

A Project Report on Real-time Facial Recognition



Information System Security

Carl Vincent Saban, Monte Kang, Jun Wang

2020.04

Table of Content

1. Overview of the Project	3
1.1 Project Name: Facial Recognition system	3
1.2 Project Period: Jan.06.2020 ~ Apr.10.2020	3
1.3 Group Member and roles	3
1.4 Project Summary	3
1.5 Project Goals	3
1.6 Project Achievement	3
2. Facial Recognition Technology	4
2.1 Identification	4
2.2 Verification	5
3. Face Recognition Deep Learning Models	6
3.1 OpenCV	6
3.2 HoG Face Detector in Dlib (face_recognition)	7
4. Project Details	8
4.1 Operational and Environmental Requirements	8
4.2 Main structure and functions	8
4.3 Main Functions	10
4.3 How to Use	13
5. Problems and Solutions	15
5.1 OpenCV works on a specific environment.	15
5.2 Choosing the face recognition module.	
5.3 The face recognition process pauses when sending alerts.	
6. Improvement of the Project	16
7. Conclusion	17
8. Reference	18

1. Overview of the Project

1.1 Project Name

Facial Recognition system

1.2 Project Period

Jan.06.2020 ~ Apr.10.2020

1.3 Group Member and roles

we all put out equal efforts in researching, developing and documenting for the project.

- Carl Vincent Saban
- Modae Kang
- Jun Wang

1.4 Project Summary

Build a facial recognition system with IFF functionality (system is able to differentiate between friends and foes), it scans faces of all users who access the system and add them to a database (only new faces should be added to the database), should have logging system (who accesses the system, times, new records in the database), and finally it should alert the user when a “foe” is detected.

Build a two way authentication system: using username/password as the first way of authentication and biometric facial feature as the second way of authentication.

1.5 Project Goals

The main goal of the project was to make a facial recognition system that can be run in a restricted room or area for surveillance and recording the information of accessed personnel. Develop a 2 way authentication system with functions adding new users and verify users' credentials.

- Accuracy:

To achieve the necessary level of accuracy of facial recognition is challenging. Our goal is to distinguish the identity of who is exposed to the surveillance camera. The model utilized in this project has an accuracy rate of 99.38%.

- Timing:

Our goal is to achieve real time processing of living streams captured by camera.

- Security:

To avoid SQL injection attack, users input should be sanitized for the 2 way authentication system. Passwords should be hashed before they are stored in the database.

1.6 Project Achievement

Developed a face recognition system, with the function of automatically creating database tables and saving face features into the database. The face_recognition model utilized in this project has an accuracy rate of 99.38%. The python opencv model is able to capture 30 frames every second. Our facial recognition processing speed reaches 6 frames every second. For the 2 way authentication system, the input is properly sanitized and tested SQL injection proof. Passwords are properly hashed by sha512.

2. Facial Recognition Technology

Facial recognition technology (FRT) has emerged as an attractive solution to address many contemporary needs for identification and the verification of identity claims. It brings together the promise of other biometric systems, which attempt to tie identity to individually distinctive features of the body, and the more familiar functionality of visual surveillance systems.

Facial recognition technology (FRT) allows the automatic identification of an individual by matching two or more faces from digital images. It does this by detecting and measuring various facial features, extracting these from the image and, in a second step, comparing them with features taken from other faces.

Facial recognition refers to a multitude of technologies that can perform different tasks for different purposes. In this regard, a key distinction is whether facial recognition is used for verification or identification.

2.1 Identification

Identification means that the template of a person's facial image is compared to many other templates stored in a database to find out if his or her image is stored there. Facial recognition can help verify personal identity. It is a one to many comparison. If the person is not present in the database, it means we have never seen this person before. So you may either add them to the database or simply say that he/she is an unknown person. This can be used in applications like Surveillance, Attendance system etc.

