TRAFİKTE YORGUNLUK TESPİT SİSTEMİ DOKÜMANI

Sema YEŞİLKAYA

26.04.2024

İçindekiler Tablosu

1 İlk Aşama: Gerekli Koşulların Belirlenmesi

1.1 Belirlenen Koşullar

- Sürücünün göz kapanma oranı
- Sürücünün esnemesi
- Sürücünün başının pozisyonu

1.2 Yöntemler

- Yüz tanıma: Cascade
  Cascade yüz tanıma yöntemi, nesne tanıma alanında oldukça yaygın olarak kullanılan bir yöntemdir. Temel olarak, bir yüzün farklı özelliklerini (gözler, burun, ağız vb.) tanımak için bir dizi önceden eğitilmiş sınıflandırıcıdan oluşur. Bu sınıflandırıcılar, yüzün farklı bölgelerini tarar ve yüzün varlığını veya yokluğunu belirlemeye çalışır. Cascade yöntemi, hızlı çalışma süresi ve yüksek doğruluk seviyeleri nedeniyle tercih edilir.

- Tanınan yüzden gözleri ayırt etme : Cascade
  Cascade yöntemi, yüz tanıma işlemi sırasında gözleri ayırt etmek için de kullanılır. Gözleri ayırt etmek için yine önceden eğitilmiş sınıflandırıcılar kullanılır. Bu sınıflandırıcılar, gözlerin varlığını veya yokluğunu tespit eder ve yüz tanıma sistemine daha fazla bilgi sağlar.

- Göz durumlarının sınıflandırılması : SVM+ HOG
  Destek vektör makinesi (Support Vector Machine,SVM), nesne tanıma ve sınıflandırma için kullanılır. Nesnelerin farklı bölümlerini (örneğin gözler, burun, ağız) tanımak için deformable part model kullanır. HOG, nesnelerin kenarlarını ve şekillerini tanımak için kullanılır. Göz durumlarını sınıflandırmak için kullanılabilir.

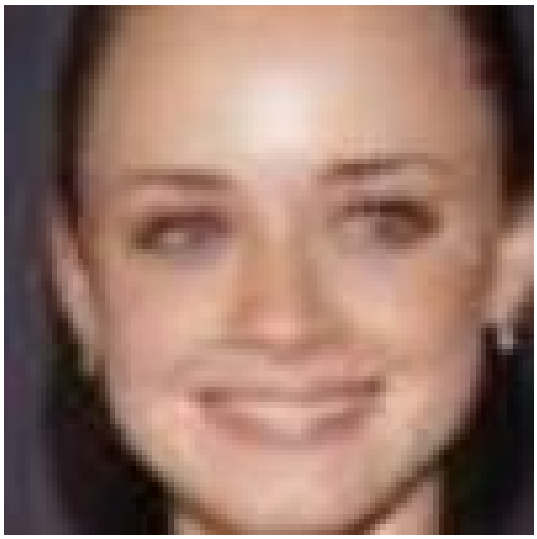
- Yorgunluk Tespiti : Perclos Sürücülerin yorgunluk seviyelerini tespit etmek için kullanılır. Sürücünün göz kapanma oranını (gözlerin ne kadar süreyle kapalı olduğunu) hesaplar. Yüksek Perclos değerleri, sürücünün yorgun olduğunu ve dikkatinin dağıldığını gösterebilir. Yöntemler genel olarak [1] baz alınarak belirlenmiştir.

$$P = \frac{\text{kapalı göz içeren kare sayısı}}{\text{toplam kare sayısı}} \times 100 \tag{1}$$

