BURSA ULUDAĞ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ

2022-2023 Eğitim Öğretim Yılı Bahar Dönemi

BMB2014 PYTHON PROGRAMLAMAYA GİRİŞ

Otonom Araç Projesi

Grup 10

Atilla Erdinç	032190098
Ayten Buse Atalay	032190088
Adnan Topçu	032190007
Muhammed Ali Gedikli	032190010
Batuhan Arslandaş	032190097

Otonom Sürüş Algoritmaları

Levha tanıma, şerit takibi ve sürüş algoritmalarını içeren bu projede levha tanıma algoritması ve şerit takibi algoritması yolov5 algoritması kullanılarak eğitilmiş yapay zeka modelleri kullanılarak hazırlanmıştır. Şerit takibi algoritması şeridin başlangıç noktasından bitiş noktasına doğru çizilen çizgi ile +y ekseni arasında yapılan açıyı seri haberleştirme yöntemi ile bir mikrodenetleyiciye iletme rolü taşımaktadır.

Etiketin Adı	Eğitim Verisi Sayısı	Doğrulama Verisi Sayısı
80kmh	180	20
Line	180	20

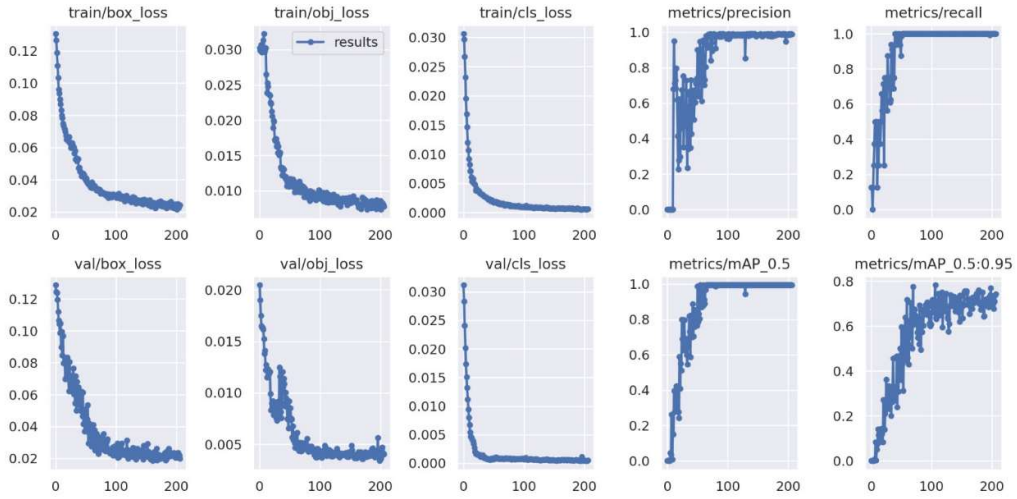
Tablo 1 – Yapay Zeka Modelinin Veri Seti Özellikleri

Yukarıdaki veri setiyle eğitilen yapay zeka modelinin veri seti özellikleri şu şekildedir;

Parametrenin Adı	Parametrenin Değeri
Epoch	430
Batch Size	64
Image Size	640

Tablo 2- Yapay Zeka Modelinin Eğitim Özellikleri

Yapay zeka modeli eğitiminin sonuçları aşağıdaki gibidir;

