Face Lock-Unlock System Using Facial Recognition and OpenCV

A Project report submitted in partial fulfilment of
the requirements for the course in *Deep Learning* in Computer Science
By
SHREE GOVIND JEE (2K22/CO/530)

Under the supervision of
Mr. Amit
Department of Computer Engineering

Department of Computer Engineering
DELHI TECHNOLOGICAL UNIVERSITY (Formally DEC)
Samayapur Badli, Daulatpur, Bawana Road, Rohini Delhi-110042

Abstract

Facial recognition has emerged as a crucial bio metric technique in modern-day security systems. This paper presents a Face Lock-Unlock System using Python and OpenCV, leveraging the Local Binary Pattern Histogram (LBPH) algo rhythm for face recognition and the Haarcascade classifier for face detection. The system captures facial data, trains a model, and performs real-time recognition to lock/unlock access based on authorized identities. The proposed approach provides a lightweight, offline, and user-friendly security solution.

Index Terms

- Face Recognition
- Biometric Security
- OpenCV
- LBPH
- Haarcascade
- Python
- Lock-Unlock System

I. INTRODUCTION

Security has become a paramount concern in both physical and digital domains. Traditional methods such as passwords or PINs are prone to security breaches. Biometric systems offer a more secure and convenient alternative. Among bio metric techniques, facial recognition is non-intrusive and user friendly. This paper proposes a facial recognition-based Lock Unlock System using Python and OpenCV, with real-time capabilities and minimal hardware requirements.

II. RELATED WORK

Several biometric systems have been proposed in recent years. Fingerprint recognition systems, iris scanners, and voice authentication are commonly used. Face recognition systems gained momentum due to advancements in machine learning and computer vision. OpenCV, with built-in face recognition modules, has been widely used in academic and industrial projects. Prior work includes face detection using Haar cascades and recognition using Eigenfaces, Fisherface, and LBPH. Our system adopts LBPH due to its efficiency and effectiveness in low-resource environments.

III. SYSTEM ARCHITECTURE

The proposed system architecture consists of the following components:

- **textbfData Collection**: Capture facial images using a webcam.
- **textbfPreprocessing**: Convert to grayscale and resize for consistency.
- **textbfModel Training**: Train an LBPH recognizer with labeled images.
- **textbfFace Detection**: Use Haarcascade to detect faces in real time.
- **textbfRecognition and Unlock**: Compare real-time input with trained data and grant access if recognized.

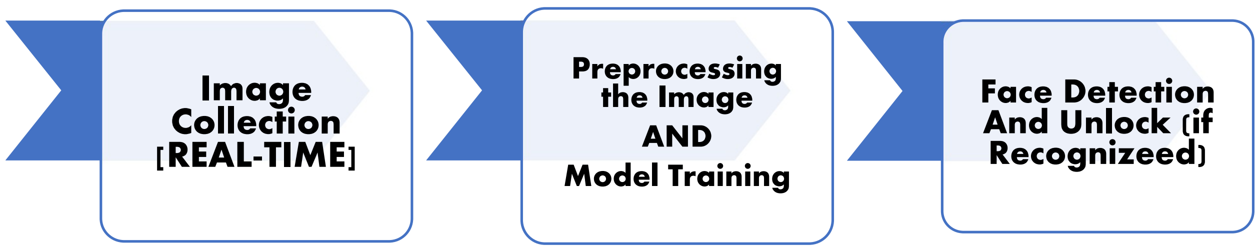


Image Collection [REAL-TIME]
Preprocessing the Image AND Model Training
Face Detection And Unlock (if Recognizeed)

