

TC.

Selçuk Üniversitesi

TEKNOLOJİ FAKÜLTESİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

YÜZ TANIYAN VE TAKİP EDEN KAMERA

Öğrenci Adı-Soyadı:

Bilge Kaan ÖZTÜRK 213302002

Sorumlu Akademisyen:

Kemal TÜTÜNCÜ



SELÇUK
ÜNİVERSİTESİ

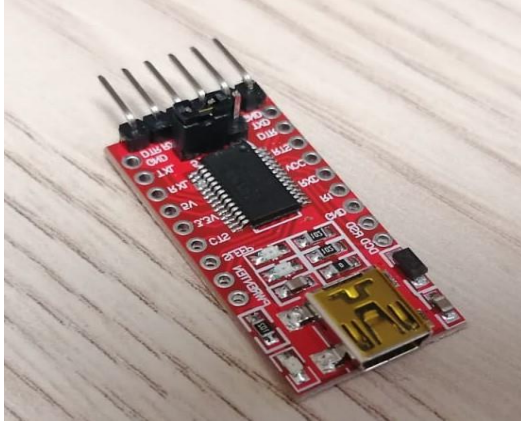
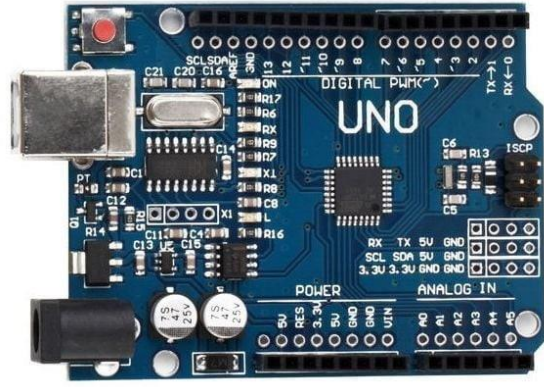
İçindekiler

Proje Kapağı.....	1
İçindekiler.....	2
1)Giriş.....	3
2)Karşılaşılan Problemler.....	4
3)Kullanılan Eleman ve Eklentiler.....	5
a) Esp32.....	.5
b) Arduino UNO.....	5
c) Tower Pro SG90 9G Servo.....	5
d) Cmake.....	.5
4)Projenin Çalışma Mantığı.....	6
a) Esp32 Yayını.....	6
b) Yüz Tanıma Gereklilikleri.....	8
c) Haar Cascade ile Hazır Model.....	11
d) Kendi Yolov8 Modelinizi Oluşturma.....	14
5)Projenin Kodları.....	23
a) Python Kodları.....	23
b) Arduino Kodları.....	27
6)Bütün Hatları ile Proje İncelemesi.....	29
7)Kaynakça.....	30

1)Giriş

Projenin Hedefi ve İhtiyaçlar

Öncelikle bu projenin amacının gerçek zamanlı yüz tanıma ve takibi yapabilen bir kamera tasarlamak olduğunu belirtmek isterim.Gerçek zamanlı bir yüz tanıma ve takip işlemi yapabilmek için bu projede esp32 ve Arduino UNO mikrokontrolcöleri kameranin hareketini sağlayabilmek amacı ile bir adet Tower Pro SG90 9G Servo kullanılmıştır.Esp32 mikrokontrolcüsü kamera modölüne sahip olduđu için ekstra bir kameraya ihtiyaç duyulmamıştır.Gerçek zamanlı yüz tanıma işlemi için Haar cascade ve ayrıca özel model için YoloV8 kullanılmış olup pyserial kütüphanesi ile iki cihaz arasında seri iletişim ile veri aktarımı gerçekleştirilmiştir.Aşağıda kullanılan cihazları görebilirsiniz.



2) Karşılaşılan Problemler

Böyle bir proje tasarlanırken dikkat edilmesi gereken bazı hususlar öne çıkmaktadır. Öncelikle kullanacağınız ana mikrokontrolcünün hafıza ve işlem yapma kabiliyetinin yeterli olup olmadığının daha projeye başlamadan farkında olmalı ve bunun bilincinde hareket etmelisiniz. Şahsen ben önemli hususa dikkat etmediğim projemde aslında yanlış bir eleman tercihi yaparak esp32 kullanmış durumdayım ve bu sebeple mikroişlemcinin içinde yapabileceğim yüz tanıma işlemini ayrı bir bilgisayarda python IDE'sinde yaparak işlemde bir zaman komplikasyonu çıkma ihtimaline sebep oldum. Bu problem aynı zamanda bir Arduino UNO kullanılmasını zorunlu kıldı

ama bu sayede bir işi yapmak için yeni yollar keşfettik ve Raspberry Pi gibi daha yüksek maliyetli bir mikroişlemci yerine daha düşük bir bütçeyle projeyi tamamladık.

3)Kullanılan Eleman ve Eklentiler

i)Esp32: “ESP32; Bluetooth ve Wi-Fi özelliği olan, düşük maliyetli ve düşük güçlü bir mikrodenetleyici sistemdir.”⁽¹⁾ Espressif adlı şirket tarafından aslında çok amaçlı bir kullanım alanına da sahiptir.

ii)Arduino UNO: “Arduino UNO, çeşitli elektronik projelere entegre edilebilen, düşük maliyetli, programlanabilir, açık kaynaklı,

esnek ve kullanımı kolay bir mikro denetleyici kartıdır.”⁽²⁾ bu projede alınan konum verisini işleyerek motoru hareket ettirmektedir.

iii) Tower Pro SG90 9G Servo: “Tower Pro SG90 9G Servo Motor Mini, küçük boyutlu projeler geliştirebilmeniz için son derece uygun bir servodur.”⁽³⁾ bu projede de kameranın hareketinin sağlanması için kullanılmıştır.

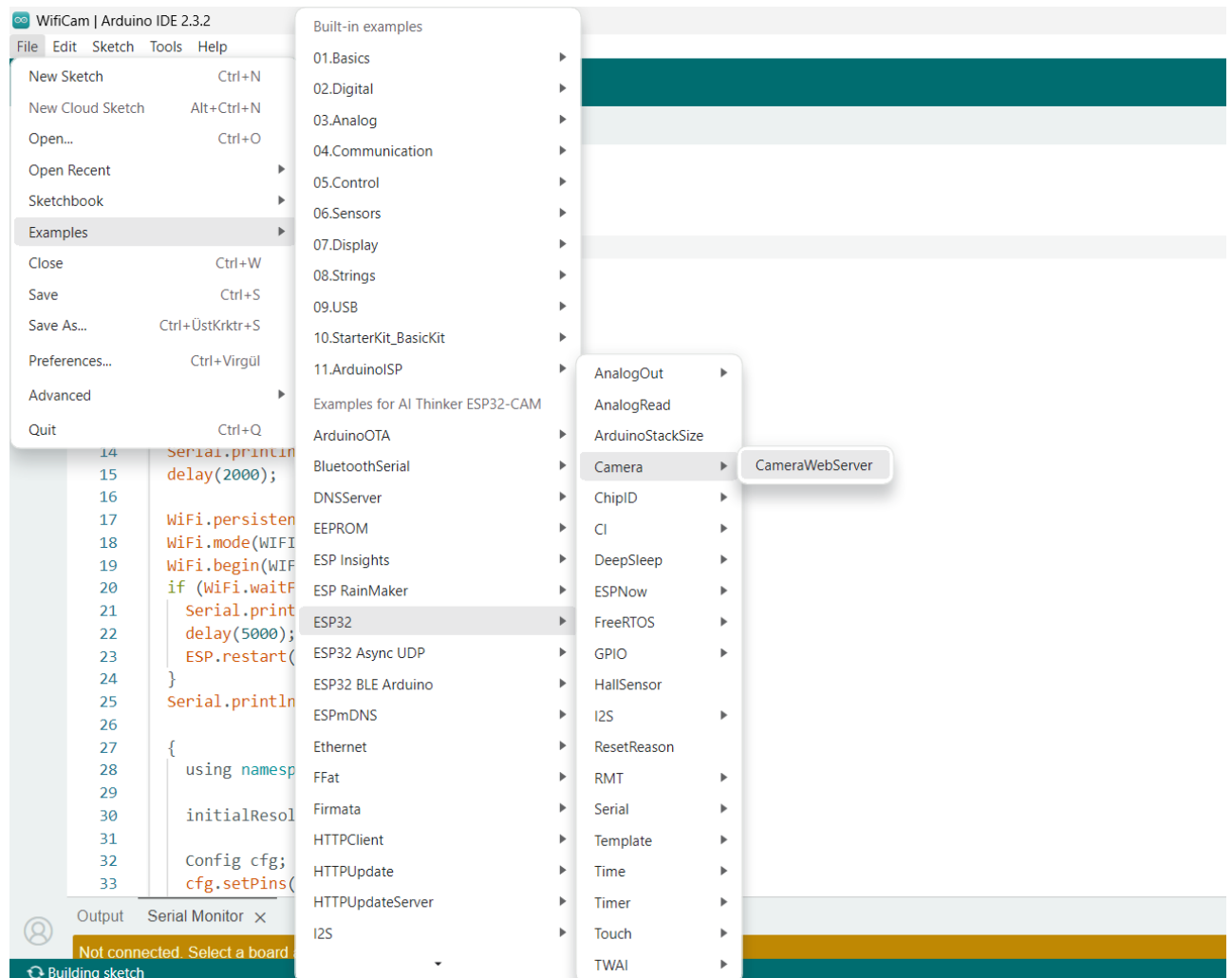
iv) CMake: “CMake, derleyiciden bağımsız olarak, yazılım inşası otomasyonu, testi, paketleme ve yüklenmesi için kullanılan çapraz-platform ve açık kaynak kodlu yazılımdır. Çoklu kütüphane kullanan uygulamaları ve dizin hiyerarşisini destekler. CMake bir inşa yazılımı (build system) değildir, kodu build etmez, onun yerine yerli inşa sistemlerinin ihtiyacı olan şeyi, inşa betiklerini (build script) oluşturur.”⁽⁴⁾ projenin olmazsa olmazı CMake görüntü işleme için olmazsa olmazdır ve bu projede kullanacağımız OpenCV’yi kullanabilmemizi sağlar.

