Mikroişlemciler Dersi Projesi

Raspberry pi kullanılarak Yüz Tanıma İle Kapı Kilidi Açma

içindekiler :

1. Projenin Tanıtımı ve Amacı :

Projenin Genel Özeti:

- **Projenin tanımı: Raspberry Pi, OpenCV kütüphanesi ve Python kullanılarak yüz tanıma tabanlı bir kapı kilit sistemi tasarladık.**
- **Hedef: Yetkisiz kişilerin girişini engelleyen, güvenli ve kullanıcı dostu bir kapı kilit sistemi oluşturmak.**
- **Özellikler: Sistem, bir web kamerası ile yüz algılar, tanımlar ve onaylanan yüzlerde kapıyı açar.**

Sistemin Hangi İhtiyacı Karşıladığı:

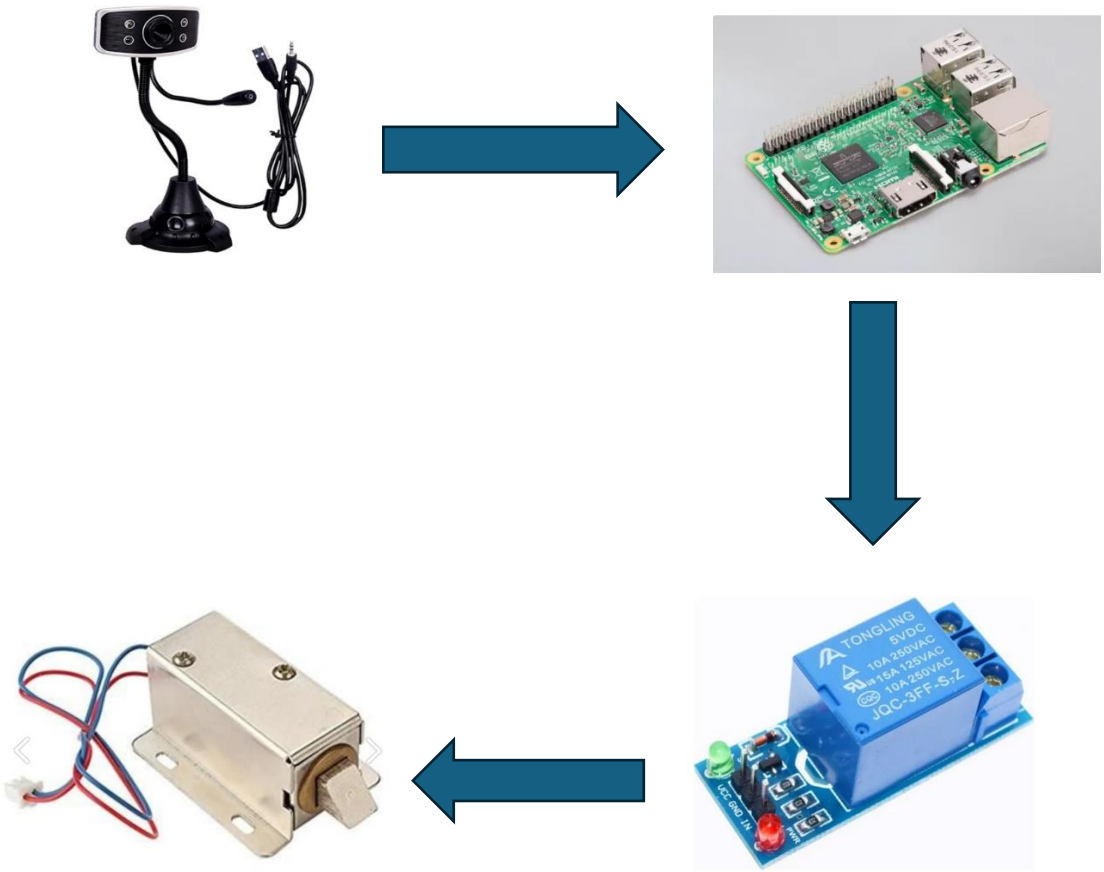
- **Ev veya işyerlerinde güvenlik ihtiyacını karşılar.**
- **Anahtar kaybolması veya çalınması gibi sorunlara çözüm sunar.**
- **Kullanıcıların yüz tanıma teknolojisi ile hızlı ve güvenli giriş yapmasını sağlar.**

Projenin Yenilikçiliği:

- **Uygulamada ekonomik ve kompakt bir platform olan Raspberry Pi'nin tercih edilmesi.**
- **Python diliyle yazılan açık kaynak kodların kullanılması.**
- **Yüz tanıma gibi ileri teknolojilerin, günlük hayatta kullanılabilir bir sistem olarak entegre edilmesi.**

Sistemin Temel Çalışma Prensibi:

1. **Web kamerası, kullanıcıdan gelen görüntüyü Raspberry Pi'ye aktarır.**
2. **OpenCV kütüphanesiyle yüz algılama ve tanıma gerçekleştirilir.**
3. **Tanınan yüz, önceden kaydedilmiş verilerle karşılaştırılır.**
4. **Doğrulama yapılırsa Raspberry Pi, tek kanallı röle yardımıyla solenoid kilidi açar.**
5. **Tanınmayan yüzlerde sistem güvenlik modunda kalır.**



2.Kullanılan Teknolojiler ve Araçlar :

2.1 Raspberry Pi :

- Raspberry Pi 3 Model B kullanılmıştır. Bu model, güçlü bir mikroişlemci ve geniş çevre birimi desteği ile projede merkezi bir rol oynamaktadır.

Raspberry Pi'nin Temel Özellikleri:

- **İşlemci: 1.2GHz 64-bit Dört Çekirdekli ARM Cortex-A53**
- **RAM: 1GB LPDDR2 SDRAM**
- **Bağlantı Seçenekleri: Ethernet, Wi-Fi, Bluetooth 4.1**
- **GPIO Pinleri: 40 pin, çeşitli sensör ve cihaz bağlantıları için kullanılabilir**
- **Depolama: microSD kart desteği**
- **Görüntü Çıkışı: HDMI ve DSI ekran desteği**
- **USB Portları: 4 adet USB 2.0 portu**

Projedeki Kullanımı:

Raspberry Pi, kamera modülünden gelen görüntüleri işlemek ve yüz tanıma algoritmalarını çalıştırmak için kullanılmıştır. Aynı zamanda elektronik kilidi kontrol etmek ve kullanıcı arayüzü için temel platform görevi görmektedir.