Fig-1: -System architecture of the Face Lock-Unlock System

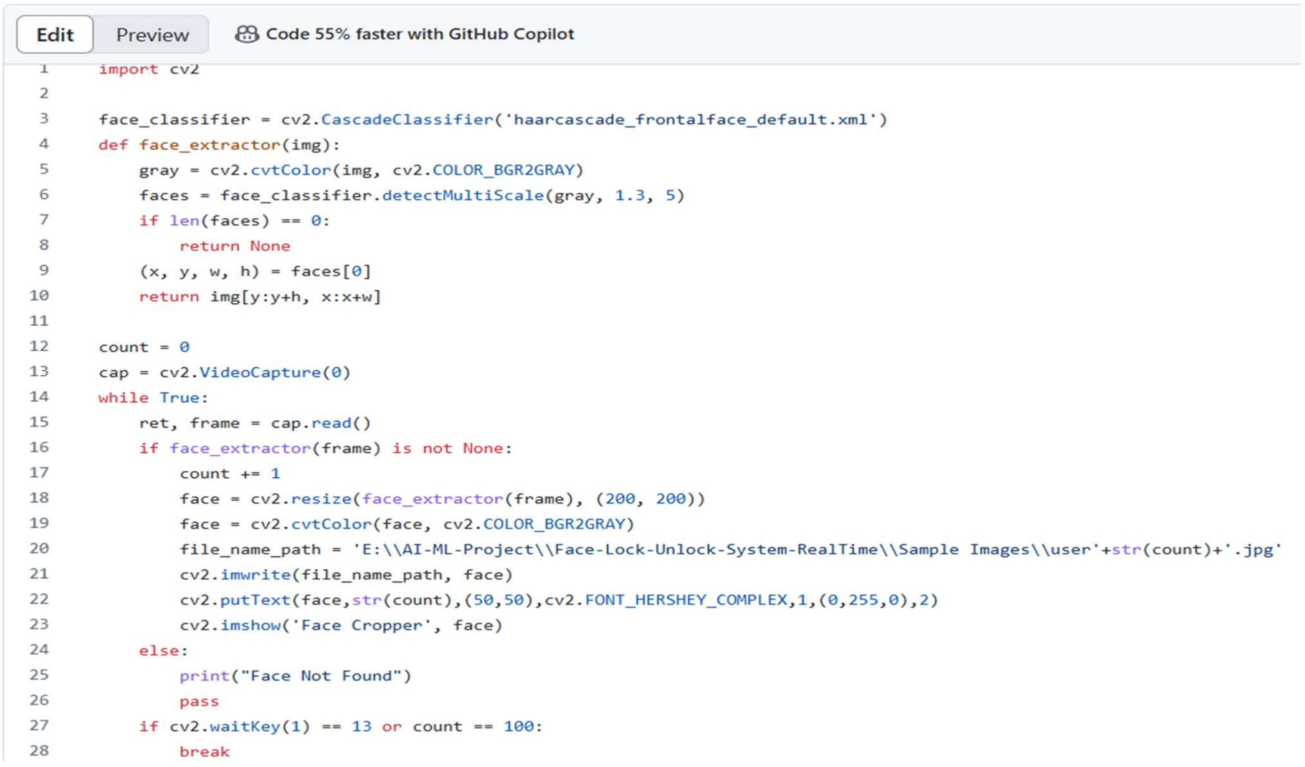
IV. IMPLEMENTATION

A. **Tools and Libraries**

i. Python 3.x
ii. OpenCV (cv2)
iii. **Haarcascade XML** for face detection
iv. **LBPHFaceRecognizer** from **OpenCV**