The facial recognition technology returns a score for each comparison indicating the likelihood that two images refer to the same person. Sometimes images are checked against databases, where it is known that the reference person is in the database (closed-set identification), and sometimes, where this is not known (open-set identification). The latter operation would be applied when persons are checked against watchlists.

In this project, our real-time face recognition system (`face_surveillance.py`) is an example of utilizing identification technology. Identification is used based on facial images obtained from video cameras. For this purpose, the system first needs to detect if there is a face on the video footage. Faces on video footage are extracted and then compared against the features of facial images in the reference database to identify whether the person on the video footage is in the database of images.

2.2 Verification

Unlike identification which performs a 1:n match against a database of known faces, verification or authentication is often referred to as one-to-one matching. It enables the comparison of two biometric templates, usually assumed to belong to the same individual. Two biometric templates are compared to determine if the person shown on the two images is the same person. The facial recognition technology compares the two facial images and if the likelihood that the two images show the same person is above a certain threshold, the identity is verified. Verification does not demand that the biometric features be deposited in a

central data-base. They may be stored, for example, on a card or in an identity/travel document of an individual.

Verification technology forms the base of the second authentication method of our 2 way authentication system. Facial Authentication is a form of biometric authentication that relies on the unique biological characteristics of an individual to verify that she is who she claims to be. The users are authenticated by verifying their faces as one credential to securely access their accounts.

3. Face Recognition Deep Learning Models

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep Learning is Hierarchical Feature Learning. Deep learning models have the ability to perform automatic feature extraction from raw data, also called feature learning.

3.1 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions for real-time computer vision. The face detection part of the project was made using an OpenCV Library and python model face-recognition. The reason was that most Face APIs are restricted to doing detection on pictures only, the project was required to have face recognition done on a live video to speed up the process and achieve real-time face recognition.

The OpenCV library proved to be flexible enough for the project as it can accurately detect a face in real time and highlight it by drawing a rectangle around the faces passing by. While faces are being detected, the application analyzes every video frame for face recognition.

Advantages:

- works almost real-time on CPU.
- Simple Architecture.
- Detects faces at different scales.

Disadvantages:

- The major drawback of this method is that it gives a lot of False predictions.
- Doesn't work on non-frontal images.
- Doesn't work under occlusion.



Figure 1. detect human face from the image using OpenCV

3.2 HoG Face Detector in Dlib (face_recognition)

The face_recognition module was built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

The face_recognition model is based on pre-trained models, meaning that the software develops rules for identification of faces based on a database of facial images.

This was possible through the increase in the availability of facial images at higher quality and the increase in computing power to process large amounts of data.

The face recognition model is trained on adults and does not work very well on children. It tends to mix up children quite easy using the default comparison threshold of 0.6.

Advantages:

- Works very well for frontal and slightly non-frontal faces
- Works under small occlusion
- Easy to implement

Disadvantages:

- It does not detect small faces as it is trained for a minimum face size of 80×80 .
- The bounding box often excludes part of forehead and even part of chin.
- Does not work for side face and extreme non-frontal faces, like looking down or up.



Figure 2. detect human face from the image using HoG Face Detector in Dlib

4. Project Details

4.1 Operational and Environmental Requirements

- OS: Ubuntu 19.10
- Equipment: webcam installed on the laptop
- Software: python 3.7
Mysql 8.0
- Python module: open-cv module
Face_recognition module



Figure 3. Operating System and Work Environment of the project

4.2 Main Structure

Basically, Our project concept is like the image below.