2.2 Kamera Modülü:

- **Yüz tanıma için projede USB bağlantılı bir webcam kullanılmıştır. Bu tür kameralar, standart USB portları üzerinden Raspberry Pi ile kolayca entegre edilebilir.**

Webcam Teknik Özellikleri:

- **Bağlantı Tipi: USB 2.0**
- **Çözünürlük: 720p (1280x720) veya üzeri**
- **Kare Hızı: 30 FPS (genel kullanım için yeterli)**
- **Desteklenen Formatlar: MJPEG, YUY2**

Projedeki Kullanımı:

Webcam, Raspberry Pi'ye bağlı olarak sürekli bir görüntü akışı sağlar. Bu görüntüler, OpenCV gibi kütüphaneler yardımıyla işlenir ve yüz tanıma algoritmasına aktarılır. Kameranın USB üzerinden bağlanması, kullanım kolaylığı ve geniş uyumluluk sağlar.

2.3 Elektronik Kilit Sistemi:

Projede kapının açılıp kapanmasını kontrol etmek için 12V Selenoid Kilit kullanılmıştır. Bu tür kilitler, manyetik bir bobin kullanarak mekanizmayı aktif hale getirir ve güvenilir bir kapı kontrol sistemi sağlar.

Selenoid Kilidin Teknik Özellikleri:

- **Çalışma Voltajı: 12V DC**
- **Akım Tüketimi: 500mA (ortalama)**
- **Aktivasyon Mekanizması: Elektromıknatıs**
- **Montaj: Sabit çelik kasa, kolay kurulum**
- **Uygulama Alanı: Kapı kilit mekanizmaları, çekme/itme sistemleri**

Projedeki Kullanımı:

- **Selenoid kilit, Raspberry Pi'nin GPIO pinlerinden kontrol edilen bir röle aracılığıyla çalıştırılmıştır.**
- **Yüz tanıma algoritması tarafından doğrulama yapıldığında Raspberry Pi röleye sinyal gönderir. Röle, selenoid kilidi aktif hale getirerek kapının açılmasını sağlar.**

Devre Bağlantısı:

- **Raspberry Pi'nin GPIO pinlerinden biri röle modülüne bağlanır.**
- **Röle, 12V DC güç kaynağı ile selenoid kilit arasında yer alır ve bir anahtar görevi görür.**
- **Doğru polarite sağlanarak röle kontrol edilir.**

2.4 Diğer Donanımlar :

Tek Kanallı Röle (5V)

- **Çalışma Voltajı: 5V DC**
- **Kanal Sayısı: Tek kanal**
- **Çıkış Kapasitesi: Maksimum 10A 250V AC veya 10A 30V DC**
- **Projedeki Kullanımı: Raspberry Pi GPIO pinlerinden gelen sinyal ile kontrol edilerek, selenoid kilit için bir anahtar görevi görür.**

18650 Şarj Edilebilir Pil (3 adet, 3.7V, 2600mAh)

- **Kapasite: 2600mAh (her biri)**
- **Gerilim: 3.7V**
- **Toplam Güç: Seri bağlantıda toplamda 11.1V sağlanır.**
- **Projedeki Kullanımı: Sistemin güç ihtiyacını karşılamak için kullanılmıştır. Pil yuvası üzerinden bağlanmıştır.**

Jumper Kablolar

- **Tip: Erkek-Erkek, Erkek-Dişi, Dişi-Dişi tiplerinde**
- **Projedeki Kullanımı: Raspberry Pi ile röle, selenoid kilit ve diğer donanımlar arasında bağlantılar kurmak için kullanılmıştır.**

3.Mikroişlemci ve Yazılım Modeli :

3.1 Raspberry Pi'nin İşleyişi:

Raspberry Pi'nin Görevleri:

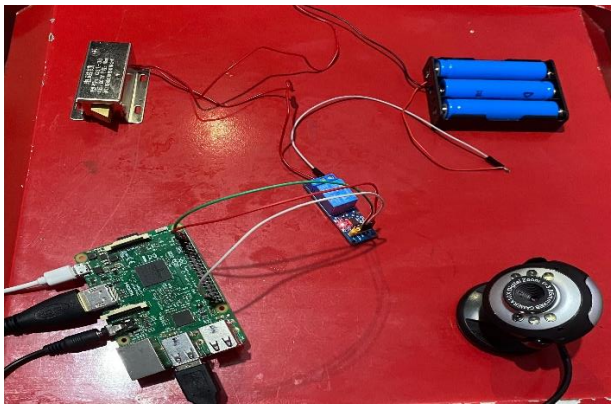
- **Kamera modülünden gelen görüntü verilerini almak.**
- **Görüntü işleme algoritmalarını (yüz tanıma) çalıştırmak.**
- **Doğrulama sonrası GPIO pinleri üzerinden elektronik kilit ve röle sistemini kontrol etmek.**
- **Kullanıcı arayüzü veya durum mesajları için bir monitöre (isteğe bağlı) bağlantı sağlamak.**

İşlem Akışı:

1. **Kamera modülü Raspberry Pi'ye sürekli görüntü akışı sağlar.**
2. **Görüntü, Raspberry Pi'de çalışan Python programı tarafından alınır.**
3. **Program, OpenCV kütüphanesini kullanarak yüz tanıma algoritmasını uygular.**
4. **Eğer tanınan yüz, önceden tanımlanan veritabanında mevcutsa:**
    - **GPIO pinleri aracılığıyla röle modülü aktif hale getirilir.**
    - **Elektronik kilit açılır ve kapı serbest bırakılır.**
5. **Eğer yüz tanınmazsa:**
    - **(Kilid açılamadı!) mesajı gönderilir.**
6. **Sistem, bir sonraki giriş denemesine hazır hale gelir.**

Teknik Özellikler:

- **Giriş: Kamera modülünden gelen görüntüler.**
- **İşlem: Yüz tanıma algoritması ve doğrulama.**
- **Çıkış: Röle kontrolü ile elektronik kilit açma sinyali.**



3.2 Python Kodlama ve Algoritma:

Python, yüz tanıma projelerinde yaygın olarak kullanılan bir dildir çünkü birçok güçlü kütüphane sunar. Bu kütüphaneler, Raspberry Pi ile entegrasyon kolaylığı sağlar ve yüz tanıma algoritmalarını basit ve verimli bir şekilde uygulamayı mümkün kılar.