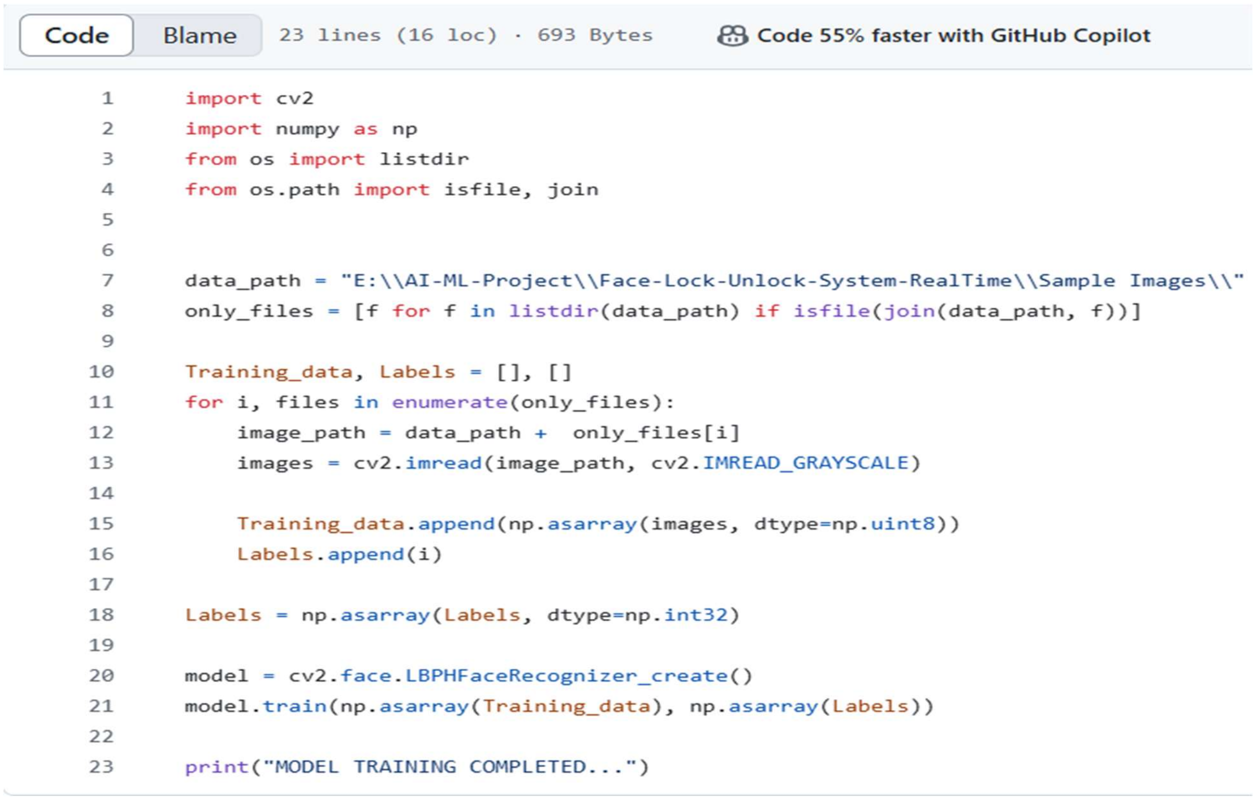
B. **Modules**

i. **dataset_creator.py**: Captures face data and stores images.



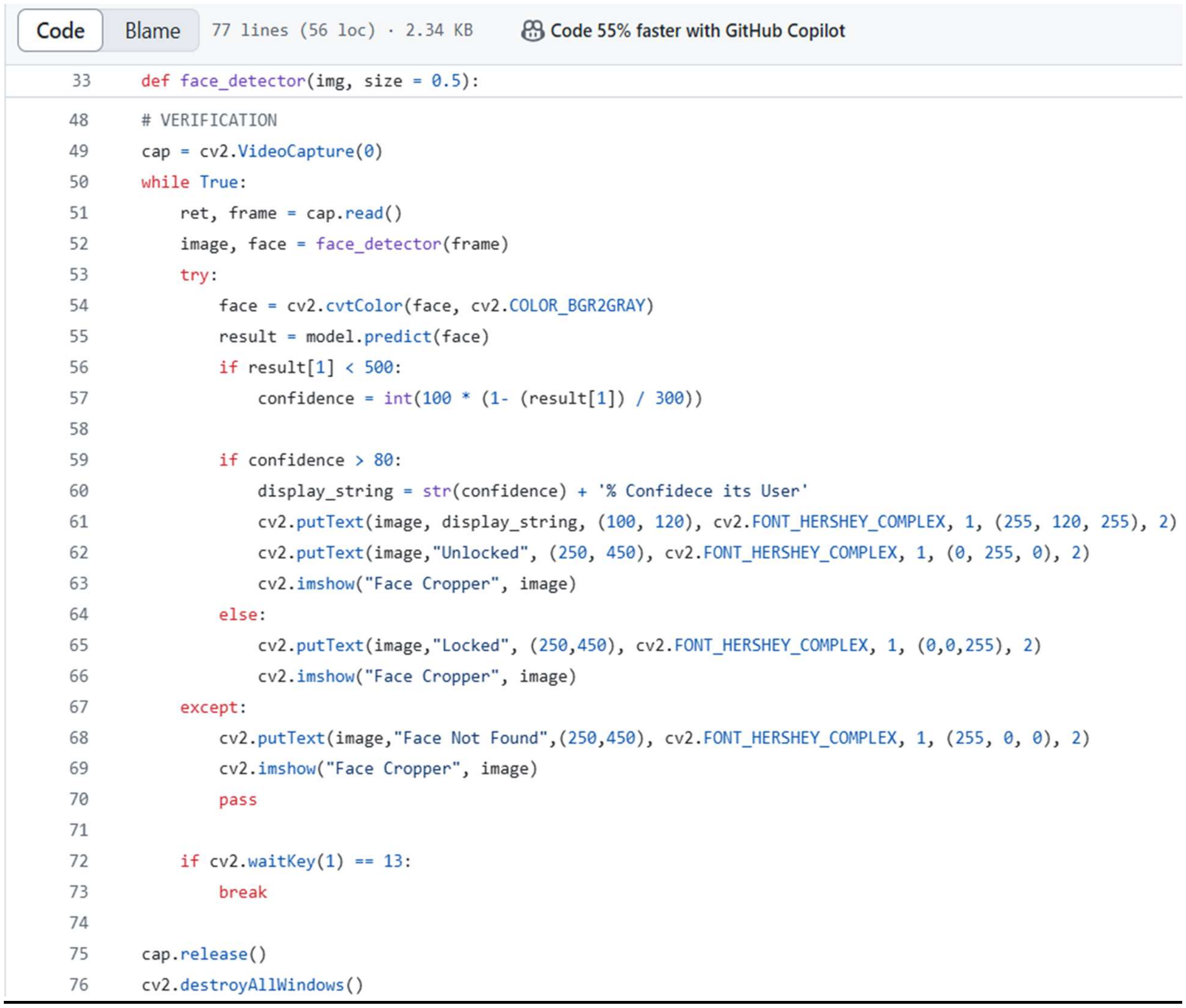
```python
import cv2

face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
def face_extractor(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    if len(faces) == 0:
        return None
    (x, y, w, h) = faces[0]
    return img[y:y+h, x:x+w]

count = 0
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame), (200, 200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        file_name_path = 'E:\\AI-ML-Project\\Face-Lock-Unlock-System-RealTime\\Sample Images\\user'+str(count)+'.jpg'
        cv2.imwrite(file_name_path, face)
        cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
        cv2.imshow('Face Cropper', face)
    else:
        print("Face Not Found")
        pass
    if cv2.waitKey(1) == 13 or count == 100:
        break
```

