**Real-time Object Tracking with OpenCV and YOLOv8 in Python**

**Projenin amacı**

Bu projede, YOLOv8 modelini ve DeepSORT algoritmasını birleştirerek video akışı veya canlı kamera görüntüleri üzerinden gerçek zamanlı nesne algılama ve izleme yapılır. YOLOv8, kareler içindeki nesneleri tanımlayıp sınıflandırmak için kullanılırken, DeepSORT algoritması bu nesnelerin hareketlerini takip etmek için kullanılır.

**Object\_detection.py**

Bu kod, YOLOv8 modelini kullanarak bir video dosyasındaki nesneleri algılar, bunların üzerine dikdörtgen kutular çizer, FPS (Frame Per Second) değerini hesaplar ve çıktı olarak işlenmiş kareleri bir dosyaya yazar. Kodun işleyişi adım adım şu şekildedir:

**1-Kütüphanelerin aktarılması**

import datetime //datetime: İşlenen her kare için geçen zamanı hesaplamak ve FPS değerini belirlemek için kullanılıyor.

from ultralytics import YOLO

import cv2 //video ve görüntü işlemleri için kullanılıyor.

from helper import create\_video\_writer //Video dosyasına yazma işlemi için kullanılıyor.

**2- Sabit değerlerin tanımlanması**

CONFIDENCE\_THRESHOLD = 0.8 //güven değeri , 0.8’i aşan nesneler ekrana çizilir

GREEN = (0, 255, 0) //Çizilecek kutunun rengi

**3-Video dosyasını okuma, yazma nesneleri**

video\_cap = cv2.VideoCapture("2.mp4") //"2.mp4" adlı videoyu okuyan bir nesne oluşturuyor

writer = create\_video\_writer(video\_cap, "output.mp4") // create\_video\_writer fonksiyonu, okunan video özelliklerine göre "output.mp4" adlı bir çıktı video dosyası başlatıyor.

**4-YOLO modelinin yüklenmesi**

model = YOLO("yolov8n.pt")

**5-Video kareleri üzerinde döngü**

while True: //videodaki her kareyi işlemesi için

start = datetime.datetime.now() //Her karenin işlenme süresini ölçmek için başlangıç zamanı olarak ayarlanıyor.

ret, frame = video\_cap.read() //video\_cap nesnesiyle bir kare alınıyor. ret doğruysa kare başarılı şekilde alındı anlamına geliyor.

if not ret: //kare bulunamıyorsa videodan çıkılır

break

**6-Nesne tespiti ve işaretleme**

detections = model(frame)[0] //YOLO modeli ile kare üzerinde nesne tespiti yapılır ve tespit edilen nesnelerin bilgileri detections değişkenine atanır.

for data in detections.boxes.data.tolist(): //Tespit edilen her nesne üzerinde döngü başlatılır.

confidence = data[4] //Tespit edilen nesnenin güvenilirlik oranı alınır.

if float(confidence) < CONFIDENCE\_THRESHOLD: //Güvenilirlik oranı CONFIDENCE\_THRESHOLD değerinden düşükse bu nesne yok sayılır.

continue

xmin, ymin, xmax, ymax = int(data[0]), int(data[1]), int(data[2]), int(data[3]) //Nesnenin dikdörtgen sınırlarının koordinatları alınır.

cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), GREEN, 2) //Kare üzerine belirlenen koordinatlar arasında yeşil renkte dikdörtgen çizilir.

**7-FPS hesaplama ve gösterme**

end = datetime.datetime.now() //Her karenin işlenme süresini ölçmek için bitiş zamanı ayarlanır.

total = (end - start).total\_seconds() //Kare işleme süresi hesaplanır.

fps = f"FPS: {1 / total:.2f}" //FPS değeri hesaplanır ve bir formatla yazı haline getirilir.

cv2.putText(frame, fps, (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255), 8) //fps değeri kare üzerine eklenir.

**8-Kareyi çıktı olarak gösterme**

cv2.imshow("Frame", frame) //İşlenmiş kareyi ekranda gösterir.

writer.write(frame) //İşlenmiş kareyi çıktı videosuna yazar.

if cv2.waitKey(1) == ord("q"): //q tuşuna basıldığında döngüden çıkar.

break

**9- Kaynakların serbest bırakılması**

video\_cap.release()

writer.release()

cv2.destroyAllWindows()

**Helper.py**

Bu fonksiyon, giriş videosunun çözünürlüğü ve fps değerine göre MP4 formatında bir çıktı dosyası başlatır. Kodun geri kalanında her kare işlendiğinde bu writer nesnesi kullanılarak çıktı dosyasına kareler eklenir.

**1-Fonksiyon parametreleri**

def create\_video\_writer(video\_cap, output\_filename): //video\_cap: OpenCV VideoCapture nesnesi, yani giriş video kaynağı (önceden açılmış video dosyası) bu parametreyle veriliyor. output\_filename: Çıktı videosunun dosya adı (örneğin, "output.mp4").

**2-Genişlik,yükseklik fps bilgilerinin alınması**

frame\_width = int(video\_cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)) //Giriş videosunun kare genişliğini (pixel cinsinden) alır.

frame\_height = int(video\_cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)) //Giriş videosunun kare yüksekliği

fps = int(video\_cap.get(cv2.CAP\_PROP\_FPS)) //Giriş videosunun saniyede kaç kare (frame per second - fps) gösterdiğini alır.