4) Projenin Çalışma Mantığı

Girişte bahsettiğim üzere bu proje wi-fi özellikli esp32’nin yayınından alınan görüntü üzerinde yüz tanıma işlemi yapılması ve bu işlemin sonucunda elde edilen konum verisinin Arduino UNO’ya gönderilerek tekrar işlenmesi ve sonucunda 180° dönebilen servo motorun kontrol edilmesine dayanmaktadır. Biri Haar cascade diğeri ise YoloV8 kullanan iki ayrı yüz tanıma sistemi kullanılmıştır. Haar cascade ile OpenCV’de bulunan genel bir yüz tanınması yapılmakta ve YoloV8 ile kendi eğittiğim spesifik olarak benim yüzümü tanıyan bir model oluşturulmuştur. İki modelin aynı anda kullanılması korkutucu gelebilir ama bir “if” satırı içine Haar cascade’in yüz

tespiti yapan modelini koyduğumuz zaman hiçbir problem kalmayacaktır.

a)Esp32 Yayını: Esp32 ile bir yayın başlatmak oldukça basit Arduino IDE kullanarak zaten hazır olan örneklerden faydalanabilirsiniz.IDE üzerinde aşağıda gösterdiğim işlemi yaparak yayın kodunu alabilirsiniz ve Esp32'ye bu kodu yükleyerek kullanabilirsiniz.

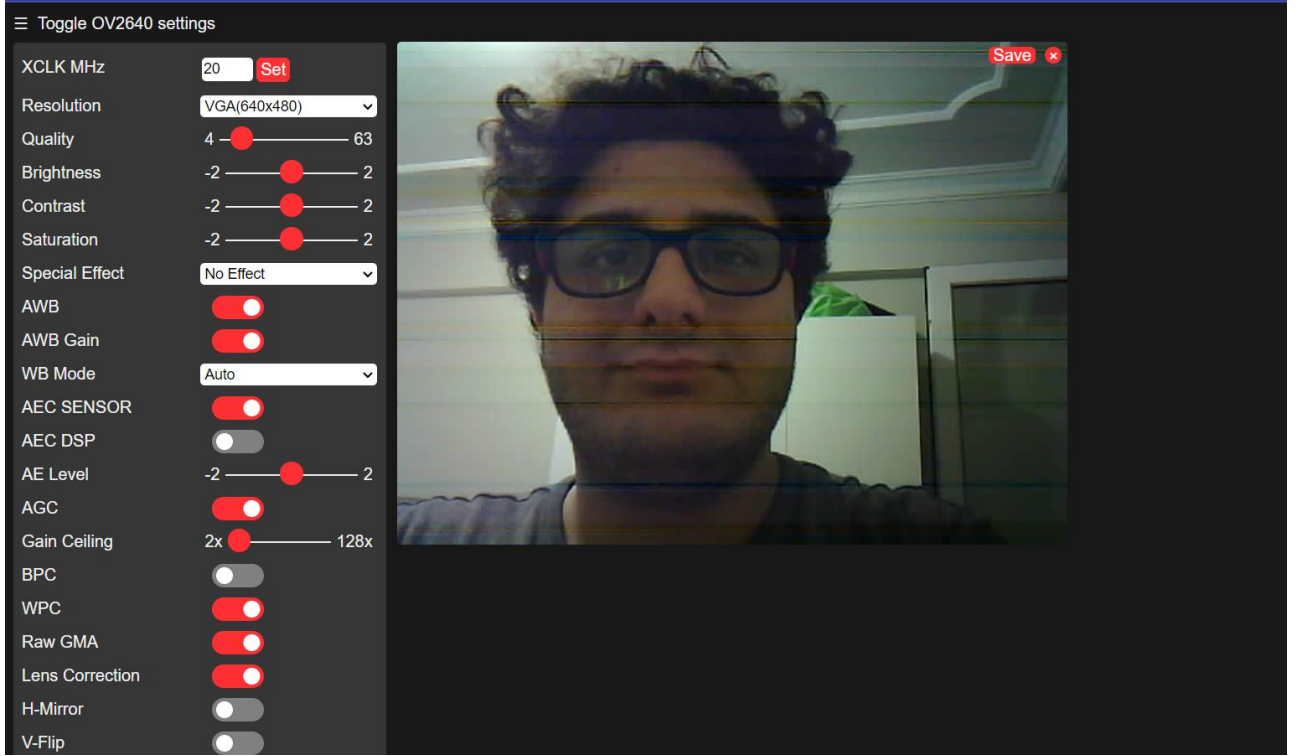


Bu koddaki yapmanız gereken tek değişiklik aşağıda görünen “ssid” bölümüne kullanacağınız wifi ismini ve “password” bölümüne wifi şifrenizi yazmalısınız ardından çalıştırdığınızda wifi bağlantısı kuracak ve şu şekilde bir çıktı verecektir.

