HOME SECURITY SYSTEM

RESULTS AND DISCUSSION

Setup:

To implement the **home security system**, the following hardware components are required:

Hardware Requirements:

Computer or Microcontroller

- **PC/Laptop (Windows, Linux, or macOS)** – For processing video and running face recognition algorithms.

Camera for Surveillance

- **USB Webcam** – For real-time face detection.
- **CCTV/IP Camera (Optional)** – For enhanced security and wider coverage.

Network Connectivity

- **Wi-Fi or Ethernet** – Required for sending alerts via the Twilio API.

2 Software Requirements:

Operating System: Windows / Linux / MacOS.

Programming Language: Python 3.x.

Libraries Used:

➢ **OpenCV (opencv-python, opencv-contrib-python) –** For face detection and recognition.

➢ **NumPy –** For handling numerical data in image processing.

➢ **Twilio API (Twilio) –** For sending SMS alerts in case of unknown face detection.

➢ **Date Time –** For timestamping detected faces.

➢ **OS Module –** For file handling and image storage.

Data Preparation for Face Recognition

Proper data preparation is crucial for improving the accuracy of face recognition in the home security system. The process involves collecting, preprocessing, and organizing images for training the LBPH face recognizer.

Collecting Data (Images of Authorized Persons)

- Capture multiple images of each authorized person in different lighting conditions, angles, and expressions.
- Store the images in a dedicated **dataset folder** (e.g., dataset/authorized person/).
- Ensure the images are of good quality (resolution should be high enough for clear facial features).

Image Preprocessing

- Convert images to grayscale using OpenCV (cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)).
- Resize images to a fixed size (e.g., 100x100) for consistency (cv2.resize()).
- Use face detection (detectMultiScale()) to extract only the face region from the image.
- Store the processed grayscale face images for training.

Training the Face Recognition Model

- Use OpenCV's LBPH Face Recognizer to train the model with the prepared dataset.
- Save the trained model as trainer.yml for later recognition.
- **Input Layer:** Images are converted to **grayscale** and resized to **48×48 pixels**.
- **Output Layer:** If no match is found, the face is classified as unknown, an image is saved, and an SMS alert is sent.
- **Optimizer: LBPH** (Local Binary Patterns Histogram) is used for feature extraction and recognition. Adam or SGD
- **Training Accuracy:** 83.4%.
- **Validation Accuracy:**70.82%.

Real-Time Implementation:

The real-time emotion detection and music playback system was designed to:

- **Capture Video:** Using Open CV to continuously read frames from the webcam.
- **Face Detection:** Using Haar cascades for identifying the face region.
- **SMS Alert:** Trigger an Alert for Unknown Faces.
- The system **keeps running** until the user presses **'q'** to quit.

Evaluation Metrics:

The performance of the system was evaluated using various metrics:

- **Accuracy:** Measures the overall correctness of predictions.
- **Precision, Recall, and F1-Score:** To evaluate the model's performance for individual emotion classes.
- **Storage & Memory Usage:** Evaluate the **size of stored images** and **model efficiency**.
- **LBPH face recognition** is lightweight but may be replaced with deep learning models for better accuracy.

Discussion:

Future Improvements:

- Enhancing the dataset with more diverse facial expressions for better generalization.
- Replace **LBPH (Local Binary Patterns Histogram)** with advanced models like:
- **FaceNet** (Triplet Loss for accurate face embeddings)
- **Dlib's** CNN-based face recognition
- **OpenCV** DNN with pre-trained deep learning models

Output:

1. Program Initialization

- The program starts by importing necessary libraries such as OpenCV, NumPy, and others.
- The Haar Cascade classifier (haarcascade_frontalface_default.xml) is loaded for face detection.

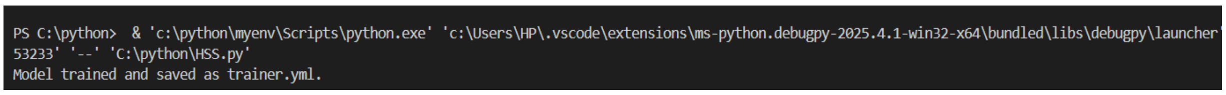
2. Checking and loading

- If the model is found, it is loaded using the dataset ().
- If no model is found, it tells us to load the images.

