Facial Recognition System

Ben Hughes and Shiv Radhakrishnan

Northeastern University

EECE 2140. Computing Fundamentals for Engineers Fall 2024

Date: 11/22/2024

Abstract

This project focuses on building a facial recognition system using OpenCV's LBPH algorithm and integrating it into a 2-step security system. The system combines facial recognition with password verification to enhance security. Through this project, we learned how the LBPH algorithm works and improved our Python coding skills. While the system performed well, we faced some challenges, like issues with lighting and limited hardware capabilities. Despite these hurdles, the project successfully demonstrated how facial recognition can be used for secure authentication and highlighted areas for future improvement, like increasing accuracy and creating a better GUI.

Contents

Chapter 1

Introduction

1.1 Background

Facial recognition is a technology that can read and recognize faces by looking at facial features. It takes a set of sample images and uses a machine-learning algorithm to train a model to distinguish between different faces. With the right hardware and algorithm, this technology can be highly accurate. There are many real-life applications for this technology ranging from Apple Face ID to NSA surveillance at airports and other high-security places. With how accurate and effective facial recognition technology has become, it is a part of many peoples every everyday lives.

1.2 Problem Statement

Facial recognition technology has become an integral part of modern life, with applications ranging from personal device authentication to high-security surveillance. This project focuses on creating a facial recognition program using OpenCV's Local Binary Pattern Histogram algorithm while also gaining proficiency in Python coding. The goal is to integrate the facial recognition system into a 2-step security system that combines password verification and facial authentication to enhance security. The project is limited to using the LBPH algorithm, excluding alternatives like Eigenfaces or Fisherfaces, and focuses on facial recognition, leaving out other biometric techniques. By narrowing the scope, the project aims to deliver a robust and focused implementation of facial recognition within a defined security system.

1.3 Objectives

The goal of this project is to develop a facial recognition program using OpenCV's Facial Recognition Library. To do this, we wanted to work and become more comfortable with coding in Python. We also wanted to gain a deeper understanding of the Local Binary Pattern Histogram algorithm that is used in OpenCV's library. Fully understanding how the algorithm works is crucial to developing a facial recognition program. We also wanted to use the program and integrate it into a 2-step security system, which would verify the identity of the user using a password and facial recognition.

1.4 Scope

This project covers using OpenCV's Local Binary Pattern Histogram Algorithm to implement a facial recognition program that is integrated into a 2-step Security system. This project delivers on the making of both facial recognition and security systems. This project focuses on facial recognition only and does not cover other forms of biometric identification techniques. This project also only uses the LBPH algorithm supposed to other forms of facial recognition such as Eigenfaces or Fisherfaces. We choose to use this form of biometric identification given the technology we have on hand and this algorithm due to its convenience with OpenCV's library.

Chapter 2

Technical Approaches and Code UML

2.1 Development Environment

We used the Python version 3.12.7. For our development environment, we chose to use Visual Studio Code. We chose VS code because it's a relatively simple yet powerful IDE. Within our code, we imported a number of libraries to help make our code easier to write. The first library we imported was OpenCV's library to connect to our webcam, capture/perforate frames, and ultimately identify faces using their built-in functions/classes as well as the contrib module functions/classes. For our GUI, we imported the TK interface library to create a simple UI. TKinter is rather old but allows us to create a UI quickly and easily. Then, we imported Json which is a library that allowed us to parse the Json file containing the userID names. Finally, we imported numpy which helped OpenCV handle the image data. Another tool we used in creating our project was the "adaboost frontal face detector," which we took in as an xml file titled "haarcascade_frontalface_default," this is a xml file created by openCV that allows our system to detect/recognize faces.

2.2 Data Collection and Preparation

Our data collection began with our taker and trainer classes. The taker class took a userID, which is put into a json file, then took 30 images/frames of the user. After this, it used openCVs functions to detect and crop the image to the face and converted it to grayscale so it

could be better used by the recognizer. After the taker class has taken in all the images, we move onto the trainer class. The trainer class takes in the photos generated by the taker class and creates and trains a facial recognition model to be used in our main class (“trainer.yml”). When the main class calls upon the created trainer, it does so by comparing the face from the frame to the saved one in the saved facial recognition model.

2.3 Implementation Details

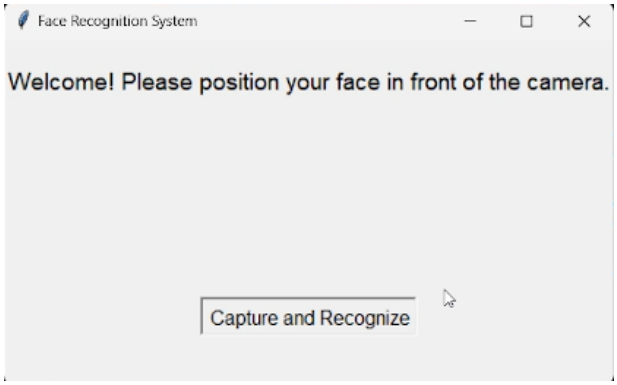
The code begins through the taker and trainer class described in 2.2. After this, it creates/initializes the window that holds the introduction message and capture button. When the user clicks on the capture button, it runs the “capture_and_identify” function which captures a frame, perforates it/finds the face for the trainer, associates a name and confidence level with the face and then recognizes the user and prints out a message identifying the user along with the image taken. Then the function waits 5 seconds before running the “show_password_entry” function that changes the label to “Enter your password below: ” and requests a password input. This password input then checks to make sure the password and userID line up, and if so, it prints out a success notification.