1. Yüz Tanıma İçin Kullanılan Python Kütüphaneleri:

OpenCV (Open Source Computer Vision Library), bilgisayarla görme ve görüntü işleme alanlarında yaygın olarak kullanılan açık kaynaklı bir kütüphanedir. Yüz tanıma, hareket algılama, nesne takibi, renk analizi, video işleme ve daha birçok görüntü işleme uygulamasını kolaylıkla geliştirmek için kullanılır. Python'da da oldukça yaygın bir şekilde kullanılmaktadır.

OpenCV'nin Temel Özellikleri:

- **Çeşitli Görüntü İşleme Fonksiyonları: Görüntü dönüştürme, filtreleme, morfolojik işlemler, kenar algılama, histograma analiz gibi birçok görüntü işleme fonksiyonuna sahiptir.**
- **Yüz Tanıma ve Algılama: Haar Cascades, LBPH (Local Binary Patterns Histograms) gibi yöntemlerle yüz tanıma işlemleri yapılabilir.**
- **Video ve Görüntü Akışı: Kamera bağlantıları, video dosyaları ile çalışma ve gerçek zamanlı video akışı desteği sunar.**
- **Makine Öğrenimi ve Derin Öğrenme Desteği: Yüz tanıma gibi derin öğrenme uygulamaları için eğitim ve tahmin yapma imkanı tanır.**

OpenCV ile Yüz Tanıma

Yüz tanıma, genellikle yüzlerin tespiti ve ardından kimlik doğrulama işlemi olarak iki aşamada gerçekleştirilir. OpenCV bu işlemleri iki şekilde sağlar: Yüz Algılama ve Yüz Tanıma.

1. Yüz Algılama

Yüz algılama, bir görüntüdeki yüzleri tespit etmeyi içerir. OpenCV, Haar Cascade ve HOG (Histogram of Oriented Gradients) gibi çeşitli algoritmalar kullanarak yüz algılamayı gerçekleştirebilir.

Haar Cascade ile Yüz Algılama

Haar Cascade, görüntülerdeki nesneleri tanımak için kullanılan klasik bir algoritmadır. Yüz gibi nesneler, farklı özelliklerinin (örneğin, gözler, burun, ağız) tanınmasıyla tespit edilir.

2. Yüz Tanıma

Yüz tanıma, tespit edilen yüzün kimliğini doğrulamak için kullanılır. Yüz tanımada genellikle daha gelişmiş yöntemler kullanılır.

LBPH (Local Binary Patterns Histograms) Yöntemi

LBPH, bir yüzü tanımak için kullanılan klasik bir tekniktir. Yüzdeki yerel özellikleri analiz ederek her bir yüz için bir histogram çıkarır ve ardından yüzü tanımlar.

3. OpenCV ile Kapı Kilidi Kontrolü

Yüz tanıma işlemi başarıyla tamamlandığında, kilidi açmak için Raspberry Pi'nin GPIO pinleri kullanılabilir. Yüz tanıma sonrasında GPIO pinleri üzerinden bir röle kontrol edilebilir.

NumPy (Numerical Python), Python için güçlü bir bilimsel hesaplama kütüphanesidir ve özellikle büyük, çok boyutlu diziler ve matrisler üzerinde işlem yapabilmeyi sağlar. NumPy, birçok matematiksel, istatistiksel ve mühendisliksel işlem için optimizasyonlar sağlar. Bu kütüphane, OpenCV gibi görüntü işleme kütüphaneleri ile birlikte sıklıkla kullanılır çünkü görüntüler aslında birer çok boyutlu (2D/3D) diziler olarak temsil edilir.

NumPy'nin Temel Özellikleri:

- **Diziler (Arrays): NumPy, büyük verileri verimli bir şekilde depolamak ve işlemek için çok boyutlu diziler sağlar.**
- **Vektörleştirilmiş İşlemler: NumPy, Python'da döngü kullanmak yerine, vektörleştirilmiş işlemlerle hesaplama yapmanızı sağlar ve bu, hız açısından büyük avantaj sağlar.**
- **Matematiksel Fonksiyonlar: NumPy, matris işlemleri, lineer cebir, istatistiksel hesaplamalar gibi işlemler için fonksiyonlar sunar.**

NumPy'nin OpenCV ile Kullanımı:

OpenCV'de, görüntüler aslında NumPy dizileri olarak depolanır. Her bir piksel, NumPy dizisindeki bir eleman olarak temsil edilir. Görüntü işleme işlemleri (filtremeler, dönüşümler, matematiksel hesaplamalar) için NumPy fonksiyonları yaygın olarak kullanılır.

Pickle, Python'da veri serileştirme ve deseralize etme (serialize/deserialize) işlemleri için kullanılan bir kütüphanedir. Yani, bir Python nesnesini (listeler, sözlükler, sınıflar, vb.) byte dizisine dönüştürmek ve daha sonra bu byte dizisini orijinal nesneye geri dönüştürmek için kullanılır. Pickle, model eğitimi ve sonuçların saklanması gibi işlemlerde çok faydalıdır.

Pickle'ın Temel Özellikleri:

- **Serileştirme: Python nesnelerini bir dosya veya başka bir veri biçiminde depolamak için serileştirme yapılır. pickle.dump() fonksiyonu ile nesne serileştirilebilir.**
- **Deseralize Etme: Serileştirilen veriyi tekrar orijinal haline getirmek için pickle.load() fonksiyonu kullanılır.**
- **Veri Depolama: Model eğitimi gibi işlemler sonunda elde edilen model sonuçlarını (örneğin, yüz tanıma modeli) bir dosyaya kaydedip daha sonra tekrar yüklemek için kullanılabilir.**

3.3 Yüz Tanıma Teknolojisi ve Algoritmaları :

1. Haar Cascades

Haar Cascades, yüz tespiti için yaygın olarak kullanılan bir makine öğrenimi tabanlı yöntemdir. OpenCV'nin sağladığı önceden eğitilmiş Haar Cascade sınıflandırıcıları, yüz, göz, ağız gibi öğeleri tespit etmek için kullanılır. Bu algoritma, Haar özellikleri adı verilen temel görsel özellikleri kullanır ve bu özellikleri sınıflandırmak için AdaBoost algoritmasını kullanarak bir kaskad (cascade) yapısı oluşturur.