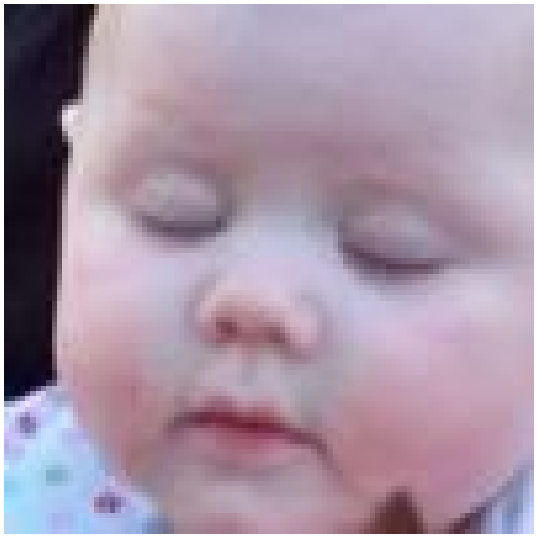
Yukarıdaki (1) numaralı eşitlik Perclos değerinin nasıl hesaplanacağını gösterir.

1.3 Ana Kod

- import cv2: OpenCV kütüphanesini içe aktarır, bu kütüphane görüntü işleme işlevleri sağlar.
- import dlib: Yüz tanıma ve yüz özelliklerini tespit etme işlevleri sağlayan bir kütüphane içe aktarır.
- from scipy.spatial import distance: İki nokta arasındaki Öklid mesafesini hesaplamak için kullanılan bir fonksiyon içe aktarır.



Şekil 1: Veri setinden açık gözlü yüz örneği [2].



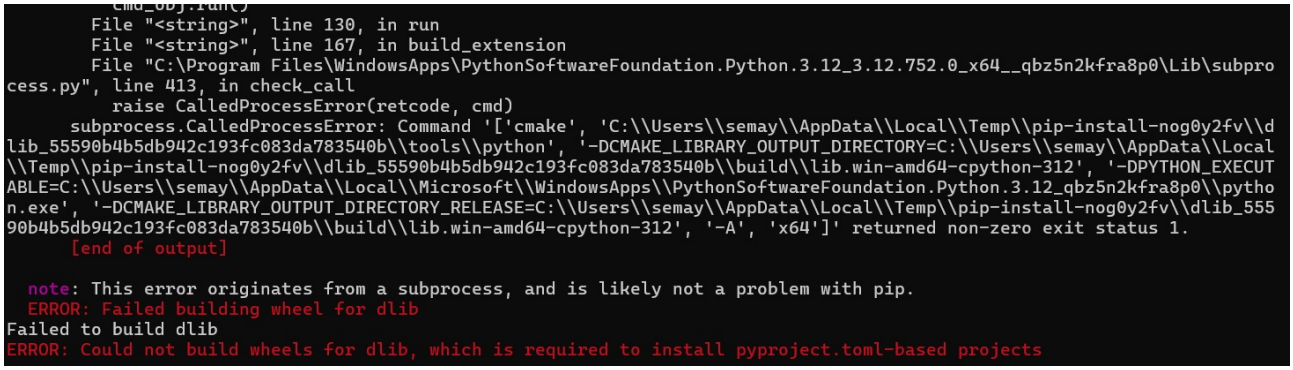
Şekil 2: Veri setinden kapalı gözlü yüz örneği [2].

- calculate EAR fonksiyonu: Gözün iç ve dış köşeleri ile üst ve alt kapakları arasındaki mesafeleri hesaplar ve gözün açıklık oranını (EAR) döndürür.
- Ana döngü (while True): Kamera görüntüsünü okur ve griye çevirir.
- hog face detector ile yüzleri tespit eder.
- Her tespit edilen yüz için, dlib facelandmark ile yüz özelliklerini (68 nokta) bulur. ı
- Sol ve sağ göz için özellik noktalarını toplar ve gözlerin EAR değerlerini hesaplar. Eğer DİS değeri 0.26'dan düşükse, kişinin uykulu olduğunu varsayar ve ekrana "DROWSY" ve "Are you Sleepy?" yazılarını çizer.DİS değerini konsola yazdırır.
- "Are you Sleepy" adlı pencereyi gösterir. ESC tuşuna basıldığında döngüyü kırar ve kamerayı serbest bırakır, tüm pencereleri kapatır.