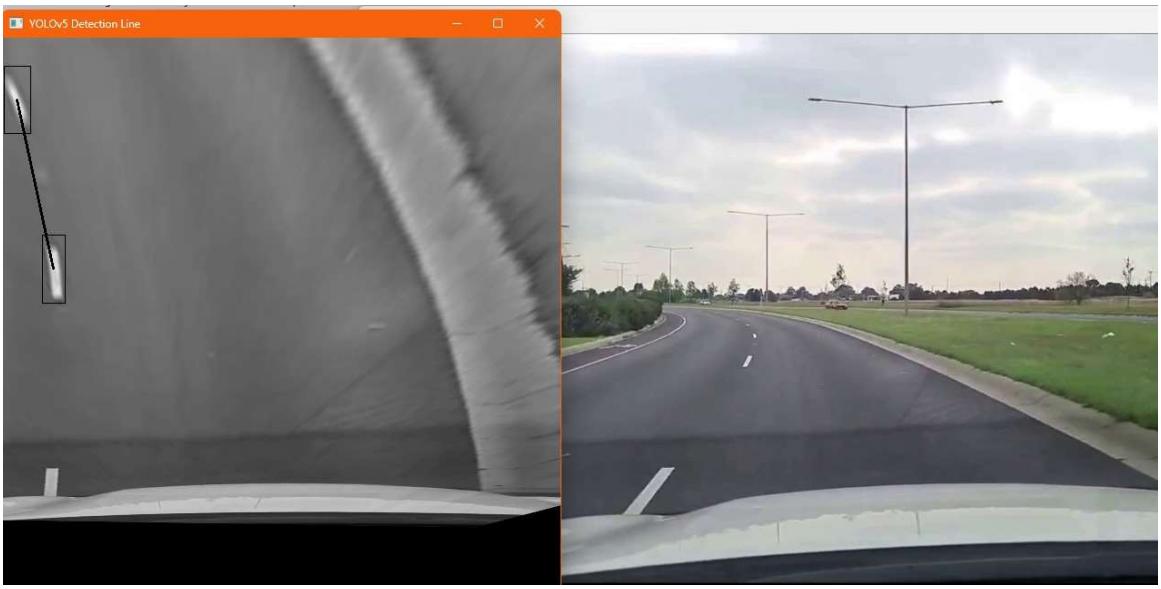


Şekil 1- Yapay Zeka Modelinin Eğitim Sonucu Grafikleri

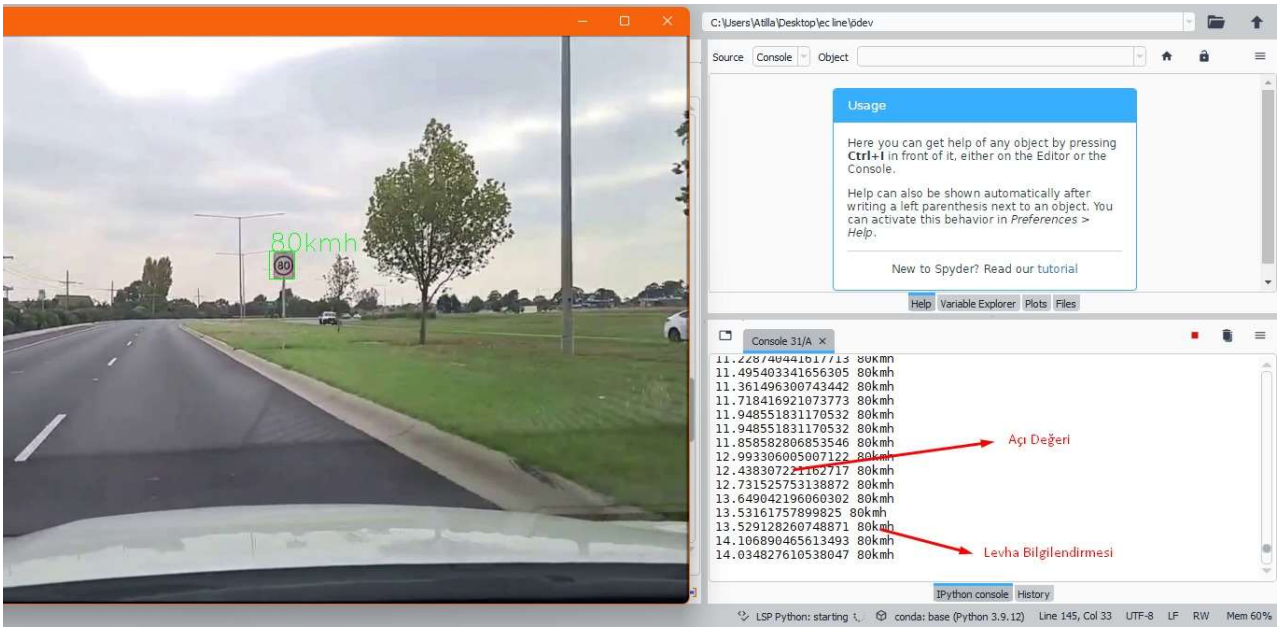
Yapay Zeka modelinin doğrulama sonuçları aşağıdaki gibidir;