ii. **training_data.py:** Reads the dataset and trains the LBPH model.


Code
Blame
23 lines (16 loc) · 693 Bytes
Code 55% faster with GitHub Copilot

```python
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join


data_path = "E:\\AI-ML-Project\\Face-Lock-Unlock-System-RealTime\\Sample Images\\"
only_files = [f for f in listdir(data_path) if isfile(join(data_path, f))]

Training_data, Labels = [], []
for i, files in enumerate(only_files):
    image_path = data_path +  only_files[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    Training_data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)

model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(Training_data), np.asarray(Labels))

print("MODEL TRAINING COMPLETED...")
```

iii. **recognizer.py**: Loads the model and performs real-time recognition using webcam.


Code Blame 77 lines (56 loc) · 2.34 KB Code 55% faster with GitHub Copilot

```python
def face_detector(img, size = 0.5):

# VERIFICATION
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    image, face = face_detector(frame)
    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        result = model.predict(face)
        if result[1] < 500:
            confidence = int(100 * (1- (result[1]) / 300))

        if confidence > 80:
            display_string = str(confidence) + '% Confidece its User'
            cv2.putText(image, display_string, (100, 120), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 120, 255), 2)
            cv2.putText(image,"Unlocked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
            cv2.imshow("Face Cropper", image)
        else:
            cv2.putText(image,"Locked", (250,450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
            cv2.imshow("Face Cropper", image)
    except:
        cv2.putText(image,"Face Not Found",(250,450), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)
        cv2.imshow("Face Cropper", image)
        pass

    if cv2.waitKey(1) == 13:
        break

cap.release()
cv2.destroyAllWindows()
```

V. FACE RECOGNITION ALGORITHM

The LBPH algorithm works by summarizing the local structure of an image. The process involves:

- Comparing each pixel with its surrounding neighbors.
- Encoding results as a binary number.
- Creating histograms of binary patterns.
- Comparing histograms for recognition. LBPH is computationally efficient and works well in varying lighting conditions.