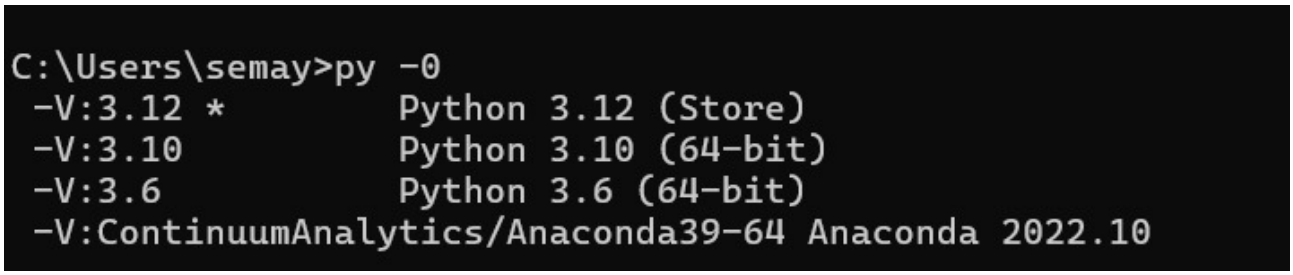
2 İkinci Aşama: Haar Cascade Sınıflandırıcı Kullanma

Bu hafta projelerde kullanabilmek için dlib kütüphanesini import etmem gerekti. Dlib yüz tanımlama gibi pek çok yerde kullanılan geniş kapsamlı ve açık kaynaklı bir kütüphane.

Bilgisayarımda başka bir python sürümü kullanmam gerektiğini öğrendim ve bunu standart olarak kullanabilmek için path olarak ekledim. Fakat yine de olumlu bir sonuç elde edemedim.

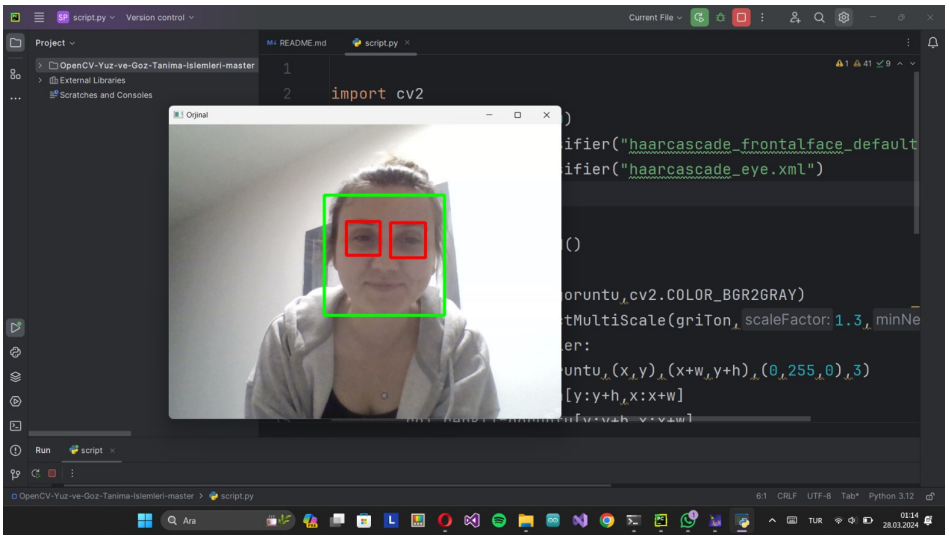


Şekil 3: Alınan hata.



Şekil 4: Python sürümleri.