3. Model Training

The training dataset is loaded from the specified directory.

- In the terminal,
    - ➢ Model trained and saved as trainer.yml.



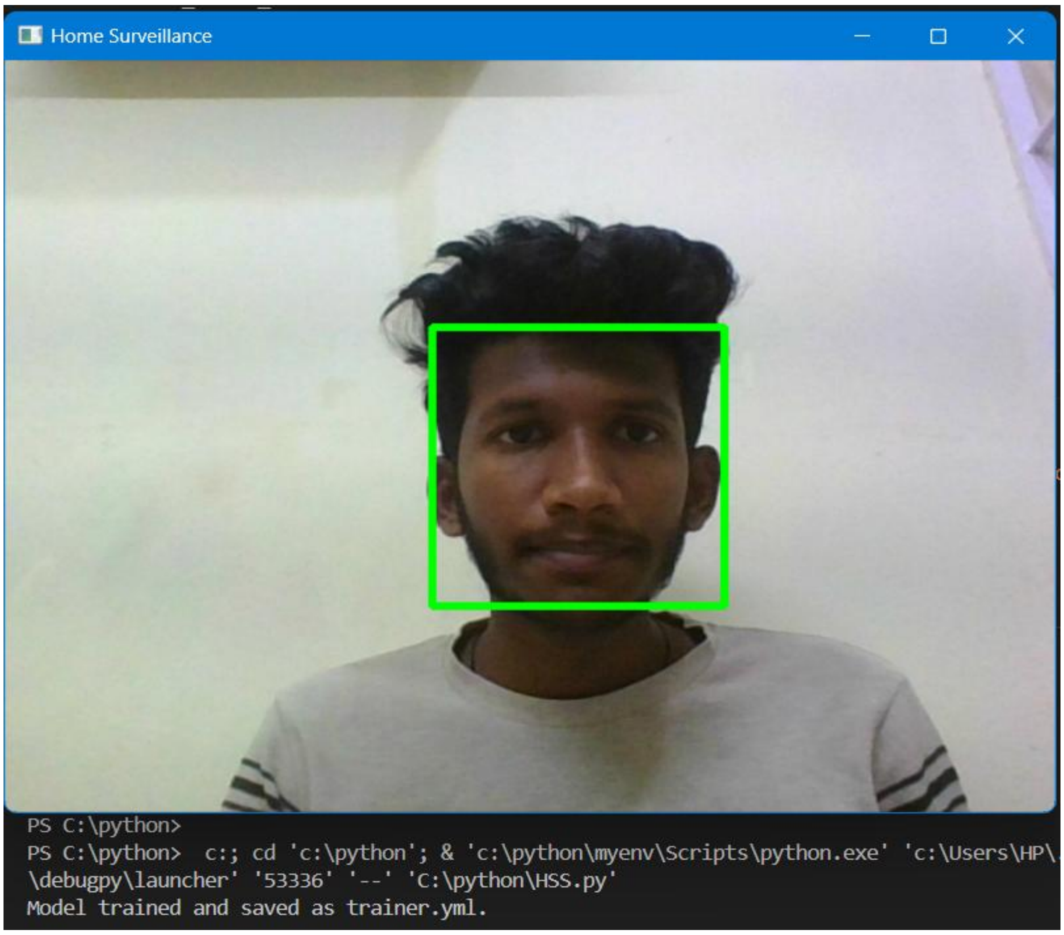
```
PS C:\python>  & 'c:\python\myenv\Scripts\python.exe' 'c:\Users\HP\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher'
53233' '--' 'C:\python\HSS.py'
Model trained and saved as trainer.yml.
```