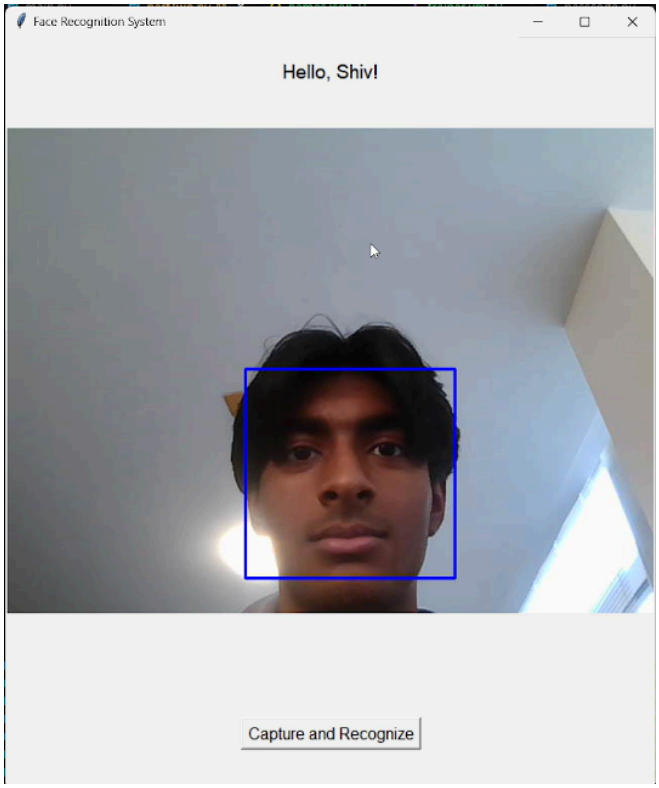
Chapter 3

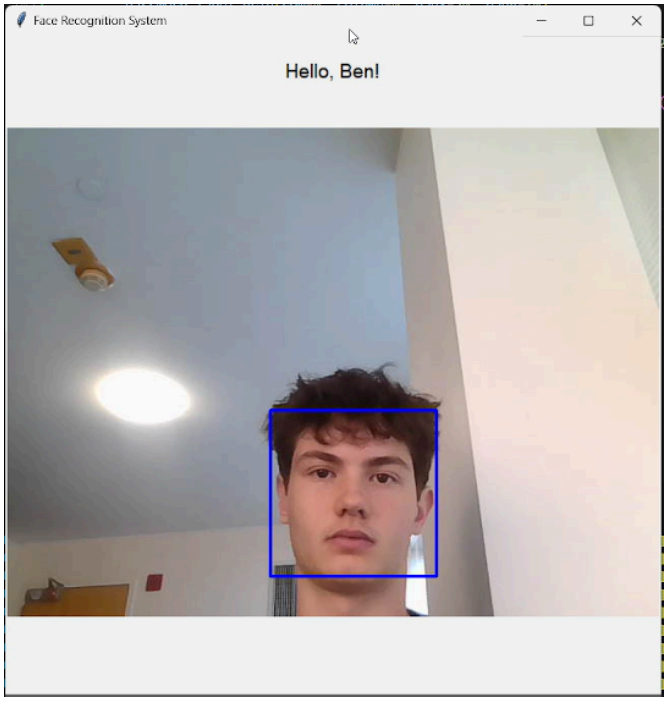
Project Demonstration

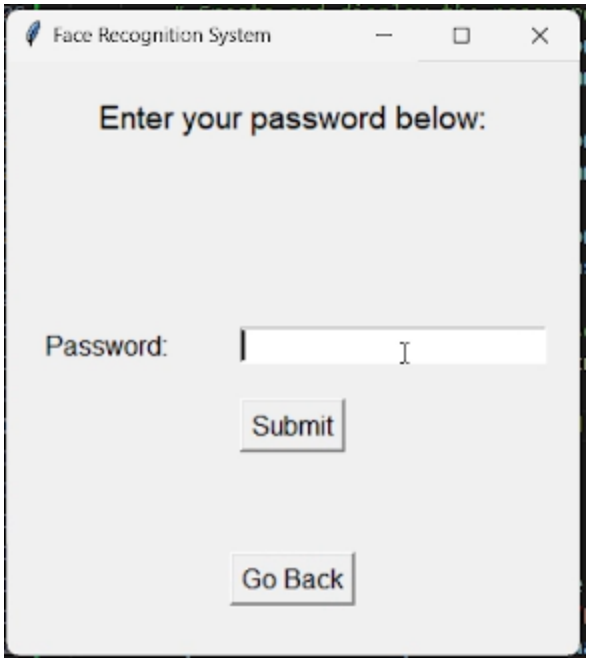
3.1 Screenshots and Code Snippets

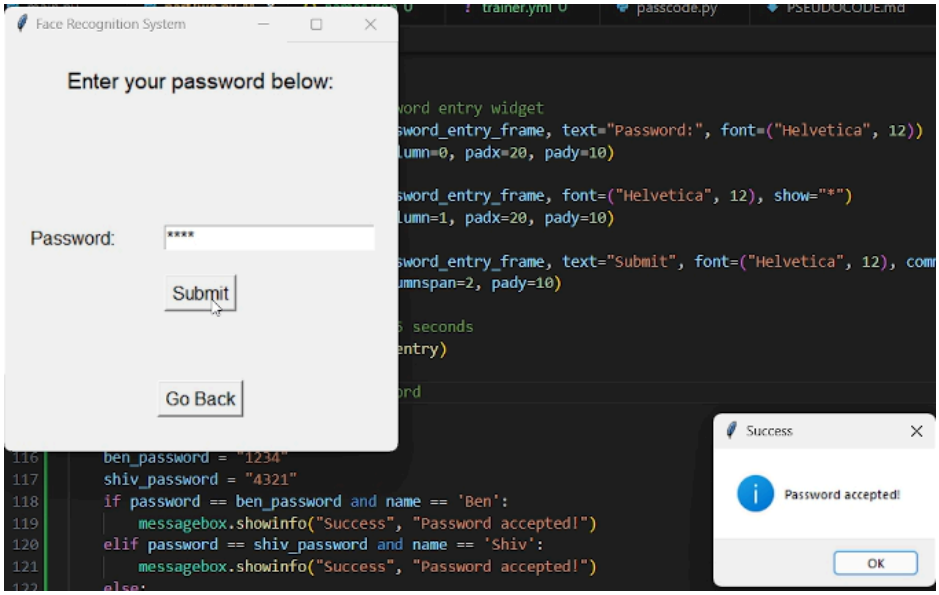
Include visual evidence of the project's outputs here. Add screenshots and code snippets as needed.




Face Recognition System
Hello, Shiv!
Capture and Recognize


Face Recognition System
Hello, Ben!


Face Recognition System
Enter your password below:
Password:
Submit
Go Back


Face Recognition System
Enter your password below:
Password:
****
Submit
Go Back
ben_password = "1234"
shiv_password = "4321"
if password == ben_password and name == 'Ben':
messagebox.showinfo("Success", "Password accepted!")
elif password == shiv_password and name == 'Shiv':
messagebox.showinfo("Success", "Password accepted!")
Success
Password accepted!
OK

Chapter 4

Discussion and Future Work

Looking at our results, we successfully developed a working facial recognition program. The taker and trainer classes effectively captured our faces and trained the model. However, integrating the program into our security system revealed some challenges. During our research on the LBPH algorithm, we found that converting images to grayscale was crucial for recognizing faces in different lighting conditions. Despite this, the program sometimes struggled to recognize faces when the lighting during recognition was significantly different from the lighting during sample collection. While this issue arose in specific cases, the program generally performed well and integrated smoothly with the security system.

This project's findings highlight the potential for enhanced security systems through the addition of multiple layers of identification. Facial recognition technology has numerous real-life applications, such as Face ID, and can be scaled for larger high-security environments.