Bilgisayarımda yaşadığım sürüm uyuşmazlığını çözemediğim için dlib olmadan bu projenin nasıl yapıldığını araştırdım. Sadece Opencv import ederek yani import cv2 diyerek yapılan bir kod örneği inceledim ve haarcascade ile de çalışıldığını gördüm.
Haarcascade için hazır kodların yanında kendimiz de oluşturabiliriz. Pozitif ve negatif resimlere ihtiyacımız olur. Bu resimleri etikletleme işlemi gibi işlemden geçirdikten sonra bir tarafa nesnenin varlığını diğer tarafa olmadığını tanıtarak bu belgeler .bat formunda kaydedilir. Sistem tarafından çalıştırılıp tanınması için de .xml uzantısı olarak değiştirilir. Bu değişimi gerekli araçlarla yapabilirsiniz. Ben [3] burada anlatılan sistemi yararlı buldum ve ilerleyen süreçte bu veri setini kullanacağım.



Şekil 5: Yüz ve göz tespiti.

Bu sisteme göz kapalılığını da eklemek istedim. Aslında bir sonraki hafta yapmam gerekn konu ama dlib yükleme süreci sebebiyle uzayacağını düşünüyorum.
Göz kapalılığını hesaplarken gözün etrafında bulunan 6 noktadan faydalanılır. Bu noktalar arası uzaklığını azalması ve belirli eşik değerin altında belirli bir süre kalması durumunda yorgunluk tespit edilmiş demektir. Bu duruma göre numpy ekledim ve hesaplama yaparak göz açıklık kapalılık kodunu yazmaya çalıştım. Kodun çalışması için dlib gerekiyormuş. Bu sebeple çalışmadı.

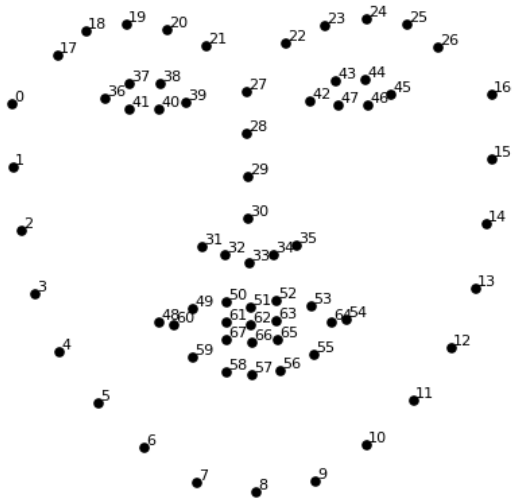
3 Üçüncü Aşama: Göz Kapalılığı Tespiti

Dlib kütüphanesini yüklemekle ilgili sorunum vardı ve bu sorunumu çözdüm. Sorunumu çözerken işlemlere baştan başladım. VsCode derleme araçlarını indirdim. Cmake kurdum. Daha sonra pip install dlib komutuyla bu kütüphaneyi indirdim [4].İndirip örnek projelerde kullanmaya devam edebilirdim fakat ben dlib olmadan kullandığım projede denedim. Open- close durumlarında hata aldım.Dlib ile denemek istedim. Dlib kütüphanesinde landmark denen bir özellik vardır.Dlib kütüphanesindeki "landmark" terimi, yüzdeki belirli noktaları veya bölgeleri tanımlamak için kullanılır.Dlib'in en yaygın kullanılan yüz landmark dedektörü, 68 noktalı bir yapıya sahiptir ve yüzün ana hatlarını hızlı ve güvenilir bir şekilde tespit edebilir.

Göz mesafesini hesaplamak için kullandığım standart olarak kullanılan eşik değerşu anlık bu ama değiştireceğim. Kullanıcıdan ilk önce gözünün açık ve kapalı hallerini isteyerek ona göre hesaplamaya geçilmesini düşünüyorum.

```python
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    dis = (A + B) / (2.0 * C)
    return dis
```

Şekil 6: Göz mesafesi hesaplama.