Haar Cascades'in Çalışma Prensibi:

1. **Görüntü üzerinde belirli bir alan taranır (sliding window).**
2. **Her tarama adımında, bölgedeki görsel özellikler çıkarılır.**
3. **Bu özellikler, daha önce eğitilmiş bir sınıflandırıcıya gönderilir.**
4. **Yüz olup olmadığına karar verilir.**

Avantajları:

- **Hızlı ve verimli.**
- **Gerçek zamanlı uygulamalar için uygundur.**

Dezavantajları:

- **Yüzün pozisyonu, ışıklandırma gibi değişkenlerden etkilenebilir.**

2. DLIB

DLIB, daha gelişmiş ve doğru yüz tespiti için kullanılan bir kütüphanedir. Özellikle yüz tanıma ve özellik çıkarımı konusunda güçlüdür. DLIB, yüz tespiti için HOG (Histogram of Oriented Gradients) ve CNN (Convolutional Neural Networks) tabanlı yaklaşımlar kullanır.

DLIB'in Çalışma Prensibi:

1. **Yüz, HOG veya CNN ile tespit edilir.**
2. **Yüzdeki özellikler (göz, burun, ağız gibi) çıkarılır.**
3. **Çıkarılan özellikler, daha sonra yüz tanımada kullanılır.**

Avantajları:

- **Yüksek doğruluk oranı.**
- **Farklı pozisyonlardaki yüzleri tespit edebilme yeteneği.**

Dezavantajları:

- **Daha yüksek işlem gücü ve zaman gerektirir.**
- **Yavaş olabilir, özellikle gerçek zamanlı uygulamalarda.**

3. OpenCV

OpenCV, yüz tanıma için çeşitli algoritmalar ve araçlar sunan açık kaynaklı bir kütüphanedir. OpenCV'nin içerisinde Haar Cascade, LBPH (Local Binary Patterns Histograms), DLIB ve diğer algoritmalarla yüz tanıma yapılabilir.

OpenCV Yüz Tanıma Algoritmaları:

- **Haar Cascades: Yüz tespiti için kullanılır.**
- **LBPH (Local Binary Pattern Histogram): Yüz tanıma için kullanılır, daha basit ve hızlıdır.**
- **DNN (Deep Neural Network): Yüz tespiti ve tanıma için derin öğrenme tabanlı yöntemleri kullanır.**

Yüz Tanıma Algoritmalarının Uygulaması

Yüz tanıma uygulamanızda bu algoritmalardan biri veya birkaçını kullanabilirsiniz. Kodda, her bir algoritmanın nasıl entegre edildiğini ve nasıl çalıştığını gösterebiliriz.

Algoritma Akış Diyagramı:

Algoritma akış diyagramı, yüz tanıma sisteminin adım adım nasıl çalıştığını görsel olarak anlatır. İşte örnek bir akış diyagramı:

1. **Görüntü Alma: Kamera veya bir görüntü dosyasından veri alınır.**
2. **Öznitelik Çıkarımı: Haar Cascades, DLIB veya OpenCV ile yüz tespiti yapılır.**
3. **Yüz Tanıma: Yüz tanıma algoritması ile tespit edilen yüzler, veri tabanındaki kayıtlarla karşılaştırılır.**
4. **Sonuçlar: Tanınan kişi açılır/kapalı işlemi yapılır veya belirli bir işlem tetiklenir.**

Kodlar Ve Anlatımı:

1. Yeni Kullanıcı Sınıfı (Yüz Verisi Kaydetme)

Bu sınıf, yeni bir kullanıcı eklemek için kullanılır ve temel işleyişi şu şekilde:

- **Kütüphaneler: OpenCV, os, sys gibi kütüphaneler kullanılıyor. cv2.CascadeClassifier ile Haar cascades yüz tespiti gerçekleştirilir.**
- **Kullanıcı Adı: Program ilk çalıştırıldığında kullanıcıdan isim istenir ve bu isimle bir klasör oluşturulur (kullanıcının yüz verilerini kaydetmek için).**
- **Görüntü Toplama: Kullanıcı kameraya karşı poz verirken, yüz tespiti yapılır ve her bir yüz görüntüsü images klasöründe, kullanıcının adıyla numaralandırılmış bir dosya olarak kaydedilir.**
- **Kontrol: Kamera üzerinden gelen görüntüler sürekli olarak işlenir. Eğer 100 adet yüz verisi kaydedilmişse döngü durdurulur.**

Önemli Fonksiyonlar:

- **cv2.VideoCapture(0, cv2.CAP_DSHOW): Kamera kullanılır.**
- **cv2.CascadeClassifier(): Yüz tespiti için Haar cascades sınıflandırıcısı kullanılır.**
- **`cv2.imwrite()`: Yüz görüntülerini kaydeder.**

```python
# Kullanacağımız kütüphaneleri koda dahil ediyoruz.
import cv2
import os
import sys

# Dosyanın bulunduğu konumu buluyoruz.
path = os.path.dirname(os.path.abspath(__file__))
# Yüz tespiti için kullanacağımız Classifier'ın yolunu koda belirtiyoruz.
detector = cv2.CascadeClassifier(path+r'\Classifiers\face.xml')

# Kod için video değil kamera kullanacağımızı '0' yazarak belirtiyoruz.
camera = cv2.VideoCapture(0, cv2.CAP_DSHOW)
# Kameranın boyut ve çözünürlük ayarlarıwaelnı yapıyoruz.
camera.set(3, 640)
camera.set(4, 480)
minW = 0.1*camera.get(3)
minH = 0.1*camera.get(4)
# Ön yüz tespit için kullanacağımız dokümanı belirtiyoruz.
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# Kod çalıştığında kaydedilecek yeni kullanıcı için isim soruyoruz.
name = input("Lütfen kullanıcı adınızı giriniz: ")
# Aldığımız kullanıcı adı için images klasörünün içinde kullanıcının adıyla bir klasör oluşturuyoruz.
dirName = "./images/" + name
print(dirName)
# Eğer kullanıcı adı önceden kullanılmamışsa klasörü oluşturuyoruz.
if not os.path.exists(dirName):
    os.makedirs(dirName)
    print("Klasör oluşturuldu")
# Kullanıcı adı mevcutsa bunu belirtip programdan çıkış yapıyoruz.
else:
    print("İsim önceden kullanılmış")
```

```python
count = 1
# Sonsuz döngü başlatıyoruz.
while True:
    # Tanımladığımız sayıcının 100 olup olmadığını kontrol ediyoruz. 100 ise döngüden çıkıyoruz.
    if count >= 100:
        break
    # Kameradan gelen görüntüler okunuyor.
    ret, im = camera.read()
    # Gelen görüntüler renkliden griye çevriliyor.
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    # Görüntülerde ki yüzler tespit ediliyor.
    faces = faceCascade.detectMultiScale(gray)
    # Gelen görüntülerde ki bulunan yüzlerin x, y koordinatı ve width(genişlik), height(uzunluk) bilgilerini alıyoruz.
    for (x, y, w, h) in faces:
        roiGray = gray[y:y+h, x:x+w]
        # Bu görüntüleri oluşturduğumuz kullanıcıya ait klasörün içine sırayla kaydediyoruz.
        fileName = dirName + "/" + name + str(count) + ".jpg"
        cv2.imwrite(fileName, roiGray)
        cv2.imshow("face", roiGray)
        # Bulunan yüzlerin etrafına dikdörtgen ekliyoruz.
        cv2.rectangle(im, (x, y), (x+w, y+h), (0, 255, 0), 2)
        # Sayıcıyı her döngünün sonunda 1 arttırıyoruz.
        count += 1
    # Görüntüyü ekrana vermek için olan kod satırı.
    cv2.imshow('im', im)
    # Görüntünün ekranda kalmasını sağlayan kod satırı.
    key = cv2.waitKey(10)
    # Eğer ki "ESC"(Escape) tuşuna basılırsa program sonlanıyor.
    if key == 27:
        break
camera.release()
cv2.destroyAllWindows()
```