.....
WiFi connected
Camera Stream Ready! Go to: <http://192.168.43.237/>

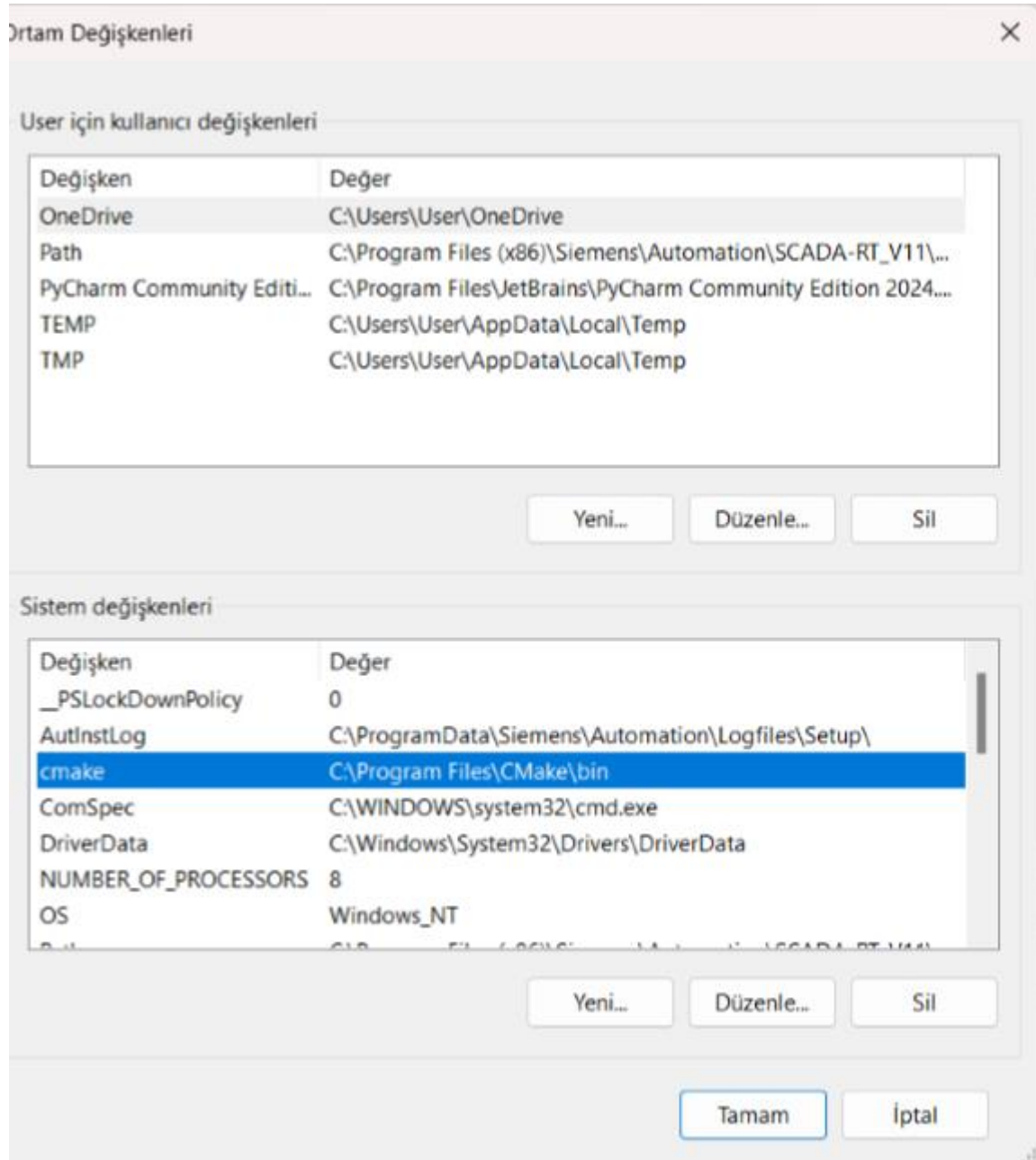
Verdiği bu linki kopyalayarak yayının yapıldığı sekmeyi açabilirsiniz.

Açılan site bu şekilde görünmeli.



Bu görüntüde çözünürlük kontrast gibi değişimler yapabilir,veya direk projeye devam etmek için yüz tanıma yapmaya başlayabiliriz.

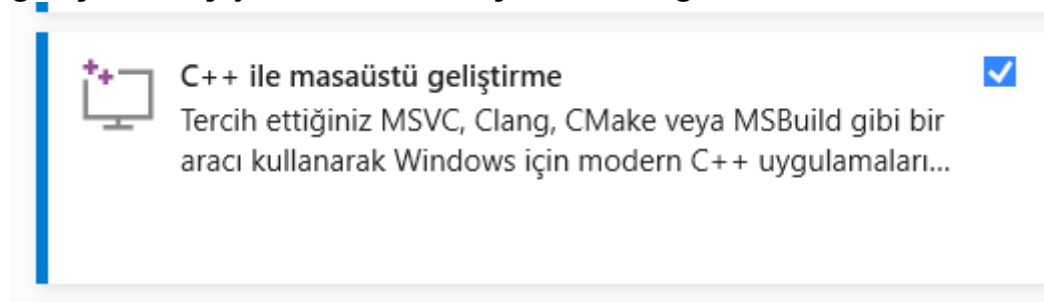
b)Yüz Tanıma Gereklilikleri:Yüz tanımadan önce bilgisayarınızda CMake'in bulunması ve ortam değişkenlerine eklenmiş olması gerekiyor.



Hemen ardından Dlib kütüphanesinin yüklü olduğundan emin olmalı ve değilse yüklemelisiniz. Aşağıdaki komutu terminalde kullanarak hızlıca Dlib'i yükleyebilirsiniz

```
> pip install dlib
```

Ardından Microsoft Visual Studio uygulamasında C++ ile masaüstü geliştirme iş yükünü indirmiş olmanız gerekmekte.



Hepsini tamamladıktan sonra artık python IDE'lerinden birinin terminalinde

```
pip install opencv-python numpy torch pyautogui
```

Komutu ile ihtiyacımız olan OpenCV NumPy PyTorch ve PyAutoGUI kütüphanelerini indirebiliriz. Kısaca bu kütüphanelerden bahsetmek gerekirse :

OpenCV: Açık kaynak kodlu bir kütüphanedir ve görüntü işleme, gerçek zamanlı hareket tanımlama, hareket algılama, ve bizim asıl ihtiyacımız olan yüz tanıma için kullanılır.

NumPy: Uzun hali Numerical Python olan bu kütüphane yaptığımız matematiksel işlemlerin daha hızlı yapılmasını sağlamakla kalmaz aynı zamanda yaptığımız matematiksel işlemleri görselleştirmemize, çok boyutlu matrisleri kullanmamıza olanak sağlar.

PyTorch: Torch kütüphanesini kullanan açık kaynak kodlu bir makine öğrenmesi kütüphanesidir. Bu kütüphane kullanarak bilgisayarla görme ve doğal dil işleme gibi modeller yapılabilir. Facebook tarafından geliştirilmiştir.

PyAutoGUI: İşletim sisteminde fare hareketleri, klavye komutları ve ekran görüntüleri gibi işlemlerini gerçekleştirmek için kullanılır uygulamamızda ekranın belirli bir kısmını keserek oranın ekran görüntüsünü almak için kullanacağız.

c) Haar cascade ile hazır model: Öncelikle basit olan hazır Haar cascade modelini kullanarak başlayalım bu sayede kodumuz hakkında da bilgi sahibi olur ve diğer aşamalarda ne yaptığımızı daha kolay bir şekilde anlayabiliriz. Bu aşamada göstereceğim kod sadece Haar cascade kullanarak ekranın belirli bir kısmında yüz olup olmadığını kontrol edecek.

Adım adım ilerlersek önce kullanacağımız yüz tanıma modelini seçeceğiz ben Haar cascade kullanarak yazdım.

```
import cv2
import pyautogui
import numpy as np

# Haar cascade ile yüz algılayıcı
face_casc = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
```

Ardından 3 adet fonksiyon tanımlayarak yüz tanıma yapmak istediğimiz alanı, ardından yüz tanıma işlemi için Haar cascade kullanımı ve son olarak yüzlerin etrafına bounding box çizme işleminin yapılması.

```
# Ekran görüntüsünü al ve belirli bir alanı kes
def Ekran_goruntusu(x, y, width, height):
    screenshot = pyautogui.screenshot(region=(x, y, width, height))
    frame = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2BGR)
    return frame

# Yüz tanıma işlemi
def Yuz_tani(frame):
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_casc.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    return faces

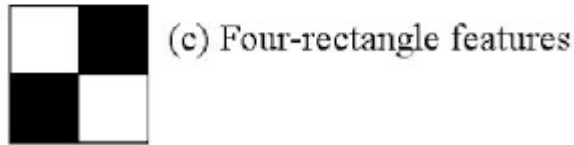
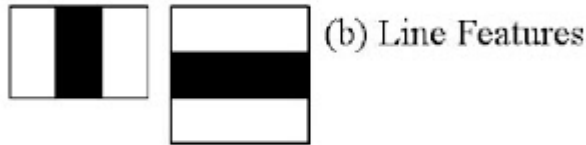
# Yüzlerin etrafına dikdörtgen çiz ve konumlarını yazdır
def Bounding_box(frame, faces):
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        # Yüzün konumunu yazdır
        print("\0 "f"{x} {y} {w} {h}")
    return frame
```

Ekran_goruntusu: İsminden anlaşılacağı üzere 4 değişken istediğimiz ve karşılığında PyAutoGUI kullanarak ekranın belirli bir kısmının görüntüsünü alıp döndürdüğümüz bir fonksiyon.