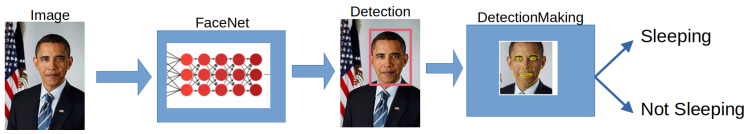
Şekil 7: Göz tespiti için Dlib kütüphanesinin kullandığı landmark değerleri[5].

4 Dördüncü Aşama: Geleneksel Yöntem ile Yeni Yöntemin Karşılaştırılması

Projede kullandığım yöntem geleneksel yöntemdi. Amacım, bu yöntem ile yeni teknolojilerden birini karşılaştırıp hangisinin daha sağlıklı sonuç verdiğini görmek. Yeni yöntem olarak CNN kullanılan bir örnek proje [6] buldum. CNN, yani Evrişimli Sinir Ağları (Convolutional Neural Networks), görüntü işlemede kullanılan bir derin öğrenme algoritmasıdır. Görselleri girdi olarak alır ve farklı operasyonlarla bu görsellerdeki özellikleri (features) yakalar ve sınıflandırır [7]. CNN, özellikle görüntü tanıma, nesne tespiti ve benzeri görsel tabanlı görevlerde etkili bir şekilde kullanılır. CNN'nin temel bileşenleri şunlardır:

Evrişim Katmanı (Convolutional Layer): Görüntü üzerinde gezinen ve özellikleri yakalayan filtreler içerir.
Aktivasyon Fonksiyonu (ReLU gibi): Modelin doğrusal olmayan özellikleri öğrenmesini sağlar.
Havuzlama Katmanı (Pooling Layer): Görüntü boyutunu küçültür ve özellikleri yoğunlaştırır.
Tam Bağlantılı Katman (Fully Connected Layer): Öğrenilen özellikleri kullanarak sınıflandırma yapar [8].

Bu katmanlar, bir görüntünün temel özelliklerinden daha karmaşık özelliklere kadar



Şekil 8: CNN ile sürücü yorgunluğu tespiti [9].

çeşitli soyutlama seviyelerinde özellikleri öğrenmek için bir arada çalışır. CNN modelleri, bu özellikleri öğrenmek ve güncellemek için eğitim sürecinde sürekli olarak kendilerini iyileştirir.

Tablo 1: Haarcascade ve CNN Karşılaştırması.

Özellik	**Haarcascade**	**CNN**
Hızlı tespit süresi	Evet	Hayır
Düşük kaynak gereksinimi	Evet	Hayır
Yüksek doğruluk	Hayır	Evet
Özellik mühendisliği gerekmez	Hayır	Evet
Yüksek hesaplama gücü gereksinimi	Hayır	Evet
Uzun eğitim süresi	Hayır	Evet

Yukarıda gördüğünüz üzere CNN daha iyi bir sonuç çıkarıyor. Eğer doğruluğu önemsemeseydim Haarcascade kullanabilirdim.CNN ile yapılmış projede Tensorflow ve Theano yüklemesinde sorun yaşadığım için çalıştıramadım. Cascade için de başın pozisyonunu algılayıp üç boyutlu sisteme dönüştürerek vücut postürünü algılayan fonksiyon eklendi.

Yeni CNN kodunda Tenforflow sürüm hatası aldım. Thaeno kütüphanesini Numpy ile benzer olması dolayısıyla kaldırdım.Kod sahibi de Numpy kullanmıştı zaten.Ayrıca göz aralıklarını hesaplayarak gözün kapalı açık durumunu tespit ettiğimiz landmark olarak adlandırdığımız noktalar mantığında ve yine aynı yöntemle ağız açıklığını tespit eden fonksiyonu da eklyerek çoklu faktör kontrolü sağlandı. Bu değerlerden 3 veya daha fazlası gerçekleşiyorsa kişi yorgun olarak tanımlanım konumuna göre en yakın otele yönlendirme yapıyor.

```
if(map_counter>=3):
    map_flag=1
    map_counter=0
    vlc.MediaPlayer('take_a_break.mp3').play()
    webbrowser.open("https://www.google.com/maps/search/hotels+or+motels+near+me")
```

