Smart Home Security: Keeping Intruders at bay with Your Face

Name: Georghios Tzioulliou(2412649)

Advisor: Mr. ‘Femi (Oluwafemi) Samuel

Aim and Objectives

This project proposes developing a home access control system that uses facial recognition technology with existing security systems. By combining facial recognition with a traditional access control method, we can create a more secure and user-friendly solution for managing home entry.

Research

Due to security concerns, no product has been developed that will unlock your house with just your face identification from what I found out, but similar products exist like unlocking your phone using a face ID. A lot of countries use face recognition to track people in public e.g. identifying criminals, understanding facial expressions, ethnic backgrounds of a person, etc… Though projects exist that come close to what my project is about, a student from the university Jaypee University of Information Technology Waknaghat did a study on Face Detection and Recognition which they talk about the problems encountered and what they used. Additional research was done on the libraries and from what I gathered regarding the speed and accuracy OpenCV and DLib in combination can create an accurate and fast system. From the article posted on Medium for “What’s the Best Face Detector?” We can see that the DLib library is one of the fastest since it can use a GPU to load the model and has high accuracy.
From what I gathered from all my research regarding face recognition libraries it led me to choose OpenCV and DLib for this project while taking into consideration the price for the specs required the accuracy and as well as the speed. Dlib performs well with GPU acceleration and OpenCV complements this by providing robust image-processing functions, which enhance pre-processing steps before running recognition models. By leveraging these tools, my approach aims to create a fast and reliable system while maintaining a high recognition accuracy rate.

Summary to date

Up to now, I managed to get the core functionality of the project to work, it can train a model in a reasonable amount of time (approx... 1-2 min for 3 people 100 pictures each), it's able to identify any faces as long they are clear up to an extent and finally, it can identify the faces from the trained model. It’s also able to do real-time face recognition using a camera, if it detects a face using OpenCV it changes the image captured to a greyscale and tries to find a correlation between the face it captured and the photos the model was trained on. If the similarity is above 50% it will display the name of the model thinks it is. **Due to some issues with previous datasets, I tried to**

use I am using celebrity faces just for testing purposes since they are easily accessible and easy to test.

The main stumbling blocks I hit so far were firstly the dataset I tried to use, after looking at countless datasets I didn't find anything that still suited my neat as to why I used celebrity faces for testing purposes just to see how my code works and if it does. Secondly, it was the time it took when I tried to train one of my first models it took an hour to train on 100 images and the accuracy was bad. The final issue I have is getting the Raspberry Pi5 to work, When I try to install an operating system on it for one reason the monitor just turns black, and all the progress is lost which I have yet to figure out why.

Personal Reflection

Looking back to where I wanted to be at this stage and where I am, is that I am a bit behind, especially on everything that's not code-related. Due to some sickness that lasted throughout December, I had no energy to put the effort I wanted into this. Luckily, I caught up a bit now that am feeling better and don't have any responsibilities to attend to, I feel am on the right track and happy that I can see my core program working as intended but not perfectly.

Plans for Remainder of the Project

Firstly, I would like to fix my Requirements specification since some mistakes were pointed out in the last version I wrote. Following that I would create a few diagrams, like an activity diagram to show the interaction between the user and the system and a class diagram to show what happens in the background when the user interacts with the system. Alongside, I would also start writing the report and some literature on research that was done on similar projects. Regarding the programming side, I would like to explore a different model that will be able to recognize the liveness of a person's face since it's a more appropriate approach for the concept of my project. I would also like to create a website where I would be able to control the system without being there, look at live feed, unlock/lock the door, etc. Finally, I will start doing the poster and preparing for the presentation. See figure 1.1 for a Gantt chart.

Face Recognition Libraries & Models

- DLib: Used for face detection and recognition due to its speed and high accuracy, especially with GPU acceleration.
- OpenCV: Utilized for image processing and handling video streams.
- Shape Predictor (DLib's shape_predictor_68_face_landmarks.dat): Helps detect facial landmarks to improve recognition accuracy.
- Face Recognition Model (DLib's dlib_face_recognition_resnet_model_v1.dat): A ResNet-based face embedding model for feature extraction.
- Alternatives Considered: OpenCV's built-in face recognition (but DLib was chosen for better performance).