2. Veri İşleme Sınıfı (Yüz Verisi Eğitimi)

Bu sınıf, yüz verilerini eğitmek ve model oluşturmak için kullanılır.

- **Hedef: Yüz verisi toplanan kullanıcıları tanımak için bir yüz tanıma modeli oluşturulur. Burada kullanılan algoritma LBPH (Local Binary Pattern Histogram)'dır.**
- **Veri Okuma: images klasöründeki tüm yüz görüntüleri okunur ve xTrain (özellikler) ve yLabels (etiketler) listelerine eklenir.**
- **Etiketleme: Her bir kullanıcının adı bir etiket olarak kaydedilir. Bu etiketler daha sonra yüz tanıma sırasında kimlik doğrulamak için kullanılır.**
- **Model Eğitimi: recognizer.train() fonksiyonu ile yüz verileri eğitilir ve trainer.yml dosyası kaydedilir. Bu dosya, daha sonra yüz tanımada kullanılacaktır.**

Önemli Fonksiyonlar:

- **cv2.face.LBPHFaceRecognizer_create(): Yüz tanıyıcıyı oluşturur.**
- **recognizer.train(xTrain, np.array(yLabels)): Eğitim işlemi.**
- **recognizer.save(): Modelin kaydedilmesi.**
- **pickle.dump(): Etiketlerin kaydedilmesi.**

```python
# Bu kod trainer.yml ve labels dosyalarını oluşturur
# Bunlar giriş kodunda kullanılarak kıyaslamayı mümkün kılar.
import os
import numpy as np
from PIL import Image
import cv2
import pickle

# Yüz tespiti için kullanacağımız Classifier'ın yolunu koda belirtiyoruz.
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
# OpenCV kütüphanesinde bulunan LBPH (Local Binary Pattern Histogram) yüz tanıyıcı kullanıyoruz.
recognizer = cv2.face.LBPHFaceRecognizer_create()

# Bulunan dosya yolunu tespit edip images klasörüne ulaşılır
baseDir = os.path.dirname(os.path.abspath(__file__))
imageDir = os.path.join(baseDir, "images")

currentId = 1
labelIds = {}
yLabels = []
xTrain = []

# Bulduğu her bir görüntüyü tek tek gezer ve bu görüntüleri NumPy dizisine dönüştürür
for root, dirs, files in os.walk(imageDir):
    print(root, dirs, files)
    for file in files:
        print(file)
        if file.endswith("png") or file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(root)
            print(label)
            if not label in labelIds:
                labelIds[label] = currentId
                print(labelIds)
                currentId += 1

            # Doğru görüntülere sahip olduğumuzdan emin olmak için yüz algılamayı tekrar gerçekleştiriyoruz.
            # Ve sonra kıyaslama verilerini hazırlıyoruz
            id_ = labelIds[label]
            pilImage = Image.open(path).convert("L")
            imageArray = np.array(pilImage, dtype: "uint8")
            faces = faceCascade.detectMultiScale(imageArray)
            # Bulunan yüzlerin x, y koordinatı ve width(genişlik), height(uzunluk) bilgilerini alıyoruz.
            for (x, y, w, h) in faces:
                roi = imageArray[y:y + h, x:x + w]
                xTrain.append(roi)
                yLabels.append(id_)
# Dizin adlarını ve etiket kimliklerini içeren sözlüğü saklıyoruz.
with open("labels", "wb") as f:
    pickle.dump(labelIds, f)
    f.close()
# Verileri işliyoruz ve dosyayı kaydediyoruz.
recognizer.train(xTrain, np.array(yLabels))
recognizer.save("trainer.yml")
print(labelIds)
```

3. Yüz Tanıma ve Kilit Açma Sınıfı

Bu sınıf, yüz tanıma algoritmasını kullanarak tanınan kullanıcıyı belirler ve buna bağlı olarak kapı kilidini açar veya kapatır.

- **Yüz Tanıma: Kameradan alınan görüntülerde yüzler tespit edilir ve LBPH yüz tanıyıcı kullanılarak hangi kullanıcının tanındığı belirlenir.**
- **Kilit Kontrolü: Eğer yüz tanınırsa, unlock() fonksiyonu çağrılır ve kilit açılır. Tanınmazsa, lock() fonksiyonu çağrılır ve kilit kapanır.**
- **Güven Katsayısı: Yüz tanımada güven katsayısı hesaplanır. Eğer güven seviyesi düşükse, "Bilinmeyen" (unknown) olarak kabul edilir.**
- **Kamera Çıkışı: Tanıma sonucu ekranda gösterilir.**

Önemli Fonksiyonlar:

- **cv2.face.LBPHFaceRecognizer_create(): Yüz tanıyıcıyı oluşturur.**
- **recognizer.predict(): Yüzü tanımak için kullanılır.**
- **cv2.putText(): Yüzün üzerine isim ve güven yüzdesi ekler.**

GPIO ile Kilit Kontrolü:

- **Kilit açma ve kapama işlemi için GPIO pinleri kullanılır. Bu kısımda, fiziksel bir röle ile kapı kilidi kontrol edilir.**

```python
# Gerekli kütüphaneleri içeri aktarıyoruz
import cv2
import os
import pickle
# import RPi.GPIO as GPIO


_id = 0

# Kilitle
def lock(pin):  2 usages
    # GPIO.output(pin, GPIO.LOW)
    print('Kilitlendi!')


# Kilit Aç
def unlock(pin):  1 usage
    # GPIO.output(pin, GPIO.HIGH)
    print('Kilit Açıldı!')


# Kaydını yaptığımız kullanıcıların adlarını 'names' listesine ekliyoruz
names = ['None']

# GPIO26 pinini seçtiğimizi belirtiyoruz
role_pini = [26]
# Gelebilecek gereksiz uyarıları devre dışı bırakıyoruz
# GPIO.setwarnings(False)
# GPIO numaralarına göre seçim yaptık
# GPIO.setmode(GPIO.BCM)
# Seçtiğimiz röle pinini çıkış olarak ayarlıyoruz.
# GPIO.setup(role_pini, GPIO.OUT)
```

```python
lock(role_pini)

# Oluşturduğumuz etiket dosyasını açıyoruz ve yüklüyoruz
with open('labels', 'rb') as f:
    dicti = pickle.load(f)
    f.close()

# Kamera kullanacağımızı belirtiyoruz
camera = cv2.VideoCapture(0, cv2.CAP_DSHOW)

# Kameranın boyut ve çözünürlük ayarlarını yapıyoruz.
camera.set(3, 640)
camera.set(4, 480)
minW = 0.1 * camera.get(3)
minH = 0.1 * camera.get(4)

path = os.path.dirname(os.path.abspath(__file__))
# Yüz tespiti için kullanacağımız Classifier'ın yolunu koda belirtiyoruz.
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# OpenCV paketinde bulunan LBPH (Local Binary Pattern Histogram) yüz tanıyıcı kullanıyoruz.
recognizer = cv2.face.LBPHFaceRecognizer_create()

# Oluşturulan eğitici dosyayı açıyoruz
recognizer.read("trainer.yml")

# Yazı tipi belirleme
font = cv2.FONT_HERSHEY_SIMPLEX

# Sonsuz Döngü
while True:
```

```
    # Kameradan gelen görüntüler okunuyor.
    ret, im = camera.read()

    # Gelen görüntüler renkliden griye çevriliyor.
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

    # Görüntülerde ki yüzler tespit ediliyor.
    faces = faceCascade.detectMultiScale(gray, minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE)
    for (x, y, w, h) in faces:
        # Yüzlerin etrafına dikdörtgen çiziliyor
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 2)
        _id, confidence = recognizer.predict(gray[y:y + h, x:x + w])

        # Güven katsayısının 100'den küçük olması gerekiyor. 0 olması en iyi eşleşmeyi temsil eder
        if confidence < 50:
            unlock(role_pini)
            # Kullanıcı ismi tespit edilip yerine koyulur
            _id = names[_id]
            # Güven hesaplanır
            confidence = "  {0}%".format(round(100 - confidence))
        else:
            lock(role_pini)
            # Tespit edilemeyen yüzler için 'Unknown'(Bilinmeyen) yazılır
            _id = "unknown"
            # Güven hesaplanır
            confidence = "  {0}%".format(round(100 - confidence))

        # Yüzün tespit edildiği yere isim ve güven yüzdesi yazılır
        cv2.putText(im, str(_id), (x + 5, y - 5), font, 1, (255, 255, 255), 2)
        cv2.putText(im, str(confidence), (x + 5, y + h - 5), font, 1, (255, 255, 0), 1)
```

```
    # Kamera açık tutulur
    cv2.imshow('camera', im)

    # Programın sonlanması için 'ESC' tuşuna basılması beklenir
    k = cv2.waitKey(1) & 0xff
    if k == 27:
        break


# GPIO.cleanup()
print("\n [BILGI] Programdan çıkış yapılıyor. Gerekli temizlikler yapılıyor..")
camera.release()
cv2.destroyAllWindows()
```

4.Beklenen Sonuçlar ve Sistem İşleyişi :

1- Yüz Tanıma ve Kullanıcı Ekleme:

- **Beklenen Sonuç: Kullanıcı adı verildikten sonra sistem, kullanıcının yüzünü tanıyacak şekilde 100 adet görüntü kaydeder ve bu görüntüler "images" klasöründe kullanıcı adına göre adlandırılır. Her yüz verisi doğru şekilde kaydedilmelidir.**
- **Sistem İşleyişi: Kullanıcı adı girildiğinde, sistem, belirtilen klasöre yüz verilerini ekler. Bu işlemin başarıyla tamamlandığı, "Klasör oluşturuldu" mesajı ile kullanıcıya bildirilir. Ayrıca, kullanıcı adı daha önce kullanılmışsa, sistem bu durumu tespit eder ve yeni bir kullanıcı eklenmesini engeller.**

2- Veri Eğitimi ve Model Kaydı:

- **Beklenen Sonuç: Sistem, toplanan yüz verilerini işleyerek, bir yüz tanıma modelini (trainer.yml) başarıyla eğitir. Etiketler (labels) dosyası da oluşturulmalıdır.**
- **Sistem İşleyişi: Yüz verileri işlendikten sonra, trainer.yml ve labels dosyaları oluşturulur. Bu dosyalar, yüz tanıma için temel eğitim verilerini içerir ve daha sonra yüz tanımada kullanılmak üzere saklanır. Eğitilen modelin doğruluğu, verilerin kalitesine ve çeşitliliğine bağlı olarak değişebilir.**

3- Yüz Tanıma ve Kilit Açma:

- **Beklenen Sonuç: Sistem, kamera aracılığıyla gelen görüntülerdeki yüzleri doğru bir şekilde tespit eder ve eşleşen kullanıcıyı tanır. Tanınan kullanıcı için sistem "kilit açma" komutunu verir, tanınmayan kullanıcılar için ise "kilit kapalı" komutunu verir.**
- **Sistem İşleyişi: Kamera görüntüsü alındığında, yüz tespiti yapılır ve LBPH yüz tanıyıcı kullanılarak tanıma yapılır. Yüz tanıma sonucu, doğru eşleşme sağlanmışsa, kilit açılır. Eğer yüz tanınmazsa, kilit kapanır ve "unknown" olarak ekrana yansıtılır. Yüz tanıma işlemi her saniye yenilenir.**

4- Güvenlik ve Hata Yönetimi:

- **Beklenen Sonuç: Sistem, düşük güven katsayısı (confidence) durumunda kullanıcıyı "Bilinmeyen" olarak etiketler ve sistemin güvenliğini sağlamak için kilidi kapalı tutar.**
- **Sistem İşleyişi: Yüz tanıma güven katsayısı 50'nin altında olduğunda, sistem kilidi kapalı tutar ve "unknown" mesajını verir. Eğer tanıma güvenilirse (confidence değeri 50'nin üzerinde), kullanıcıyı doğrular ve kilidi açar.**