Şekil 9: Yorgunluk eşiğine göre otele yönlendirme[6].

```
def getFaceDirection(shape, size):
    image_points = np.array( object: [
                                shape[33],    # Nose tip
                                shape[8],     # Chin
                                shape[45],    # Left eye left corner
                                shape[36],    # Right eye right corner
                                shape[54],    # Left Mouth corner
                                shape[48]     # Right mouth corner
                            ], dtype="double")

    # 3D model points.
    model_points = np.array([
                                (0.0, 0.0, 0.0),             # Nose tip
                                (0.0, -330.0, -65.0),        # Chin
                                (-225.0, 170.0, -135.0),     # Left eye left corner
                                (225.0, 170.0, -135.0),      # Right eye right corne
                                (-150.0, -150.0, -125.0),    # Left Mouth corner
                                (150.0, -150.0, -125.0)      # Right mouth corner

                            ])
```

Şekil 10: Baş pozisyonu tahmini için tanımlamalar[6].

5 Sonuç

Sonuç olarak projemde göz kapalılık durumu tespiti, ağza bakılarak uykululuk tespiti ve başın pozisyonuyla vücut duruşu tespiti sağladım. CNN'e bu kodlamaları geçirmeyi düşünüyorum. Çoğu yerde referans almıştım ve CNN e geçişte yine destek alsam da mantığı kendim kurarak ilerleyeceğim.

Referanslar

[1] MonaOmidyeganehShervinShirmohammadiBehnooshHariri, "Yawning detection dataset." `http://ieee-dataport.org/open-access/yawdd-yawning-detection-dataset`, Sun, 12/13/2020. 21.03.2024.

[2] M. R. Lab, "Mrl eye dataset." `http://mrl.cs.vsb.cz/eyedataset`, 2021. 21.03.2024.

[3] Dasar, "driver dataset." `https://www.mediafire.com/file/1aq02tpidk105fv/dasar_haartrain.rar/file`, Sun, 12/13/2020. Erişim tarihi: 21.03.2024.

[4] O. Şahin, "Windows'a dlib kütüphanesini kurma." `https://onursahin.net/windowsa-dlib-kutuphanesini-kurma/`, 2024. Erişim tarihi: 03.04.2024.

[5] B. Amos, "The 68 landmarks detected by dlib library." `https://www.researchgate.net/figure/The-68-landmarks-detected-by-dlib-library-This-image-was-created-by-Br fig2_329392737`, 2018. Erişim tarihi: 03.04.2024.

[6] V. Dhawan, "Driver drowsiness detection." https://github.com/SuperThinking/driver-drowsiness-detection, 2024. Erişim tarihi: 13.04.2024.

[7] T. Ergin, "Cnn nedir." https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad, 2024. Erişim tarihi: 13.04.2024.

[8] Özgür Doğan, "Cnn nedir." https://teknoloji.org/cnn-convolutional-neural-networks-nedir, 2024. Erişim tarihi: 13.04.2024.

[9] Erkan, "Driver drowsiness detection system process diagram." https://github.com/eadali/pytorch-drowsiness-detection/blob/main/doc/process.png, 2024. Erişim tarihi: 03.04.2024.

[10] aayushrai, "Driversafety." https://github.com/aayushrai/Driver_safety.git, 2019. 21.03.2024.

[11] bulentsezen, "tensorflowderinogrenme." https://github.com/bulentsezen/tensorflow_derin_ogrenme.git, 2022/2024. 21.03.2024.

[12] F. Cogen, "Driver fatigue detection with image processing." https://www.researchgate.net/publication/345253476_Driver_fatigue_detection_with_image_processing, 2020. 21.03.2024.