Machine Learning Models for Classification

- K-Nearest Neighbors (KNN) Classifier
- Used to classify face embeddings from DLib.
- Explored confidence scores by measuring the distance to the nearest neighbor.

Model Training & Storage

- Training Data Handling
- Loaded and processed images from a dataset.
- Applied train-test split (80-20) using sklearn.model_selection.train_test_split.
- Handled cases where faces were not detected in images.

Model Storage & Loading

- Pickle was used to save and load the trained model.
- Label Dictionary Storage: Mapped labels (names) to numerical values
- Explored different data serialization options for efficiency.

Real-time Face Recognition

- Live Camera Feed via OpenCV
- Performed real-time face detection and recognition.
- Displayed confidence scores and dynamically adjusted the label (showing “Other” when confidence was low).
- Alternatives Considered: YOLO or MobileNet-based face detectors for real-time efficiency.

Performance & Optimization Considerations

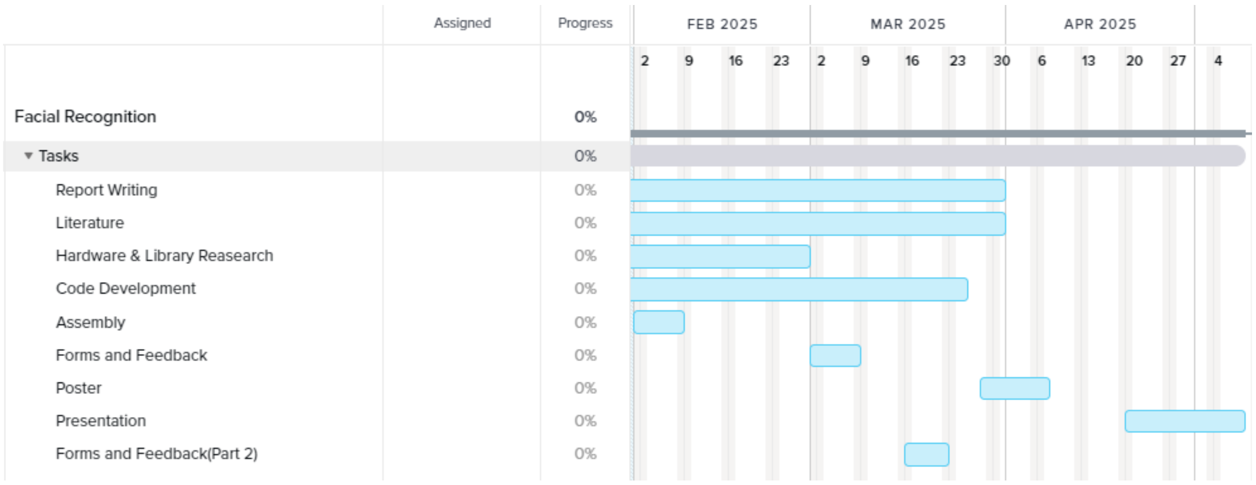
- GPU Acceleration: Explored DLib’s ability to use CUDA for faster model inference.
- Progress Bars (tqdm): Added to improve user experience during training/testing.

Error Handling & Warnings:

- Skipped images where no faces were detected.
- Displayed warnings for low-confidence predictions.

Appendix:

Gantt Chart 1.1



	Assigned	Progress
Facial Recognition		0%
▾ Tasks		0%
Report Writing		0%
Literature		0%
Hardware & Library Reasearch		0%
Code Development		0%
Assembly		0%
Forms and Feedback		0%
Poster		0%
Presentation		0%
Forms and Feedback(Part 2)		0%

Reference:

http://ir.juit.ac.in:8080/jspui/bitstream/123456789/6301/1/Face%20Detection%20and%20Recognition%20System%20by%20Shivam%20Bhargava.pdf

https://medium.com/pythons-gurus/what-is-the-best-face-detector-ab650d8c1225