Şekil 2. Yapay Zeka Modelinin Testi

Program İçinden Görüntüler



Şekil 3. Tespit Edilen İki Şerit Arasına Çizilen Çizgi



Şekil 4. Console Ekranındaki Bilgilendirme Satırları

Kaynak Kod:

```
import torch
import numpy as np
import cv2
from time import time
import os
import math
import serial


#arduino_port = 'COM3'  # Arduino'nun bağlı olduğu seri portun adı
#baud_rate = 9600  # Arduino'nun seri port hızı
#ser = serial.Serial(arduino_port, baud_rate)



class Detector:

    last_angle = 0
```

```
def _init_(self, capture_index, model_name):

    self.capture_index = capture_index
    self.model = self.load_model(model_name)
    self.classes = self.model.names
    self.device = 'cuda' if torch.cuda.is_available() else 'cpu'
    print("Using Device: ", self.device)

def get_video_capture(self):

    return cv2.VideoCapture(self.capture_index)

def load_model(self, model_name):

    model = torch.hub.load('ultralytics/yolov5', 'custom', path='odev.pt', force_reload=True)

    return model

def score_frame(self, frame):
    self.model.to(self.device)
    frame = [frame]
    results = self.model(frame)
    labels, cord = results.xyxyn[0][:, -1], results.xyxyn[0][:, :-1]
    return labels, cord

def class_to_label(self, x):

    return self.classes[int(x)]

def plot_boxes(self, results, frame, gray, image, control):
```

```
if control == 1:
    global last_angle
    angle_control = 0

    x_min = frame.shape[0]
    x_max = 0
    y_min = frame.shape[1]
    y_max = 0

    y_control_0 = 0
    y_control_1 = 0

    labels, cord = results

    n = len(labels)
    x_shape, y_shape = frame.shape[1], frame.shape[0]

    if(n>=2):
        x_1 = int(((cord[0][0]+cord[0][2])/2)*x_shape)
        x_2 = int(((cord[1][0]+cord[1][2])/2)*x_shape)
        y_1 = int(((cord[0][1]+cord[0][3])/2)*y_shape)
        y_2 = int(((cord[1][1]+cord[1][3])/2)*y_shape)

        if x_1 < frame.shape[1]/2:

            cv2.line(image, (x_1,y_1), (x_2,y_2), 0, 2)

        if(y_1<y_2):
            if(x_1<x_2):
```

```
                angle_control = -1
                #print("SOLA")
            else:
                angle_control = 1
                #print("SAĞA")
        else:
            if(x_1<x_2):
                angle_control = 1
                #print("sağa")
            else:
                angle_control = -1
                #print("sola")

    for i in range(n):
        row = cord[i]
        if row[4] >= 0.75:
            x1, y1, x2, y2 = int(row[0]*x_shape), int(row[1]*y_shape), int(row[2]*x_shape),
int(row[3]*y_shape)

            bgr = (0, 255, 0)

            if x1 < frame.shape[1]/2:

                if (x1+x2)/2 < x_min:
                    x_min = int((x1+x2)/2)

                if (y1+y2)/2 < y_min:
                    y_min = int((y1+y2)/2)

                if (x1+x2)/2 > x_max:
                    x_max = int((x1+x2)/2)
```

```
                if (y1+y2)/2 > y_max:
                    y_max = int((y1+y2)/2)

                cv2.rectangle(image, (x1, y1), (x2, y2), bgr, 1)

                gray_ = gray

                gray = gray.flatten()

                value = (np.bincount(gray).argmax())


        if(y_max - y_min != 0):
            #cv2.putText(image, str((math.atan((x_max-x_min)/(y_max-y_min)))*57.29) ,
(int(image.shape[0]/2), int(image.shape[1]/2)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, 0, 1)
            angle = (math.atan((x_max-x_min)/(y_max-y_min)))*57.29
            last_angle = angle
        else:
            angle = last_angle
        if(angle == 0):
            angle_value = (str(angle_control*last_angle))#+"burası") # 0 yazıyor
        else:
            angle_value = (str(angle_control*angle))#+"şurası")
        return image, angle_value

    else:

        name = ""
        labels, cord = results
```

```
        n = len(labels)
        x_shape, y_shape = frame.shape[1], frame.shape[0]



        for i in range(n):
            row = cord[i]
            if row[4] >= 0.5:
                x1, y1, x2, y2 = int(row[0]*x_shape), int(row[1]*y_shape), int(row[2]*x_shape),
int(row[3]*y_shape)


                bgr = (0, 255, 0)


                if self.class_to_label(labels[i]) != "line":
                    cv2.rectangle(image, (x1, y1), (x2, y2), bgr, 1)
                    cv2.putText(frame, self.class_to_label(labels[i]), (x1, y1), cv2.FONT_HERSHEY_SIMPLEX,
0.9, bgr, 1)
                    name = self.class_to_label(labels[i])


        return image,name




    def _call_(self):


        video = cv2.VideoCapture("drive3.mp4")


        while True:


            ret, frame = video.read()


            frame_sign = frame
```

```
kernel = np.ones((7, 7), np.uint8)

src = np.float32([[438, 327],
                  [566, 326],
                  [237, 522],
                  [874, 507]
                  ])

dst = np.float32([[0, 0],
                  [500, 0],
                  [0, 500],
                  [500, 500]])

matrix = cv2.getPerspectiveTransform(src, dst)

birdseye = cv2.warpPerspective(frame, matrix, (615,615))

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

frame = birdseye

results = self.score_frame(frame)
results_sign = self.score_frame(frame_sign)

frame, angle = self.plot_boxes(results, frame, gray, gray,1)
frame_sign, info = self.plot_boxes(results_sign, frame_sign, frame_sign, frame_sign,0)

if angle !="0.0" and info !="line":
   print(str(angle)+" "+str(info))
   #ser.write(str(angle)+":"+str(info))
```

```
            cv2.imshow('YOLOv5 Detection Line', frame)
            cv2.imshow('YOLOv5 Detection Sign', frame_sign)
            cv2.waitKey(1)

        cap.release()
        cv2.destroyAllWindows()



detector = Detector(capture_index=0, model_name='odev.pt')
detector()
```