5- Kullanıcı Dostu Deneyim:

- **Beklenen Sonuç: Kullanıcı, sisteme kolayca yeni bir kullanıcı ekleyebilir ve tanınan kullanıcılar için kapı kilidini açabilir.**
- **Sistem İşleyişi: Kullanıcı dostu bir arayüz sunulur. Kullanıcı adı girildiğinde, yüz kaydı yapılır ve sistem otomatik olarak çalışmaya başlar. Sistemin doğru çalıştığının bir göstergesi olarak ekranda kullanıcı adı ve güven yüzdesi görüntülenir.**

6- Performans ve Doğruluk:

- **Beklenen Sonuç: Yüz tanıma, en az %90 doğruluk oranına sahip olmalıdır. Yüksek doğruluk oranı, kullanıcıların güvenle sistemin kilit açma işlevini kullanabilmesini sağlar.**
- **Sistem İşleyişi: Eğitilen model, kullanıcıların yüzlerini doğru şekilde tanıyacak şekilde çalışmalıdır. Eğitim verilerinin çeşitliliği ve kaliteyi arttıkça, yüz tanıma doğruluğu artar. Ayrıca, sistemin hızlı çalışması sağlanarak kullanıcı deneyimi iyileştirilir.**

7- Sistemin Çalışma Süresi ve Kesintisizlik:

- **Beklenen Sonuç: Sistem, her kullanıcı tanıdığında doğru şekilde çalışacak ve kesintisiz bir şekilde işlem yapacaktır.**
- **Sistem İşleyişi: Yüz tanıma işlemi sürekli olarak çalışır. Herhangi bir hata veya donma durumunda kullanıcı, sistemin doğru çalışmaması konusunda bilgilendirilmelidir.**

5. Teknik Çizimler ve Akış Şemaları :

1. Sistem Mimarisi Şeması

Bu diyagram, projenin donanım ve yazılım bileşenlerinin nasıl birbiriyle etkileşim içinde çalıştığını görselleştirir. Sistemin genel işleyişini anlamak için önemli bir araçtır.

Sistem Mimarisi Şeması:

- **Kamera: Yüz tespiti ve tanıma için görüntü alır.**
- **Raspberry Pi: Sistemin merkezi işlem birimi olarak çalışır, kamera verilerini işler, yüz tanıma algoritmalarını çalıştırır ve kapı kilidini açar.**
- **OpenCV ve LBPH Algoritması: Yüz tanıma işlemi burada yapılır.**
- **Kapı Kilidi: Tanımlanan kullanıcılar için açılır veya kapanır.**
- **Veritabanı/Label Dosyası: Kullanıcı etiketleri ve eğitilmiş yüz verileri burada saklanır.**

2. Akış Şeması

Akış şeması, sistemin nasıl çalıştığını adım adım gösterir. Bu diyagramda, her bir işlem ve bu işlemlerin nasıl sıralandığı belirtilir.

Akış Şeması:

1. **Kamera Başlatılır: Sistem çalıştırıldığında, kamera başlatılır ve görüntü akışı başlar.**
2. **Yüz Tespiti: OpenCV'nin Haar Cascade Classifier algoritması ile yüz tespiti yapılır.**
3. **Yüz Tanıma: Tanınan yüzler, daha önce eğitilmiş yüz veritabanı ile karşılaştırılır.**
4. **Tanımlama ve Kilit Kontrolü: Yüz tanındığında, veritabanındaki etiket ile eşleştirilir. Eğer yüz doğru tanınırsa, kapı kilidi açılır.**
5. **Tanıma Başarısızsa Kilitleme: Yüz tanınmazsa, kapı kilitli kalır.**
6. **Ekrana Bilgi Yazdırılır: Tanınan kişinin adı ve güven oranı ekrana yazdırılır.**
7. **Program Sonlandırma: Kullanıcı programı sonlandırmak istediğinde, sistem sonlanır ve kameranın kaynakları serbest bırakılır.**

3 .Yüz Tanıma Akış Diyagramı

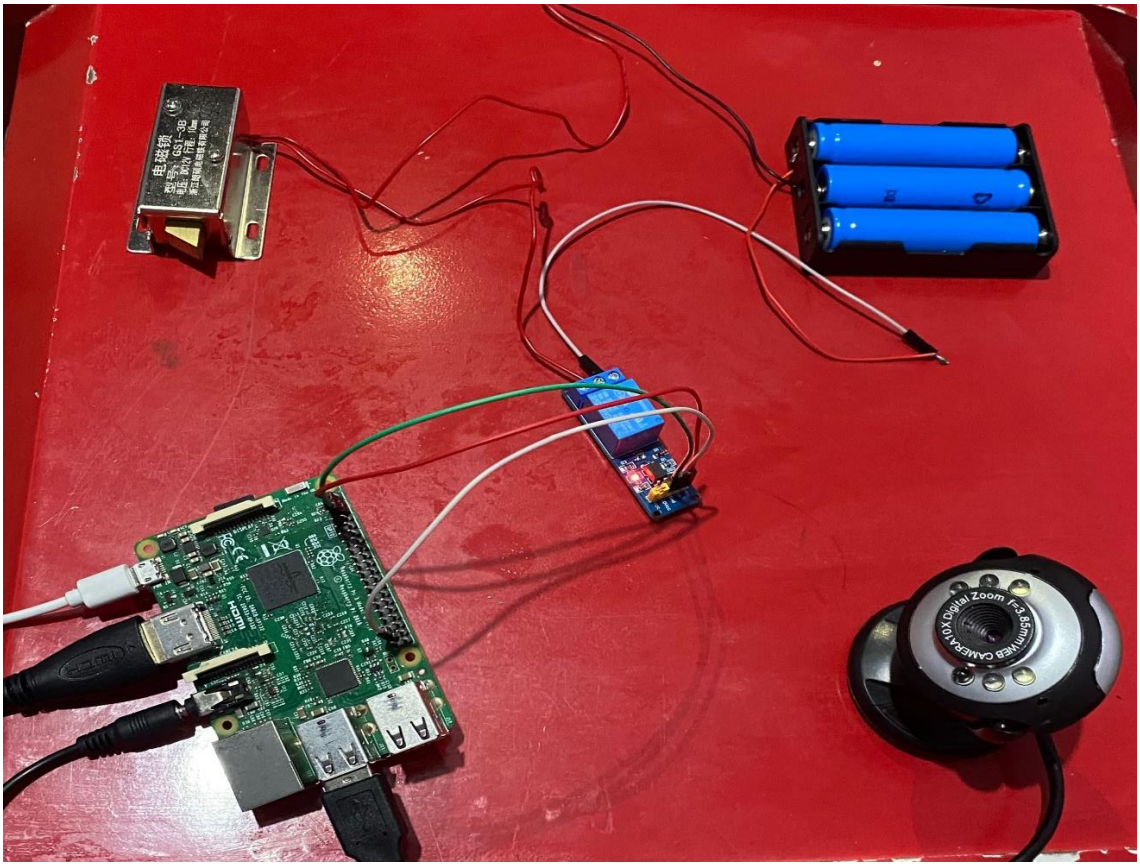
Bir başka önemli diyagram, yüz tanıma sürecinin adımlarını ve verilerin nasıl işlendiğini gösteren bir akış şemasıdır:

1. **Görüntü Alımı: Kamera aracılığıyla anlık görüntü alınır.**
2. **Görüntü İşleme: OpenCV kullanılarak, renkli görüntü gri tonlara dönüştürülür ve yüz tespiti için işlenir.**
3. **Yüz Tespiti: Haar Cascade Classifier ile yüzler tespit edilir.**
4. **Yüz Tanıma: Tespit edilen yüz, daha önce eğitilen veritabanındaki yüzler ile karşılaştırılır (LBPH algoritması).**
5. **Kişi Tanımlanması: Eğer tanıma başarılıysa, kişiye ait ad ve güven yüzdesi ekrana yazdırılır.**
6. **Kapı Kilidi Kontrolü: Tanıma sonucu doğru ise kapı açılır, aksi takdirde kilitli kalır.**

4 .Veri Akış Diyagramı

Bu diyagram, yüz verilerinin nasıl toplandığı ve işlendiğini açıklar:

- **Kamera: Yüz verisi (görüntü) alır.**
- **Yüz Tanıma Modülü: Görüntü işlenir, yüz tespiti yapılır ve veri tanımlanır.**
- **Veri Depolama: Tanınan yüzlerin etiketleri ve görüntü verileri veritabanında (pickle, trainer.yml) saklanır.**
- **Kapı Kilidi Sistemi: Tanımlanan kullanıcılar için kilit açılır veya kapanır.**



6.Sonuç ve Tartışma :

-Bu proje, Raspberry Pi tabanlı yüz tanıma sistemi ile fiziksel güvenlik sistemlerine entegre bir çözüm sunmayı amaçladı. Proje sürecinde kullanılan teknolojiler arasında OpenCV, NumPy, Dlib ve Pickle gibi önemli kütüphaneler yer aldı. Bu teknolojiler, yüz tanıma işlemlerinin yüksek doğrulukla gerçekleştirilmesini ve kullanıcıların güvenli bir şekilde tanımlanmasını sağladı.
Projede karşılaşılan başlıca zorluklardan biri, OpenCV kurulumuydu. İlk başta sanal ortamda OpenCV kurulumunu yapmayı tercih ettim, ancak bu süreçte bazı bağımlılık sorunları ve derleme hataları ile karşılaştım. Özellikle OpenCV'nin kurulumu çok uzun süre aldı ve 4 saatten fazla bir sürede tamamlanamıyordu. Bu, sanal ortamın izole yapısının, bazı gerekli bağımlılıkların düzgün bir şekilde yüklenmesini engellemesinden kaynaklanıyordu. Bu problemi aşmak için sanal ortamdan vazgeçerek, OpenCV'yi Raspberry Pi'nin ana işletim sistemine doğrudan kurdum. Bu çözüm, OpenCV'nin tüm gerekli bağımlılıkları ve derleme araçları ile uyumlu bir şekilde yüklenmesini sağladı ve sorun ortadan kalktı. Bu deneyim, sanal ortam kullanmanın bazı durumlarda yazılım geliştirme sürecini zorlaştırabileceğini ve doğrudan sistemde kurulumun daha verimli olabileceğini gösterdi.
Sonuç olarak, proje başarıyla tamamlandı ve yüz tanıma sistemi, yüzleri doğru bir şekilde tespit ederek kullanıcı doğrulama işlemlerini gerçekleştirebildi. Sistem, gerçek zamanlı video akışından yüzleri tespit etti, tanıdı ve güvenlik kilidini açma işlemini doğru bir şekilde gerçekleştirdi.
Bu sistemin potansiyeli büyük olup, benzer projelerde de kullanılabilir. Ancak, kullanılan teknolojiler ve donanım sınırlamaları doğrultusunda, sistemin doğruluk oranı daha da artırılabilir. Gelecekte, daha gelişmiş yüz tanıma algoritmalarının entegrasyonu ve sistemin hızlandırılması gibi iyileştirmelerle daha güvenli ve hızlı bir sistem elde edilebilir.

Özetle, bu proje yüz tanıma teknolojilerinin pratikteki kullanımını gözler önüne serdi ve karşılaşılan zorluklar sayesinde önemli deneyimler kazandırdı.

7. KAYNAKÇA :

1. **Youtube**
   https://youtu.be/wa6pBai8FhU?si=0_TKKP3jcaf8dCrE
2. **Githup**

   https://github.com/thekombite/facerecognition
3. **Chatgpt**

Ekip Adları Ve Okul Numaraları:

1-WAEL ASSAD 2022123143

2-RAMADAN ARAP 2022123155

3-AHMED SHERİF 2022123167

4-MUSTAFA ELBARRİ 2022123131

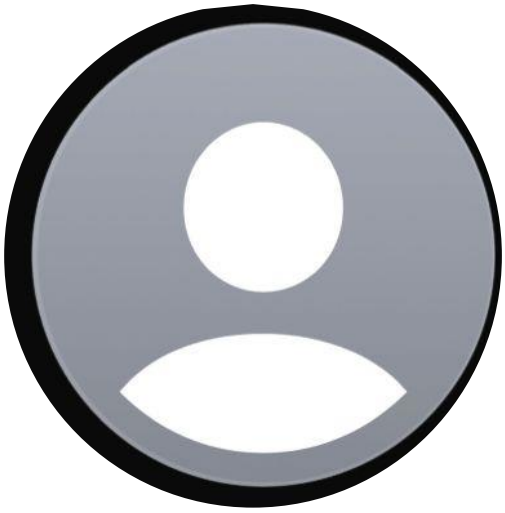
5-ELSHIMA ESMAIL 2022123141

ramadanarap30@gmail.com

a7med.shif0@gmail.com

Wailasad06@gmail.com

mustafaelbarri26@gmail.com



alshimaaahmed877@gmail.com