2.VideoWriter("output.mp4", fourCC, 80, (640, 480))
```

Video karelerini yazmayı bitirdiğinizde VideoWriter nesnesinin serbest bırakılması gerektiğine dikkat etmek önemlidir. Bu, nesne üzerinde release() yöntemi çağrılarak yapılır.

Genel olarak, OpenCV'deki cv2.VideoWriter() işlevi, video karelerini bir dosyaya veya kameraya yazmak için güçlü bir araçtır. Çok çeşitli bilgisayarla görme uygulamalarına kolay entegrasyon sağlar.

Örnek:

```
import cv2
# Aşağıdaki parametrelerle bir VideoWriter nesnesi oluşturun:
# - Çıktı dosyası yolu ve adı
# - FourCC kodu (video kodekini belirtmek için kullanılan 4 baytlık bir kod)
# - Videonun kare hızı
# - Video karelerinin boyutu

#VideoCapture nesnesi kullanarak video çerçevelerini okuyun
x = cv2.VideoCapture('daria_walk.avi')
ret, frame = x.read()
h, w, _= frame.shape # Genişlik ve yükseklik elde etmek için çerçeveyi kullanın

frameTime = 100
fourcc = cv2.VideoWriter_fourcc(*"XVID") # XVID herhangi bir şekilde değiştirilebilir
fps = 1000 / frameTime # fps oranını hesapla
# Video yazma nesnesi
yaz = cv2.VideoWriter("Pothole testing 2.mp4", fourcc, fps, (w, h))

while ret:
    frame=frame+40
    yaz.write(frame)
    cv2.imshow("Frame", frame)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
    ret, frame=x.read()
# VideoCapture ve VideoWriter nesnelerini serbest bırakın
x.release()
yaz.release()
# Herhangi bir açık pencereyi yok edin (isteğe bağlı)
cv2.destroyAllWindows()
```

Kameradan Canlı Görüntü Okuma

Bilgisayarın dahili kamerasından veya USB/Ethernet vs. ile bağlanan kameralardan alınan görüntüleri okuyup ekrandan görüntüleyebiliriz. Aşağıda bir örnek kod verilmiştir.

```python
import numpy as np
import cv2

cap = cv2.VideoCapture(1)

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:

        cv2.imshow('frame',frame)

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

# Herşeyi serbest bırak
cap.release()
cv2.destroyAllWindows()
```

Burada '0' kullanılırsa dahili kamera, '1' kullanılırsa diğer kameralar devreye girer.

Döngü içinde okuma yapılmalıdır.

Ret=True ise okuma başarılıdır.

Kameradan Canlı Görüntü Okuma ve Kaydetme

Videodan yazma işleminden tek farkı görüntü kaynağının PC'nin entegre kamerası ya da USB'den bağlı kamera olmasıdır.

```python
import cv2

cap = cv2.VideoCapture(0)

# CODEC tanımı yapılıyor.
fourcc = cv2.VideoWriter_fourcc(*'MP4v')
#Yazma nesnesi oluşturuldu
out = cv2.VideoWriter('output2023.mp4',fourcc, 30, (640,480))

while(cap.isOpened()):
    ret, frame = cap.read()