Yuz_tani: Yüz tanıması yaptığımız bu fonksiyon ekrandaki görüntünün gri tonlamasına çevrilmesini ve bu sayede Haar cascade'in verimli çalışmasını sağlar. Tabii bu noktada Haar cascade'in nasıl çalıştığından bahsetmemiz gerekir.

Haar Cascade: Haar cascade tonlamacıları gri tonlamaları üzerinde çalışarak bize bir ayırım işlemi yapar peki bunu nasıl yapar?

Bir sürü pozitif(nesnenin bulunduğu) ve negatif(nesnenin bulunmadığı) fotoğraf kullanılarak eğitilir ve hat ayırımı yapılır. Her hat, beyaz dikdörtgenin altındaki piksellerin toplamının siyah dikdörtgenin altındaki piksellerin toplamından çıkarılmasıyla elde edilen tek bir değerdir.



Böyle bir uygulama 24x24'lük bir ekran için bile 160000 farklı hatta sahip olabilir bu yüzden integrasyon işlemi kullanır ve inanılmaz bir hız farkı yaratır.Ardından yaptığı yüz tanıma işlemlerinde bir sınır değeri belirleyerek en optimize modeli karşımıza çıkarmaya uğraşır.

Bounding_box:Bounding box kelime anlamı olarak bağlayıcı kutu anlamına gelmekte ve cv2(OpenCV) kütüphanesi kullanarak belirttiğimiz konuma bir dikdörtgen çizmemizi sağlar,belirttiğimiz konumlarıda Yuz_tani ve Ekran_goruntusu fonksiyonlarının döndürdüğü "frame" ve "face" değişkenlerinden alır.

```

def main():
    # Ekran görüntüsünün kesileceği alanı belirle
    x, y, width, height = 100, 100, 900, 900

    while True:
        # Ekran görüntüsünü al ve belirli bir alanı kes
        frame = Ekran_goruntusu(x, y, width, height)
        # Yüz tanıma işlemi yap
        faces = Yuz_tani(frame)
        # Yüzlerin etrafına dikdörtgen çiz ve konumlarını yazdır
        Cikti = Bounding_box(frame, faces)
        # Görüntüyü göster
        cv2.imshow('Screen Face Recognition', Cikti)
        # 'q' tuşuna basıldığında döngüyü kır
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

```

Artık main fonksiyonuna geldik herkesin bildiği gibi main fonksiyonu işlemlerin yapıldığı fonksiyondur. Burada önce ekran üzerinde almak istediğimiz kesit için x,y,genişlik ve yükseklik verilerini giriyoruz ve bir while döngüsü içinde “Ekran_goruntusu” fonksiyonunu bu değerleri kullanarak çağırıyoruz ve bunun sonucunu “frame” değişkenine atıyoruz. Ardından “Yuz_tani” fonksiyonunu “frame” ile çağırıyoruz bu sayede yalnızca “frame” içinde yüz tanımasını sağlıyoruz ve bu çıktıyı “faces” değişkenine atıyoruz. Son olarak “Bounding_box” fonksiyonunu çağırıyor ve “frame” değişkeninin kapsamı içerisindeki “faces” değişkenlerinin etrafına bir bounding box çizmesini sağlıyor ve bunların konumlarını yazdırıyoruz burada “Cikti” değişkenine aktarıyoruz.

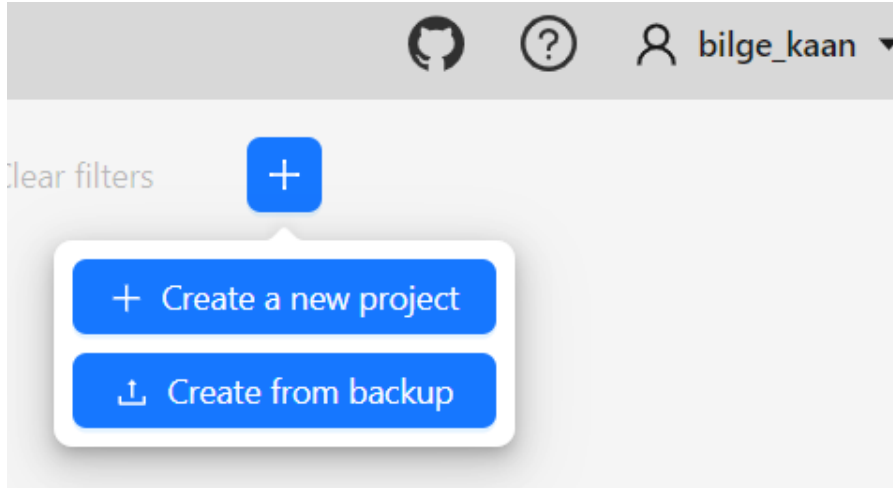
Artık görüntümüzü cv2.imshow komutu ile gösterebiliriz bu komut OpenCV’nin yeni bir pencere açmasını sağlar ‘Screen Face Recognition’ pencerenin ismi, Cikti ise bu pencerede gösterilecek NumPy dizisinin ismidir. Son olarak cv2.waitKey(1) ile 1ms bekleme yapılır ve bu arada 0 ile FF hexadecimal değerleri arasında bir işlem yapar ve ‘q’ tuşuna basılıp basılmadığını kontrol eder ‘q’ tuşunun ASCII kodu karşılığı 113’tür bu yüzden basıldığı anda break komutu ile while true döngüsünden çıkarır ve işlemi sona erdirir. Bu kod çalıştırıldığı zaman aşağıdaki gibi bir çıktı verecektir.

d)Kendi YoloV8 modelinizi oluřturma:

Kendi YoloV8 modelimizi oluřturmak iin algılanmasını istediėim nesnenin fazlaca fotoėrafına sahip olmamız gerekmektedir.Bu fotoėraflar btnnden raporun kalanı boyunca veri seti olarak bahsedeceėim.eřitli ve zengin bir veri setine sahip olmak modelimizin genelleme yapabilme yeteneėini ciddi anlamda geliřtirecektir.Benim kullandığım veri setinde 344 adet farklı fotoėraf bulunmaktadır.Verit setimizi oluřturduktan sonra sırasıyla annotation,labeling ve training adımlarını takip edeceėiz.

Annotation kelimesinin bu alanda net bir Trke evirisi olmaması ile birlikte sanırım dipnot olarak kullanmak uygun olacaktır kendi oluřturacaėımız modelin aslında ėrenebilmesi iin gereken ilk adım.Ben annotation iin CVAT⁽⁵⁾ kullandım,bir yapay zeka ve local olarak alıřmanız gerekmediėi iin ıktılar dıřında hibir řey indirme mecburiyetiniz olmuyor,bu sayede diėer annotation aralarından ok daha kolay bir kullanım sunuyor.

ncelikle CVAT belirttiėim linke girip yeni bir proje oluřturuyoruz.



Create a new project

* Name

Face_Recognizer ✓

Labels:

Raw Constructor

Bilge ✓ Any Add an attribute

Continue Cancel

> Advanced configuration

Submit & Open Submit & Continue

Submit & Open diyerek projemizi oluřturuyoruz,lütfen tanıtacađınız nesne sayısı kadar label(etiket) oluřturun ki sonraki adımda kullanımı kolay olsun.řimdi bir task(görev) oluřturarak iřlemimize bařlayabiliriz

< Back to projects Actions

Face_Recognizer

Project #162229 created by bilge_kaan on July 20th 2024 Assigned to Select a user

Project description

Edit

Issue Tracker

Raw Constructor

Add label Setup skeleton From model

Search ... Sort by Quick filters Filter Clear filters

+ Create a new task + Create multi tasks

Select files bölümüne dosyalarımızı yükledikten sonra tekrar Submit & Open butonuna basarak görevimizi oluřturuyoruz.Bu iřlem biraz zaman alabilir.

Bilge_recognition ✓

Project

Face_Recognizer

Subset


Input subset

Labels

Project labels will be used

* Select files

My computer Connected file share Remote sources Cloud Storage



Click or drag files to this area

You can upload an archive with images, a video, or multiple images

7 files selected

> Advanced configuration

Submit & Open Submit & Continue

Açılan sekmedeki mavi ile görünen job butonuna basarak tanıma işlemini gerçekleştirmek istediğimiz nesneyi işaretleyebiliriz.