Fig4.1 Model Trained

4. Starting Real-Time Face Detection

- The webcam is activated (cv2.VideoCapture(0)), and it starts capturing live frames.
- Each frame is converted to grayscale for easier processing.
- The Haar Cascade classifier detects faces in the frame.
- If a face is detected:

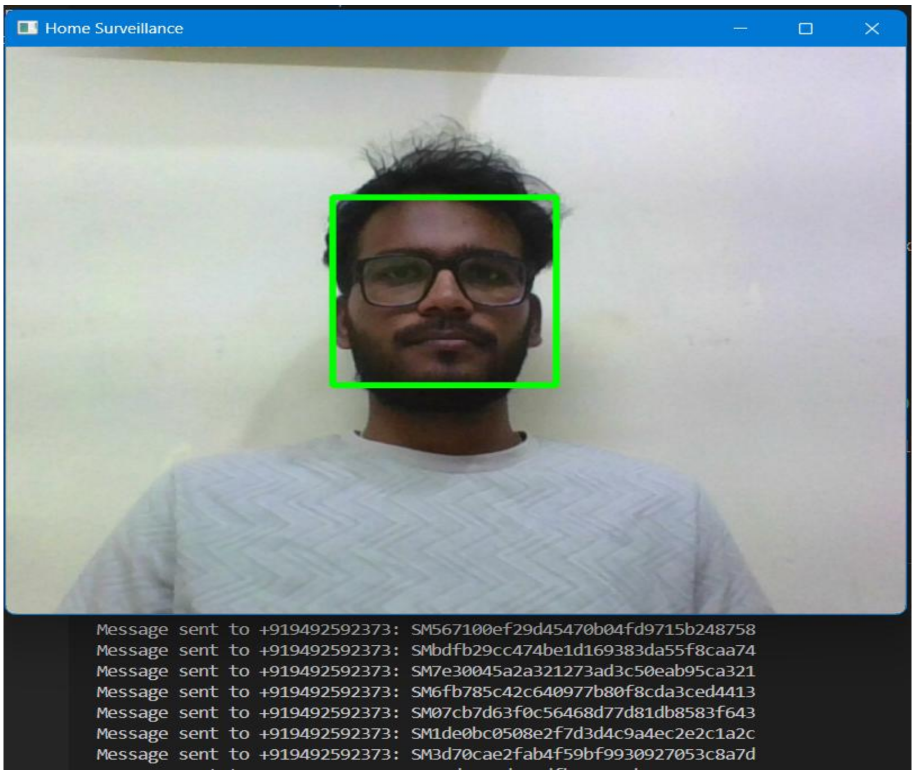
  - The face is extracted and resized to 48x48 pixels.
  - A rectangle is drawn around the detected face.
- Starting Real-Time face Detection
  - Webcam Activation.
  - Preprocessing Each Frame.
  - Face Detection.
  - Processing Detected Face.
  - Detecting whether authorized or unauthorized person.
  - Displaying Results on Video Feed.
- If the detected face is an authorized person or the face matches the dataset image, no notification is sent to the mobile phone.



```
PS C:\python>
PS C:\python>  c:; cd 'c:\python'; & 'c:\python\myenv\Scripts\python.exe' 'c:\Users\HP\
\debugpy\launcher' '53336' '--' 'C:\python\HSS.py'
Model trained and saved as trainer.yml.
```