    if ret==True:
        frame = cv2.flip(frame,0) #Çerçeve'yi (kare) ters çevir

        # Çerçeve'yi yaz.
        out.write(frame)
        cv2.imshow('frame',frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
# Herşeyi serbest bırak
cap.release()
out.release()
cv2.destroyAllWindows()
```

Kameradan Alınan Görüntüleri İşleme:

Nesne Tespiti

Video görüntü işleme, bir dizi görüntünün bir zaman çizelgesi boyunca ele alınarak analiz edilmesini içeren bir alandır. Bu alan, genellikle bir video akışından anlam çıkarma, nesne tanıma, hareket analizi, izleme ve video sıkıştırma gibi konuları kapsar. Video görüntü işleme, bir dizi statik görüntüden daha karmaşık bilgi çıkarmak ve dinamik sahneleri anlamak için özel algoritmalar, filtreler ve teknikler gerektirir.

Hareket analizi, bir video içindeki nesnelerin hareketini izleme ve analiz etme sürecidir. Nesne tanıma, bir video akışındaki nesneleri tanımlama ve sınıflandırma çabasını içerir. İzleme, bir nesnenin bir video içinde nasıl hareket ettiğini takip etme yeteneğini ifade eder. Bu konuların birleşimi, güvenlik sistemlerinde, trafik kontrolünde, medikal görüntüleme uygulamalarında ve birçok endüstriyel alanda kullanılan çeşitli uygulamaların temelini oluşturur.

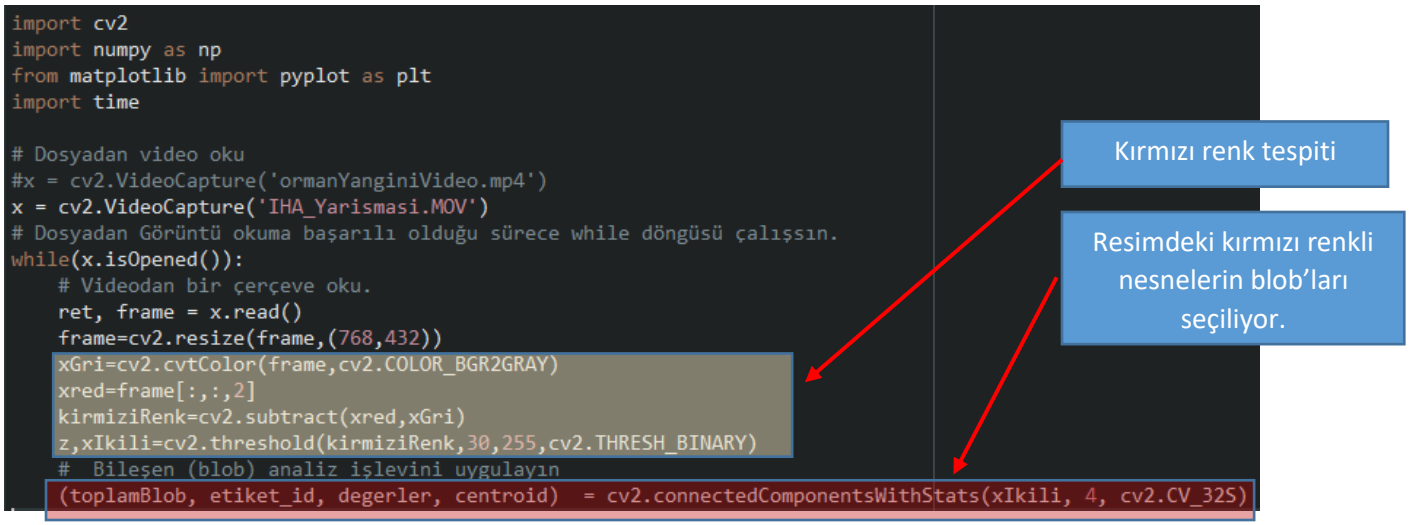
Örnek:

```python
#Kamera Kırmızı Nesne Bulma
import cv2

# Kamera örüntülerini okumaya başkla
x = cv2.VideoCapture(0)

while(x.isOpened()):
    # Videodan bir çerçeve oku.
    ret, frame = x.read()
    frame=cv2.resize(frame,(768,432))
    xGri=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    xred=frame[:,:,0]
    kirmiziRenk=cv2.subtract(xred,xGri)
    z,xIkili=cv2.threshold(kirmiziRenk,70,255,cv2.THRESH_BINARY)
    #  Bileşen (blob) analiz işlevini uygulayın
    (toplamBlob, etiket_id, degerler, centroid)  = cv2.connectedComponentsWithStats(xIkili, 4, cv2.CV_32S)
    # Her bir blob için işlem yapılıyor.
    for i in range(1, toplamBlob):
        # Blobların alanları
        area = degerler[i, cv2.CC_STAT_AREA]
        print(area)
        # 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.
        pt1 = (x1, y1)
        pt2 = (x1+ w, y1+ h)
        (X, Y) = centroid[i]
        # Herbir blob'a kutu çiziliyor.
        cv2.rectangle(frame,pt1,pt2,(0, 0, 255), 2)
```