Job #1049671 🔗

Created on July 20th 2024 03:25
Last updated July 20th 2024 03:25

Assignee:

Stage:

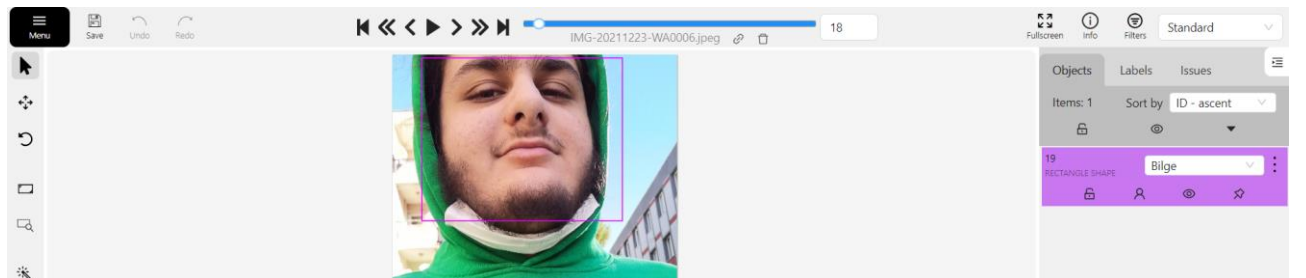
State:

Duration: a few seconds

☐ Frame count: 7 (100%)

☐ Frame range: 0-6

Yaklaşık olarak aşağıdaki gibi tanınmasını istediğiniz yüzün etrafını dikdörtgen içine almanız ve bunu veri setinizdeki her veri için tekrar etmeniz gerekmektedir.



Görebileceğiniz gibi zaten tek label ayarladığımız için dikdörtgen için aldığımızı hemen label ile bağlıyor.İşimiz bittiğinde sol üstteki

menu butonundan “export job dataset” butonuna basarak işlemimizde elde ettiğimiz verileri bilgisayarımıza aktarmalıyız. Karşımıza çıkan bu sekmede ise Export format olarak Yolo seçtiğimizden emin olalım çünkü farklı nesne tanıma algoritmaları farklı girdilere ihtiyaç duyabiliyor.

Export job #1041423 as a dataset

✕

* Export format

↓ YOLO 1.1

▼

☐ Save images

Custom name

Recognizer

.zip

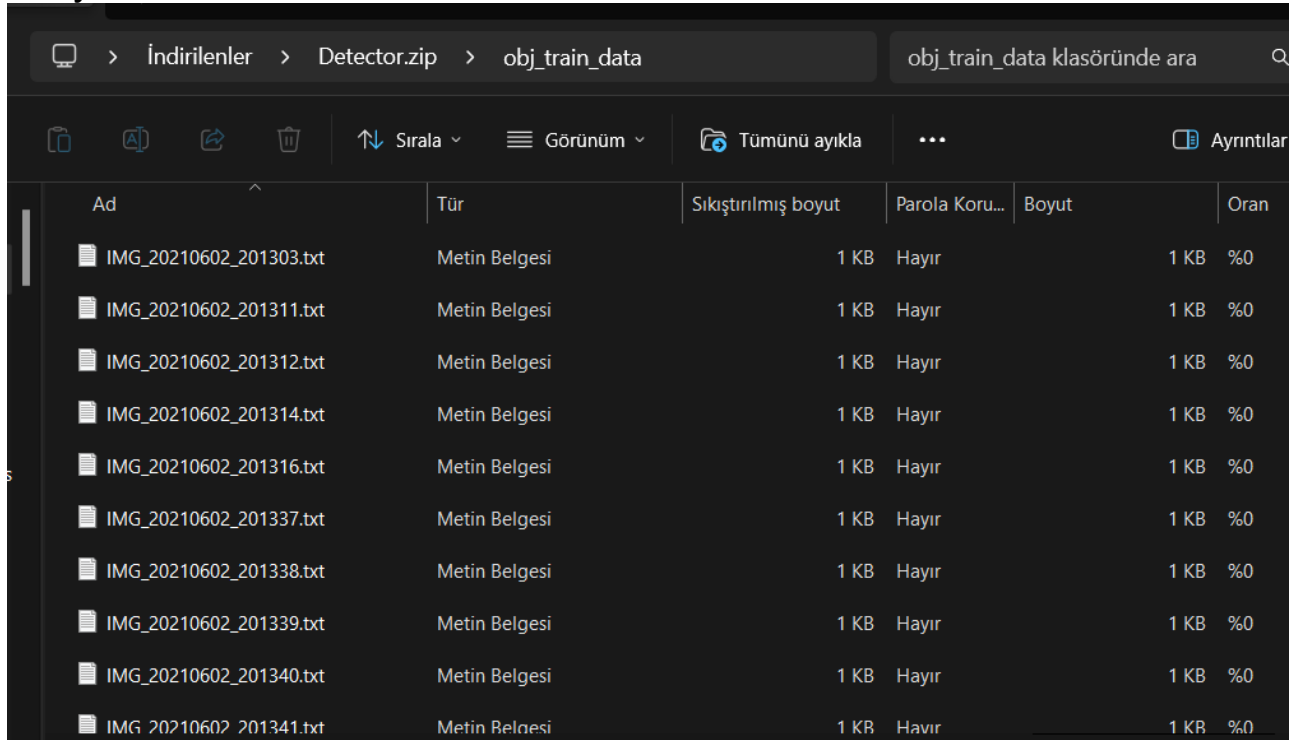
☒ Use default settings ?

Cancel

OK

İndirdiğimiz .zip dosyası içinde “obj_train_data” klasörünün içinde her fotoğrafınız için fotoğraflarınız ile aynı isim etiketinde bir .txt

dosyası olacaktır.

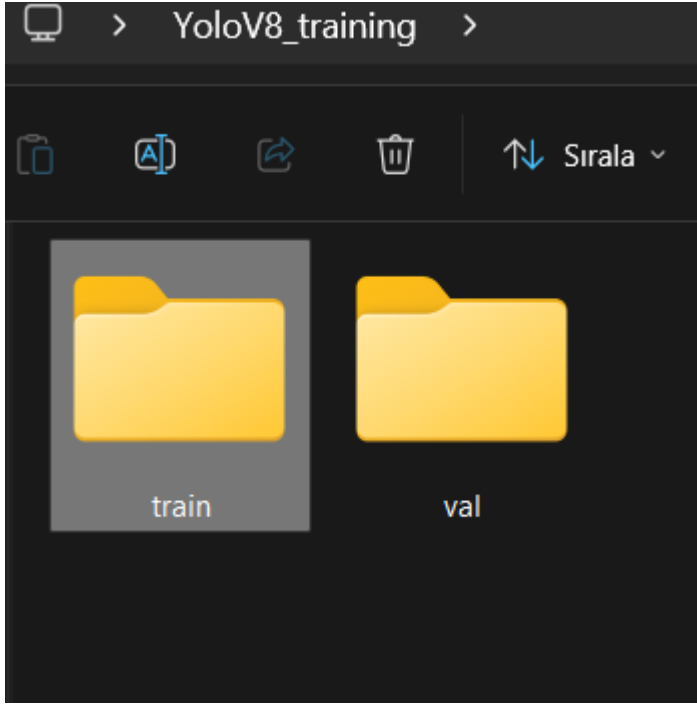


Ad	Tür	Sıkıştırılmış boyut	Parola Koruması	Boyut	Oran
IMG_20210602_201303.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201311.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201312.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201314.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201316.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201337.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201338.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201339.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201340.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0
IMG_20210602_201341.txt	Metin Belgesi	1 KB	Hayır	1 KB	%0