**3-Video Codec Belirleme ve VideoWriter Nesnesini Başlatma**

fourcc = cv2.VideoWriter\_fourcc(\*'MP4V') //MP4V kodlayıcısını (codec) kullanarak FourCC kodunu belirler. Bu, çıktı videosunun MP4 formatında olması için gerekli kodlayıcıyı seçer.

writer = cv2.VideoWriter(output\_filename, fourcc, fps,

(frame\_width, frame\_height))

// cv2.VideoWriter(...): Video yazma işlemini başlatan nesne oluşturulur:

* output\_filename: Çıktı dosyasının adı.
* fourcc: Belirlenen codec (MP4V).
* fps: Giriş videosundaki kare hızına uyum sağlamak için alınan fps değeri.
* (frame\_width, frame\_height): Giriş videosunun çözünürlüğünü belirleyen genişlik ve yükseklik değerleri.

**4-Fonksiyon sonucu**

return writer //cv2.VideoWriter nesnesi, çıktı videosuna kare eklemek ve işlenmiş videoyu kaydetmek için geri döndürülür.

**Object\_detection\_tracking.py**

Bu kod, bir video dosyasındaki nesneleri tespit etmek ve takip etmek için YOLOv8 ve Deep SORT algoritmalarını kullanarak gerçek zamanlı nesne algılama ve izleme yapar. İşlemi kare kare gerçekleştirir, FPS değerini hesaplar ve takip edilen nesneleri çerçeveye çizerek görüntüler. Kodu adım adım inceleyelim:

**1-Kütüphanelerin içe aktarılması**

import datetime //Kare işleme süresini ölçmek için kullanılır.

from ultralytics import YOLO

import cv2

from helper import create\_video\_writer

from deep\_sort\_realtime.deepsort\_tracker import DeepSort //Nesne takip algoritması.

**2-Sabit değerlerin tanımlanması**

CONFIDENCE\_THRESHOLD = 0.8 //Tespit edilen nesnenin güven puanı bu değerin altında ise, nesne geçersiz sayılır.

GREEN = (0, 255, 0)

WHITE = (255, 255, 255) //Kareye çizilecek nesne ve takip id’lerinin renkleri.

**3-Video giriş ve çıkış nesnelerinin oluşturulması**

video\_cap = cv2.VideoCapture("2.mp4") //Giriş videosunu (örneğin, 2.mp4) açar.

writer = create\_video\_writer(video\_cap, "output.mp4") //Çıktı videosu için yazıcıyı output.mp4 dosyasına yazar.

**4-Model ve takip nesnesinin yüklenmesi**

model = YOLO("yolov8n.pt")

tracker = DeepSort(max\_age=50) //Deep SORT algoritması ile nesne takibi sağlar.

**5- Döngü başlangıcı**

while True:

start = datetime.datetime.now()

ret, frame = video\_cap.read() //Bir kare okur; daha fazla kare yoksa döngüden çıkar.

if not ret:

break

**6- Nesne tespiti**

detections = model(frame)[0] //YOLO modeli ile karedeki nesneleri tespit eder.

results = [] //Tespit edilen nesnelerin detaylarını (koordinatlar, güven puanı, sınıf kimliği) saklar.

**7- Tespit edilen nesneler üzerinde döngü**

for data in detections.boxes.data.tolist(): //Tespit edilen nesnelerin her birinin bilgilerini alır.

confidence = data[4]

if float(confidence) < CONFIDENCE\_THRESHOLD:

continue

xmin, ymin, xmax, ymax = int(data[0]), int(data[1]), int(data[2]), int(data[3])

class\_id = int(data[5]) //Tespit edilen nesnenin sınıf kimliği.

results.append([[xmin, ymin, xmax - xmin, ymax - ymin], confidence, class\_id]) //Her nesnenin (x, y, genişlik, yükseklik) koordinatlarını, güven puanını ve sınıf kimliğini kaydeder.

**8-Nesne takibi**

tracks = tracker.update\_tracks(results, frame=frame) //results listesindeki verileri kullanarak her nesneyi takip eder.

for track in tracks:

if not track.is\_confirmed(): //Sadece doğrulanan takipleri işler.

continue

track\_id = track.track\_id //nesnenin takip kimliği

ltrb = track.to\_ltrb() //xmin, ymin, xmax, ymax şeklinde takip edilen nesnenin sınırlarını alır.

xmin, ymin, xmax, ymax = int(ltrb[0]), int(ltrb[1]), int(ltrb[2]), int(ltrb[3])

//cv2.rectangle ve cv2.putText: Çerçeveye nesne ve takip kimlikleri çizilir.

cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), GREEN, 2)

cv2.rectangle(frame, (xmin, ymin - 20), (xmin + 20, ymin), GREEN, -1)

cv2.putText(frame, str(track\_id), (xmin + 5, ymin - 8), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, WHITE, 2)

**9- FPS hesaplama ve çizimi**

end = datetime.datetime.now() //end: Bir kareyi işleme süresinin bitiş zamanını alır.

print(f"Time to process 1 frame: {(end - start).total\_seconds() \* 1000:.0f} milliseconds")

fps = f"FPS: {1 / (end - start).total\_seconds():.2f}" //fps: Bir kareyi işleme süresinden hesaplanan FPS değeri kareye çizilir.

cv2.putText(frame, fps, (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255), 8)

**10-çerçeveyi görüntüleme ve kaydetme**

cv2.imshow("Frame", frame) //Çerçeveyi ekranda gösterir.

writer.write(frame) // Çıktı videosuna çerçeveyi kaydeder.

if cv2.waitKey(1) == ord("q"): //q tuşuna basınca döngü sona erer.

Break

**11-Kaynakları serbest bırakma**

video\_cap.release()

writer.release()

cv2.destroyAllWindows()