```python
    # Ekrandan görüntüleme
    cv2.imshow('frame',frame)
    cv2.imshow('Ikili',xIkili)

    # q tuşuna basıldığında çık.
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# İşin bittikten sonra her şeyi serbest bırak.
x.release()
cv2.destroyAllWindows()
```

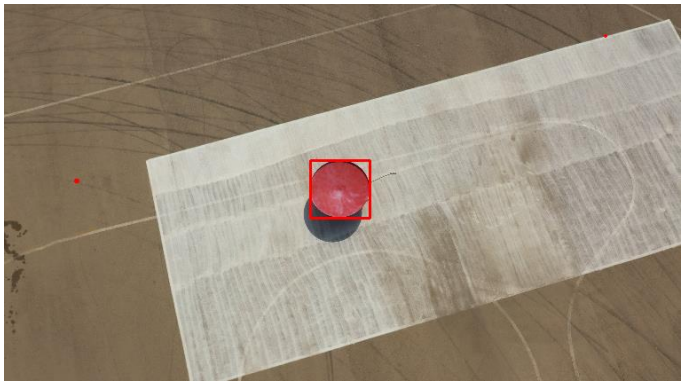
Video Görüntülerinde Nesne Tespiti

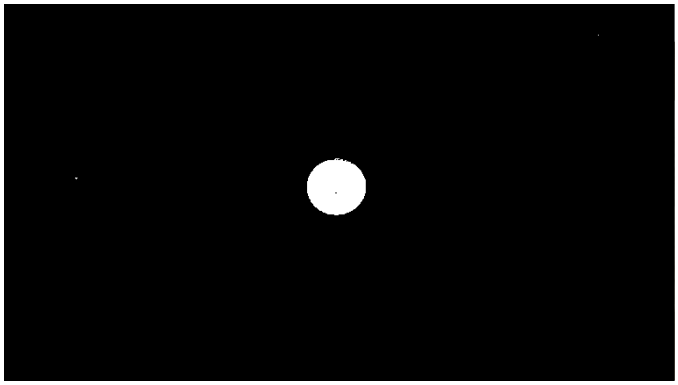
Görüntü işlemede nesne tespiti, bir görüntünün içindeki nesneleri tanımak ve konumlarını belirlemek için kullanılan bir tekniktir. Bu işlem genellikle bir görüntünün içindeki nesneleri sınıflandırmak için kullanılır ve bu sınıflandırma birçok farklı amaç için kullanılabilir. Örneğin, bir güvenlik kamerasında nesne tespiti kullanılarak insanların ve araçların hareketleri takip edilebilir veya bir alışveriş merkezinde insanların nerelerde daha fazla zaman geçirdiği belirlenebilir. Örnek çalışmamızda video görüntüsündeki belirli bir renge odaklanacağız ve o renkteki nesneleri kapsayıcı kutu içine almaya çalışacağız.



```
    # Her bir blob için işlem yapılıyor.
    for i in range(1, toplamBlob):
        # Blobların alanları
        area = degerler[i, cv2.CC_STAT_AREA]
        print(area)
        # 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.
        pt1 = (x1, y1)
        pt2 = (x1+ w, y1+ h)
        (X, Y) = centroid[i]
        # Herbir blob'a kutu çiziliyor.
        cv2.rectangle(frame,pt1,pt2,(0, 0, 255), 2)

    # Ekrandan görüntüleme
    cv2.imshow('frame',frame)
    cv2.imshow('Ikili',xIkili)
    #istenirse video görüntüleme yavaşlatılabilir.
    time.sleep(0.1)

    # q tuşuna basıldığında çık.
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# İşin bittikten sonra her şeyi serbest bırak.
x.release()
cv2.destroyAllWindows()
```

Sonuç1: (Yarışma Hedef Görüntüsü)





Sonuç2: (Orman yangını videosu)