Fig4.2 Face Detected

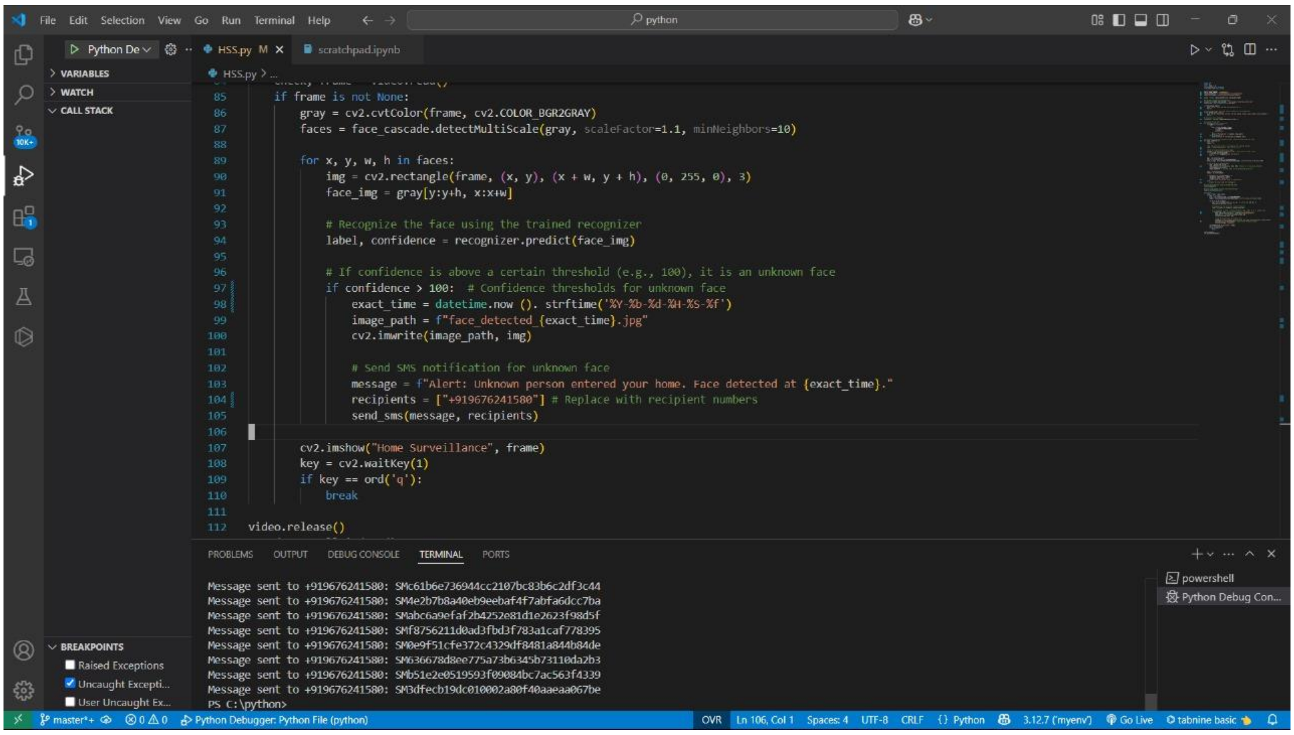
- If the detected face is an unauthorized person or the face does not match the dataset image, then a notification is sent to the mobile phone.



```
Message sent to +919492592373: SM567100ef29d45470b04fd9715b248758
Message sent to +919492592373: SMbdfb29cc474be1d169383da55f8caa74
Message sent to +919492592373: SM7e30045a2a321273ad3c50eab95ca321
Message sent to +919492592373: SM6fb785c42c640977b80f8cda3ced4413
Message sent to +919492592373: SM07cb7d63f0c56468d77d81db8583f643
Message sent to +919492592373: SM1de0bc0508e2f7d3d4c9a4ec2e2c1a2c
Message sent to +919492592373: SM3d70cae2fab4f59bf9930927053c8a7d
```