We encountered some limitations related to technology and time. Algorithm accuracy was constrained by the limited number of samples, as our computers lacked the processing power to handle a larger dataset. More samples would have allowed the trainer to gather more accurate facial feature data, improving recognition confidence. Additionally, time constraints

meant that most of our focus was on the facial recognition aspect, leaving less time to refine the security system's graphical user interface (GUI), which remained simple.

For future work, we suggest exploring other facial recognition algorithms to assess potential improvements in accuracy. If continuing with the LBPH algorithm, using more powerful hardware to collect and process a larger dataset would be beneficial. Furthermore, the UI could be enhanced using a different Python GUI library to create a more unique and user-friendly experience.

Chapter 5
Conclusion

This project successfully implemented a facial recognition program using OpenCV's Local Binary Pattern Histogram (LBPH) algorithm and integrated it into a 2-step security system combining facial recognition and password verification. Through this process, we gained a deeper understanding of the LBPH algorithm, improved our Python skills, and showed the potential of facial recognition as a security mechanism.

The system effectively trained and recognized faces under most conditions, showing these tools uses in real-world scenarios such as identity verification and enhanced security systems. However, limitations such as lighting inconsistencies, sample size data, and hardware constraints should be noted as areas for improvement. The basic functionality of the security system was achieved, the GUI could benefit from further changes to improve appearance/aesthetic.

Future work should explore alternative algorithms to enhance accuracy under varying conditions, ways to take the project into a real life scenario, and develop a more aesthetic interface. Despite these limitations, the project demonstrated the practicality and efficiency of facial recognition technology.