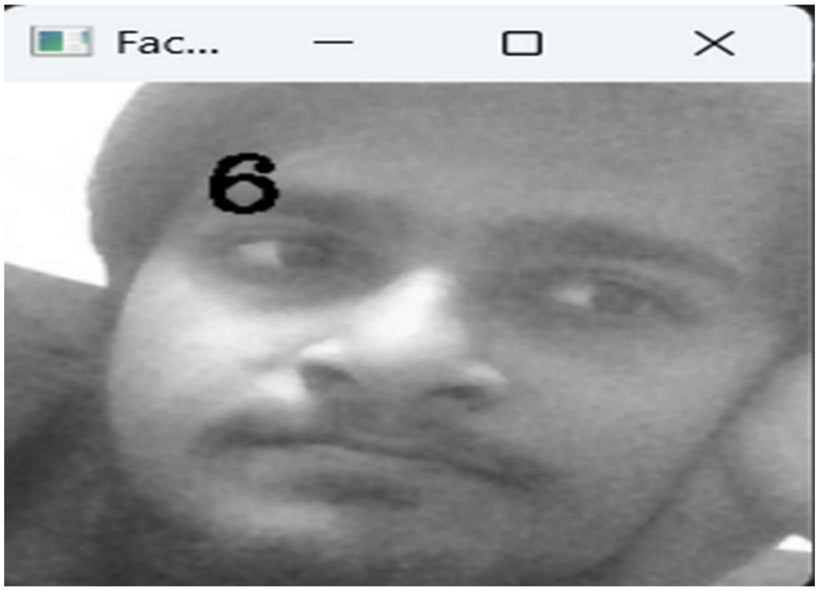


Fig-2: - Sample Face Collection

VI. RESULTS AND EVALUATION

The system was tested on a standard laptop with an integrated webcam. Results demonstrate:

- Successful face detection and recognition under normal lighting.
- Fast recognition (1s) and response time.
- **Accuracy above 90%** with sufficient training samples per user. Challenges include performance drops under poor lighting or occlusion.

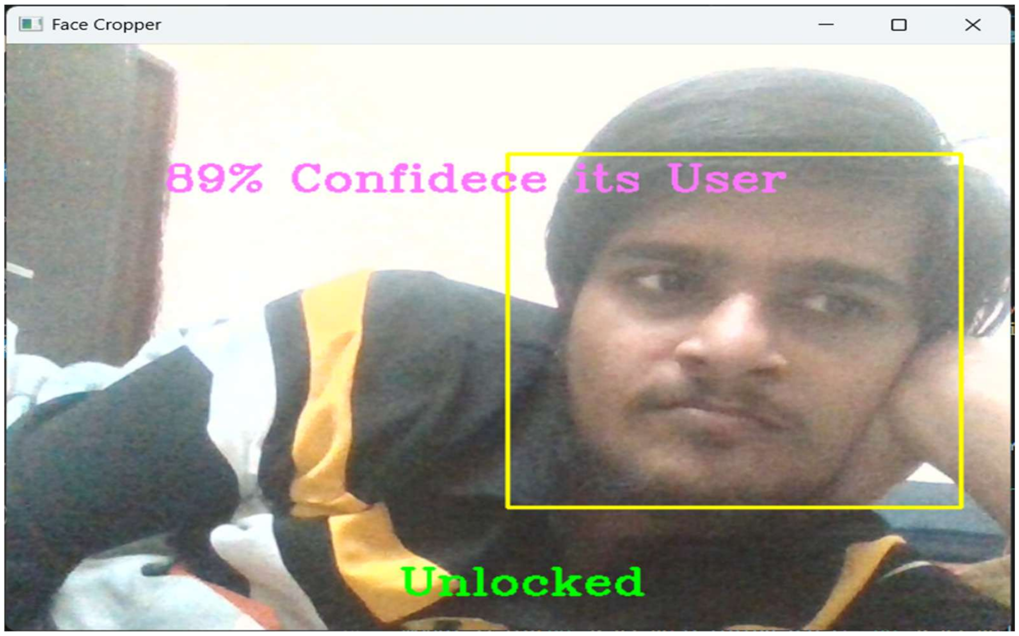


Fig-3: - Face detection and recognition sample

VII. LIMITATIONS AND FUTURE WORK

- Limited performance in low-light conditions.
- Susceptible to spoofing via printed images.
- Can be improved using deep learning (e.g., FaceNet, Dlib).
- GUI integration for usability.
- Hardware integration for IoT-based door lock systems.

VIII. CONCLUSION

This paper presents a simple and efficient facial recognition-based lock-unlock system using OpenCV. It achieves reliable results with minimal resources and can be extended for practical security applications. Future enhancements will focus on robustness and user-friendliness.

IX. ACKNOWLEDGMENT

The author would like to thank open-source contributors and the developers of OpenCV for enabling accessible computer vision research.

X. REFERENCES

[1] P. Viola and M. Jones, ”Rapid object detection using a boosted cascade of simple features,” Proceedings of the 2001 IEEE CVPR.
[2] G. Bradski, ”The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.
[3] T. Ahonen, A. Hadid, and M. Pietikainen, ”Face description with local binary patterns: Application to face recognition,” IEEE TPAMI, vol. 28, no. 12, pp. 2037-2041, 2006.
[4] OpenCV Documentation. [Online]. Available: https://docs.opencv.org