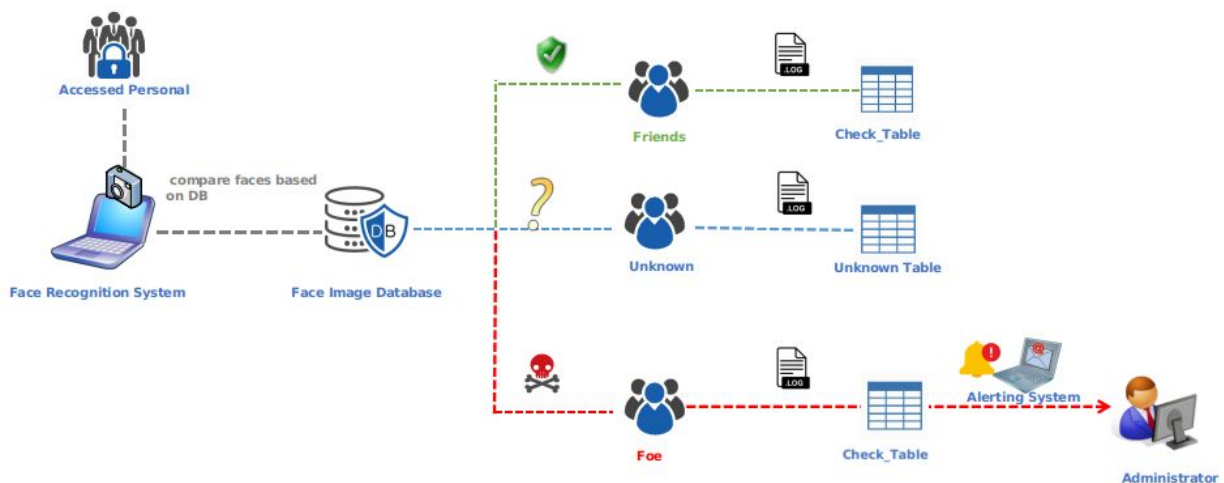


Figure 4. Concept of the facial recognition project

We created 5 Mysql tables (face, check_table, unknown, users, usergroups) in iss database. There are three to six fields depending on the purpose of the table.

SAIT_face_recognition2020					
> __pycache__ > images > Donald_Trump > Jun_Wang > Modea_Kong > Vincent_Saban > unknown 1.jpg add_new.py camera.log database.log drip.ogg face_surveillance.py hash.py installed-mysql-python logger.py login.py mysql_functions.py Will_Smith.jpg					
face					
Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	Auto_increment
fname	varchar(40)	NO		NULL	
lname	varchar(40)	NO		NULL	
encoding	blob	NO		NULL	
isfriend	tinyint(1)	YES		0	
isdelete	tinyint(1)	YES		0	
check_table					
Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	Auto_increment
fname	varchar(40)	NO		NULL	
lname	varchar(40)	NO		NULL	
date	date	NO		NULL	
time	time	NO		NULL	
unknown					
Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	Auto_increment
name	varchar(40)	NO		NULL	
encoding	blob	NO		NULL	
date	date	NO		NULL	
time	time	NO		NULL	
filename	varchar(80)	NO		NULL	
users					
Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	Auto_increment
username	varchar(45)	NO		NULL	
password	varchar(192)	NO		NULL	
encoding	blob	NO		NULL	
gid	int	YES		1	
usergroups					
Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	Auto_increment
name	varchar(40)	NO		NULL	
gid	int	NO		NULL	

Figure 5. Structure of the Project & Database Design: (Tables in database iss)

The purpose of each table is:

3 tables for facial recognition surveillance system:

- face: stores the known personal information, which includes id, first name, last name, face feature encoding and whether this person is a friend or a foe.
- check_table: stores record of known person who accesses the surveillance area including first name, last name, accessing date and time.
- unknown: stores the unknown personal information. It includes id, name, face feature encoding, accessing date and time and the image filename taking for the unknown person.

2 tables for the 2 way authentication system:

- users: stores the information of a registered user (id, username, password, encoding, gid).
- usergroups: stores the user group information, it can be used as permission control.

4.3 Main Functions

① face_surveillance.py

The face_surveillance.py is the major code for capturing and processing captured frames from the surveillance camera. It combines the functions of face recognition, IFF function, alerting, logging, labeling captured faces and displaying processed frames on screen.