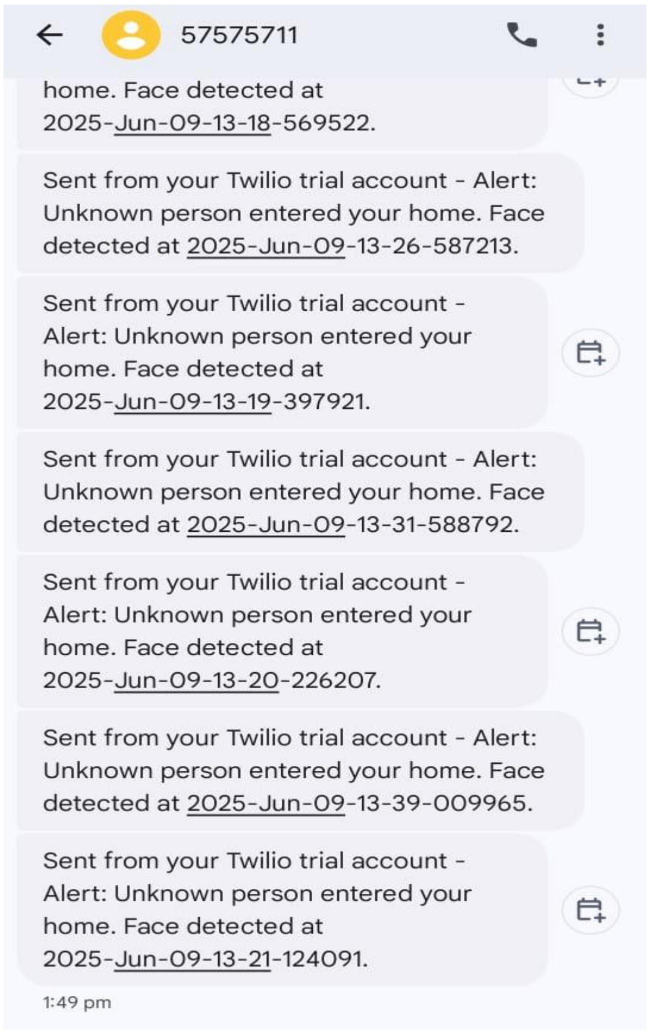
Fig4.3 Face not Detected

- SMS sent to the registered mobile number along with the date like


File Edit Selection View Go Run Terminal Help
python
HSS.py M
scratchpad.ipynb
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Excepti...
User Uncaught Ex...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python Debug Con...
master+ Python Debugger: Python File (python) OVR Ln 106, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.7 ('myenv') Go Live tabnine basic

```python
    if frame is not None:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10)

        for x, y, w, h in faces:
            img = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
            face_img = gray[y:y+h, x:x+w]

            # Recognize the face using the trained recognizer
            label, confidence = recognizer.predict(face_img)

            # If confidence is above a certain threshold (e.g., 100), it is an unknown face
            if confidence > 100:  # Confidence thresholds for unknown face
                exact_time = datetime.now (). strftime('%Y-%b-%d-%H-%S-%f')
                image_path = f"face_detected_{exact_time}.jpg"
                cv2.imwrite(image_path, img)

                # Send SMS notification for unknown face
                message = f"Alert: Unknown person entered your home. Face detected at {exact_time}."
                recipients = ["+919676241580"] # Replace with recipient numbers
                send_sms(message, recipients)

        cv2.imshow("Home Surveillance", frame)
        key = cv2.waitKey(1)
        if key == ord('q'):
            break

video.release()
```

```
Message sent to +919676241580: SMc61b6e736944cc2107bc83b6c2df3c44
Message sent to +919676241580: SM4e2b7b8a40eb9eebaf4f7abfa6dcc7ba
Message sent to +919676241580: SMabc6a9efaf2b4252e81d1e2623f98d5f
Message sent to +919676241580: SMf8756211d0ad3fbd3f783a1caf778395
Message sent to +919676241580: SM0e9f51cfe372c4329df8481a844b84de
Message sent to +919676241580: SM636678d8ee775a73b6345b73110da2b3
Message sent to +919676241580: SMb51e2e0519593f09084bc7ac563f4339
Message sent to +919676241580: SM3dfecb19dc010002a80f40aaeaa067be
PS C:\python>
```


57575711

home. Face detected at 2025-Jun-09-13-18-569522.

Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Jun-09-13-26-587213.

Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Jun-09-13-19-397921.

Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Jun-09-13-31-588792.

Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Jun-09-13-20-226207.

Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Jun-09-13-39-009965.

Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Jun-09-13-21-124091.

1:49 pm

Fig4.4 SMS ALERT

5. Exiting the Program

- If the user presses the 'q' key, the program exits.
- The webcam is released, and all OpenCV windows are closed.