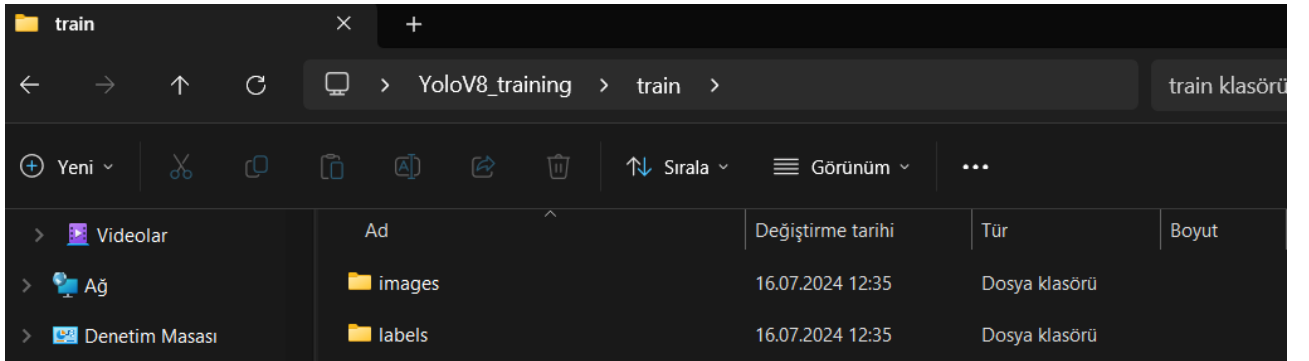
İçinde bu şekilde veriler bulunacaktır

```
0 0.696877 0.387838 0.191420 0.158415
```

Bu veriler nesnenizin etiketi, konumu, genişliği ve yüksekliğini belirtmektedir. Şimdi bu .txt dosyalarını fotoğraflar ile tek bir klasör altında toplamalıyız ama öncelikle verilerimizi %80 eğitim %20 doğrulama için ayırmamız önerilir bu yüzden eğitimi yapacağımız klasörün içine bir adet “train”, bir adet “val” adlı klasör açmamız gerekmektedir.



Şimdi bu klasörlerin her ikisinde içine ayrı ayrı 1 adet images,1 adet “labels” klasörü eklememiz gerekiyor.



Fotoğraflarımızı “images”,aynı isme sahip .txt dosyalarımız “labels” klasörünün içine koyacağız ve aynı işlemi “val” klasörü içinde gerçekleştireceğiz.Tüm bu ayarlamaları yaptıktan sonra artık projemiz içinde “config.yaml” isimli bir dosya oluşturacağız.Oluşturduğumuz dosyamız yaklaşık olarak bu şekilde görülmeli.

```
path: "C:/Users/User/Desktop/YoloV8_training"
train: train
val: val

nc: 1
names:
  0: 'Bilge'
```

Path: Eğitim için kullanacağımız dosyaların klasörlerinin konumu.

Train: Eğitim için kullanacağımız dosyalarda eğitim dosyalarının “path” yoluna görece konumu.

val: Eğitim için kullanacağımız dosyalarda doğrulama dosyalarının “path” yoluna görece konumu.

Nc: “Number of classes” yani sınıf sayısı annotation yaparken kullandığımız farklı label sayısı kadardır, bu modelde tek bir yüzü tanıtmak istediğim için 1 yazıyorum.

Names: Altında gördüğünüz “0” 0. Sınıfın ismini tanımlamak için kullanılmakta.nc değerine 1 dediğimiz için 1 adet sınıf ve index değerleri 0’dan başladığı için 0. Sınıfı ismi ‘Bilge’ olarak verilmiş.

Bunları yaptığımıza göre artık eğitim kısmına geçebiliriz.

`pip install ultralytics` kodunu terminalde çalıştırarak “ultralytics” kütüphanesini indirebilirsiniz.Ardından bu kodu kullanarak eğitime başlayabilirsiniz.

```
from ultralytics import YOLO

model = YOLO("yolov8m.yaml") # Kendi modelini üret

# Eğitim parametreleri
model.train(data="C:/Users/User/Desktop/haar_cascade/haar_cascade/config.yaml", epochs=50, batch=16)
```

Öncelikle “ultralytics” kütüphanesinden “YOLO”yu import ediyoruz.YOLO(“yolov8m.yaml”) fonksiyonunda bilmemiz gerekiyor ki eğitim için tek model değil farklı modeller kullanabiliriz ve model seçimi işte bu ‘m’ harfinde saklı bunun yerine ‘n’ ‘s’ ‘l’ gibi değişiklikler yaparak model seçimi yapabiliriz peki bu modeller ne işe yarar ve nasıl fark yaratır?

Performans						
Algılama (COCO)	Algılama (Open Images V7)	Segmentasyon (COCO)	Sınıflandırma (ImageNet)	Pose (COCO)	OBB (DOTAv1)	
Önceden eğitilmiş 80 sınıf içeren COCO üzerinde eğitilen bu modellerle kullanım örnekleri için Algılama Dokümanları'na bakın.						
Model	boyut (piksel)	mAPval 50-95	Hız CPU ONNX (ms)	Hız A100 TensorRT (ms)	params (M)	FLOP'lar (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Yukarıdaki örnekte de görülebileceği gibi büyük bir hız farkı ama aynı zamanda da algılama farkı bulunmakta bu konuda yapılabilecek tek yorum daha az karmaşık veri setleri ile çalışılırken daha hızlı olan “YOLOv8n” modeli ama daha tutarlı tahminler için daha güçlü modeller kullanılabilir.

```
# Eğitim parametreleri
model.train(data="C:/Users/User/Desktop/haar_cascade/haar_cascade/config.yaml", epochs=50, batch=16)
```

Yukarıdaki eğitim paramterelerini teker teker açıklamak daha uygun olacaktır.

Data: Verilerimizin bulunduğu .yaml dosyasının konumudur.

Epochs: Bir epoch, tüm eğitim veri setinin model tarafından tamamen kullanıldığı bir döngüdür. Bu, modelin tüm eğitim verilerini bir kez görmesi ve öğrenmesi anlamına gelir. Epoch seçimi model için en önemli değişken olabilir çünkü modelin öğrenme miktarını doğrudan etkileyecektir, yüksek epoch seçimi overfitting(aşırı uydurma) sebep olabilir, düşük epoch seçimide modelin tahmin yapmamasına sebep olabilir.

Batch: Modelimizin tek bir seferde ne kadar veri ile işlem yapacağını belirler düşük yapmak modelimizin daha dikkatli ve düşük bir sistem gücü ile işlem yapmasını sağlarken yüksek seçmek daha yüksek bir bellek kapasitesi kullanıp daha hızlı yapmasına sebep olur ama dikkatli olunmalı gerekenden yüksek bir batch kullanımı modelin öğrenmesini olumsuz etkileyebilir.

Ben eğitim yaparken fazla parametre kullanmadım ama size kullanabileceğiniz ek parametrelerden bahsetmek isterim.

lr0: İlk epochtaki öğrenme miktarıdır ve bu öğrenme değeri “lr”ye yaklaştıkça azalır.

lr_f: Son epochtaki öğrenme oranıdır.

imgsz: Kullanılacak fotoğrafların boyutudur. Default olarak 640 verilmiştir yani veri setinizdeki fotoğrafları 640x640 olacak şekilde genişletir veya daraltır ama fotoğrafların en boy oranı ile oynamak yerine öğren 4:3 olan bir fotoğrafın kısa kenarını gri bölümle tamamlar ve bu bölümü eğitime dahil etmez.

Yaklaşık bir eğitim epochunda aşağıda olduğu gibi verilerin çıktısı terminalde görülecektir.

	Epoch	GPU_mem	box_loss	cls_loss	df_l_loss	Instances
Size	47/50	0G	1.22	0.903	1.827	2
640: 100%	18/18 [22:45<00:00, 75.83s/it]					
	Class	Images	Instances	Box(P	R	
mAP50	mAP50-95): 100%	3/3 [01:26<00:00, 28.99s/it]				
all	66	66	0.847	0.682	0.759	0.462

Genel olarak bir eğitimin iyi gidip gitmediğini görmek için “loss” değerleri kontrol edilebilir, problemsiz bir eğitim sürecinde “loss” değerlerinde düşüş eğilimi görülecektir. Bu aşamanın sonuna geldikimize ve bir model eğittiğimize göre bu modeli kullanarak yüz tanıma yapalım.

5)Projenin Kodları

a)Python Kodları

Bu aşamaya kadar projemizi gerçekleştirmek için ihtiyacımız olan her şeyden bahsettik artık projemizin son adımını atarak işimizi tamamlamalıyız.Esp32 ile nasıl yayın yapıldığından bahsettiğim ve örnek kodları kullandığım için bu adımda kodlarına yer vermeyeceğim,Python kodları ile başlamanın daha uygun olacağını düşünüyorum.

Zaten daha önce OpenCV,PyAutoGUI,NumPy,ultralytics kütüphanelerinden bahsetmiştik bu adımda pip install ile kurmamız gereken kütüphane serial ve aşağıdaki kodu terminale girerek kurabilirsiniz.

```
In [1]: pip install pyserial
```

“Time” ve “sys” kütüphaneleri zaten default olarak Python IDE’lerinde bulunmakta onlar için herhangi bir işlem yapmamıza gerek yok.

```
1 import cv2
2 import pyautogui
3 import numpy as np
4 from ultralytics import YOLO
5 import serial
6 import time
7 import sys
8
9 # YOLOv8 modelini yükle
10 model = YOLO("C:/Users/User/.spyder-py3/runs/detect/train9/weights/best.pt")
11
12 # Haar cascade ile yüz algılayıcı
13 face_casc = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_default.xml')
14
```

Model değişkeni içine eğittiğimiz modelin konumunu girerek çağırıyoruz ve “best.pt” doğruluk ve kayıplar açısından eğittiğimiz en iyi ağırlıkları içermekte bu yüzden bu dosyayı kullanıyoruz.Face_casc değişkeni ise Haar cascade default modelini içermekte.

```
# Arduino ile seri iletişim kurmak için
try:
    seri = serial.Serial('COM4', 9600) # 'COM4' yerine Arduino'nun bağlı olduğu portu yazın
    time.sleep(2) # Arduino'nun resetlenmesi için kısa bir bekleme
except serial.SerialException as e:
    print(f"Seri port hatası: {e}")
    sys.exit()
```

Try komutu ile hata yaklama bloğumuzun başlangıcını yapıyoruz,”seri“ değişkeninin içine seri bağlantı için ihtiyacımız olan

port ve baud rate değerlerini giriyoruz. Seri port iletişimde baud rate saniye içinde sinyal değişim miktarıdır ve hem Arduino hemde kodumuzda aynı değerde olması önemlidir. Ardından “sleep” komutu ile Arduino’nun resetlenmesini bekliyoruz. “serial.SerialException” hatasının ortaya çıkması durumunda ‘except’ bloğu çalışacak ve bize hata mesajını yazdıracaktır. Son olarak “sys.exit” bizi programdan çıkaracaktır.

```
# Ekran görüntüsünü al ve belirli bir alanı kes
def Ekran_goruntusu(x, y, width, height):
    screenshot = pyautogui.screenshot(region=(x, y, width, height))
    frame = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2BGR)
    return frame
```

Burada tanımladığımızı fonksiyon ‘x’, ‘y’, ‘genişlik’, ‘yükseklik’ bilgilerini alarak bize ekranımızın istediğimiz bölgesinin görüntüsünü alıyor ve ‘frame’ değişkeninin içinde OpenCV tarafından kullanılacak bir hale getiriyor.

```
# YOLOv8 ile yüz tanıma işlemi
def Yolo_tani(frame, conf_threshold=0.7):
    results = model(frame)
    faces = results[0].boxes.xyxy.cpu().numpy()
    confidences = results[0].boxes.conf.cpu().numpy()

    # Threshold değerinin altındaki tahminleri filtrele
    taninan_yuz = []
    for i, conf in enumerate(confidences):
        if conf >= conf_threshold:
            taninan_yuz.append(faces[i])

    return np.array(taninan_yuz)
```

Burada hemen önceki fonksiyonumuzda döndürdüğümüz ‘frame’ değişkenini ve 0 ile 1 arasında seçtiğimiz confidence(eminlik) değerini kullanarak önce ‘results’ değişkeni ile yüz tanıma gerçekleştirilmesi ardından ‘faces’ değişkenine konumlarının ‘confidences’ değişkeninin confidence değerlerinin NumPy dizisi olarak tanımlanmasını içerir. Altındaki döngü ile seçtiğimiz confidence değerinin üzerindeki tanımları ‘taninan_yuz’ listesine ekler, bu listeyi NumPy dizisi olarak döndürür.


```
# Haar cascade ile yüz tanıma işlemi
def Haar_tani(frame):
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    yuz = face_casc.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    return yuz
```

Sadece frame değişkeni ile tanımlanan bu fonksiyonda önceden bahsettiğimiz gibi Haar cascade kullanım kolaylığı için gri tonlamasına dönüştürülür. En az 30x30luk yüzleri tespit eder.

```
# Yüzlerin etrafına dikdörtgen çiz ve konumlarını yazdır
def Bounding_box(frame, faces, is_yolo=True):
    for det in faces:
        if is_yolo:
            x1, y1, x2, y2 = det[:4]
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (255, 0, 0), 2) # Lacivert dikdörtgen
            # Yüzün konumunu yazdır
            print(f"0 {int(x1)} {int(y1)} {int(x2-x1)} {int(y2-y1)}")
        else:
            x, y, w, h = det
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2) # Yeşil dikdörtgen
            # Yüzün konumunu yazdır
            print(f"0 {x} {y} {w} {h}")
    return frame
```

Öncelikle Yolo modelinin yüz algılayıp algılamadığına bakar ve algılanan yüzü algılayan kendi eğittiğimiz model ise etrafına lacivert bir dikdörtgen çizer, algılayan Haar cascade ise yeşil bir dikdörtgen çizer.

```
# Motor kontrol fonksiyonu
def control_motor(x_center):
    # Arduino'ya yüzün x merkez koordinatını gönder
    try:
        seri.write(f"{x_center}\n".encode())
        print(f"Arduino'ya gönderildi': {x_center}")
    except serial.SerialException as e:
        print(f"Seri port yazma hatası: {e}")
```

Arduino'ya 'x_center' değişkenindeki bilgiyi gönderir ve bu konuda geri bildirim verir, Seri port yazma hatası ile karşılaşılır ise terminalde çıktı olarak görünür.

```
def main():
    # Ekran görüntüsünün kesileceği alanı belirle
    x, y, width, height = 400, 100, 900, 900
    screen_center = width / 2

    while True:
        # Ekran görüntüsünü al ve belirli bir alanı kes
        frame = Ekran_goruntusu(x, y, width, height)

        # YOLOv8 ile yüz tanıma işlemi yap
        faces = Yolo_tani(frame, conf_threshold=0.3)

        if len(faces) == 0:
            # Eğer YOLOv8 ile yüz algılanmazsa Haar cascade ile yüz tanıma işlemi yap
            faces = Haar_tani(frame)
            Kareli_goruntu = Bounding_box(frame, faces, is_yolo=False)
        else:
            Kareli_goruntu = Bounding_box(frame, faces, is_yolo=True)
```

'x','y','width','height' değişkenlerinin tanımlanması ve 'screen_center' değişkeninin genişliğin yarısına eşit olduğunun belirtilmesi. Ardından "while True" yani biz istemediğimiz sürece sonlanmayan bir döngü oluşturulması. 'Ekran_goruntusu' fonksiyonunun çıktısının değeri 'Main' fonksiyonu içinde tanınan 'frame' değişkeni olarak atanmıştır, 'Yolo_tani' fonksiyonunun çıktısı da yine 'Main' içinde tanımlanan 'faces' değişkenine atanmıştır. Yolo modeli kullanılarak tanınan yüz miktarının 0 olması durumunda Haar cascade kullanılarak yüz tanıma gerçekleştirilmiştir. Eğer tanınan yüz değeri 0'dan farklı ise Yolo ile yüz tanıma işlemine devam edilmektedir.

```
# Yüzlerin merkezini hesapla ve motoru kontrol et
if len(faces) == 1:
    if isinstance(faces[0], np.ndarray):
        x1, y1, x2, y2 = faces[0][:4]
        x_center = int(x1 + (x2 - x1) / 2)
    else:
        x, y, w, h = faces[0]
        x_center = x + w // 2
    kontrol_motor(x_center)

# Görüntüyü göster
cv2.imshow('Yüz tanıma', Kareli_goruntu)

# 'q' tuşuna basıldığında döngüyü kır
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Yüzler dizisinin eleman sayısının 1 olup olmadığını kontrol eder. 1 olmasının sebebi daha fazla yüz tespitinde motorumuzun bir sağa bir sola dönerek saçmalamasını önlemek. Ardından bu dizinin 0. Indexindeki eleman Yolo ile tespit edildi ise bir NumPy dizisi içinde olacaktır ve 'x1' yani yüz konumunun en solu ile 'x2' yüz genişliği bu iki veri ile yapılan matematiksel işlem ile yüzün orta noktasının konumunu buluyoruz. Eğer yüz tanıma işlemi Haar cascade ile yapılmışsa 'x' ve genişliğin yarısını toplayarak sonuca ulaşıyoruz, bu 'x_center' değişkeninide 'control_motor' fonksiyonu ile Arduino'ya gönderiyoruz cv2.imshow ile yüz tanıma penceresini açıyoruz ve q tuşuna basarak kapatabiliyoruz.

```
# Pencereyi kapat
cv2.destroyAllWindows()
seri.close() # Seri portu kapat

if __name__ == "__main__":
    main()
```

Döngüden çıkılmasının ardından tüm pencereler ve seri port kapatılıyor. Seri iletişim son buluyor.

b)Arduino Kodları

servoRotation.ino

```
1  #include <Servo.h>
2
3  Servo myservo;
4  int servoPin = 9;
5  int pos = 90; // Servo başlangıç pozisyonu (90 derece)
6  int screenWidth = 900; // Ekran genişliği
7  int centerX = screenWidth / 2;
8  int speed = 5; // Dönüş hızı
9
```

Öncelikle Servo kütüphanesini dahil ediyor ardından servomuzu tanımlıyoruz. Servomuzun 9. Pinde olduğunu, 90 derece açıda olmasını istediğimizi, ekran genişliğini 900, ekranın ortasının genişliğin yarısı olduğunu ve servo dönüş hızının 5 olmasını istediğimizi belirliyoruz.

```

10 void setup() {
11     Serial.begin(9600);
12     myservo.attach(servoPin);
13     myservo.write(pos); // Servoyu başlangıç pozisyonuna geti
14 }
15

```

Bu kodlar ile yukarıda belirlediğimiz değişkenleri gerçekleştiriyoruz ve 'Serial.begin' ile baud rate ayarı yapıyoruz.

```

void loop() {
    if (Serial.available() > 0) {
        String data = Serial.readStringUntil('\n');
        int x_center = data.toInt();

        // Ekran merkezine göre servonun dönüş yönünü belirle
        if (x_center < centerX) {
            pos += speed; // Sağa döndür
            if (pos > 180) pos = 180; // 180 dereceden fazla dönmemesi için sınır koy
        } else {
            pos -= speed; // Sola döndür
            if (pos < 0) pos = 0; // 0 dereceden az dönmemesi için sınır koy
        }

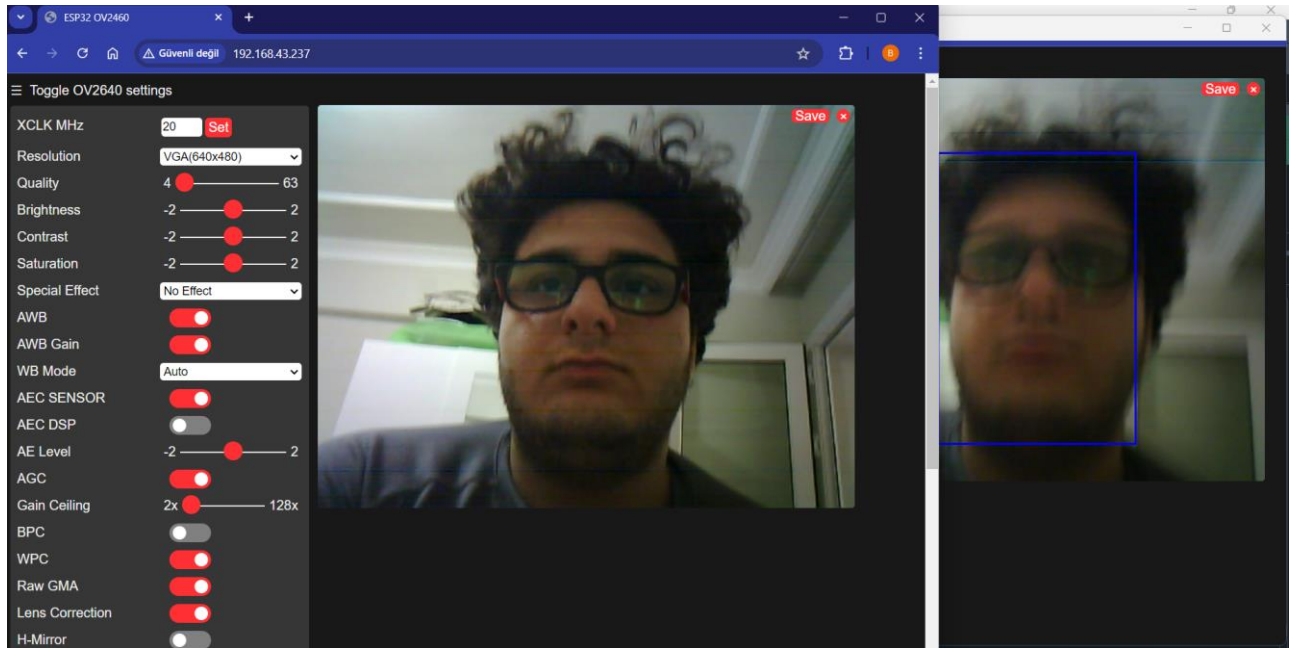
        myservo.write(pos); // Servo motoru yeni pozisyona getir
        delay(15); // Servonun hareketine zaman tanı
    }
}

```

Sonsuz bir döngü içinde önce seri portta veri olup olmadığını kontrol eder, varsa bir değişkene atar ve '\n' karakterine kadar gelen veriyi okur. Gelen veriyi 'x_center' olarak Python kodunda tanımlamıştık. Bu koddaki 'centerX' değeri ile karşılaştırır yani yüz ekranın ortasına göre nerede eğer ekranın ortasına göre solda ise motoru sağa, ekranın ortasına göre sağda ise sola döndürür. 'myservo.write' ile servonun açısını ayarlar ve 'delay' ile 15ms gecikme sağlayarak servo motorun hareketini tamamlamasına olanak sağlar.

6)Bütün Hatları ile Proje İncelemesi

Sonuç olarak düşük maliyet ile gerçek zamanlı yüz tanıma yapabilen bir kamera tasarladık ve bu kameramızın tanıdığı yüzü takip etmesini sağladık.Başarılı olduğunu düşündüğüm bu projede birkaç önemli noktanın altını çizmem gerektiğini düşünüyorum.Öncelikle Arduino kodunda benim verdiğimin dışında nötral bir alan belirlemeniz motorun daha stabil olması için daha uygun olacaktır,bu konum ekranın büyüklüğüne göre değişebileceği için koduma eklemedim bu halde de yüz algılama konusunda sıkıntı yaşatmayacak tutarlılıkta çalışıyor ama servonun daima hareket halinde olması tutarlılığı düşürmekte,yüzün ortasının ekranın merkezi ile olan mesafesi için 50 pixel bir tolerans değeri vermek mantıklı olacaktır.



Python kodunun çalıştırılabilmesi için farkettiğiniz üzere seri portla başka bir iletişim yapılmıyor olmalı yani Arduino IDE'de seri monitörü kapatmanız şart.Eğitim için kullandığım veri seti 344 adet fotoğraftan oluşmaktadır.80 epochluk eğitim ile mükemmel olmasada yeterli bir tutarlılığa sahip.

7)Kaynakça

- 1) <https://tr.wikipedia.org/wiki/ESP32>
- 2) <https://maker.robotistan.com/arduino-nedir-ne-ise-yarar>
- 3) <https://www.robishop.com/urun/tower-pro-sg90-rc-mini-9gr-servo-motor-1>
- 4) <https://tr.wikipedia.org/wiki/CMake>
- 5) <https://app.cvat.ai/projects>