• Facial recognition function:

```
68 def face_capture():
69     global frame, face_names, face_locations, talert
70     # Grab a single frame of video
71     ret, frame = video_capture.read()
72     # Resize frame of video to 1/4 size for faster face recognition processing
73     small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
74     # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
75     rgb_small_frame = small_frame[:, :, ::-1]
76     # Find all the faces and face encodings in the current frame of video
77     face_locations = face_recognition.face_locations(rgb_small_frame)
78     face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
79     face_names = []
80     for face_encoding in face_encodings:
81         # See if the face is a match for the known face(s)
82         matches = face_recognition.compare_faces(known_face_encodings, face_encoding, tolerance=0.45)
```

Figure 6. facial recognition function

We accomplished this function by calling functions from opencv and face_recognition module. The video_capture.read(), captures a live frame from camera. After changing the frame into the right format, face_locations function is called to detect the faces in the frame. Then face_encodings function is to extract face features of each face and save the features of each face into a list of numpy arrays. By matching each face feature in the list with the faces we saved in the database, the function compare_faces will determine whether the face is a known face. The default parameter tolerance value is 0.6 and lower numbers make face comparisons more strict. According to our test, 0.45 is the best value suitable for our project.

• IFF function:

We utilized a "isfriend" variable in the mysql_functions.py and a column in the database table face. The value 1, means a friend, 0 means a foe. If a face is recognized as a known face means we found a match in our database, we will check the value of "isfriend" to determine which group the person is in. The result will then be displayed on the screen with a labeled frame around the detected face in the frame.

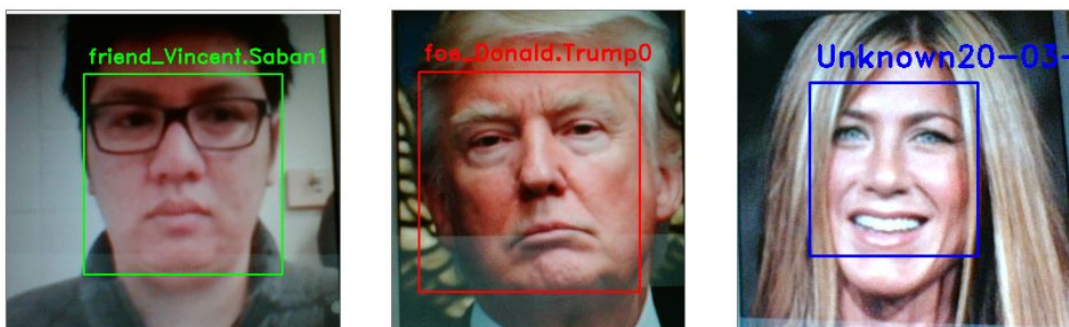


Figure 7. Effects of IFF function

- **Alerting functions:**

We implemented 2 methods for the alert system.

```
18 def alert():
19     playsound('drip.ogg')
```

Figure 8. Function of playing alarm

```
97 logger[("Found a foe!" + str(name[:-1])), 'critical', 'camera']
98 #start a new thread for sound the alarm
99 talert=threading.Thread(target=alert)
00 talert.start()
```

Figure 9. Playing alarm thread

When a foe is found in one frame, the code will initiate a new thread to play the drip.ogg one time, which is an audio file commonly used in Linux alarm systems.

```
114 #send email for alerting
115 tmail=threading.Thread(target=email_sender,args=(name,date,time))
116 tmail.start()
```

Figure 10. Sending alerting email thread

An alerting email will be sent to the administrator's email address, when one foe is recognized for the first time in a certain day.

- **Logging function:**

We designed 2 log files, camera.log and database.log. The camera.log keeps records for all the captured faces by the camera. The database.log keeps logs for changes to the database. Each log file contains 3 categories of logs, for friends we used info, for unknown we used warning and for foes we used critical.

```
INFO: 2020-02-29 19:41:33,977 table unknown created
INFO: 2020-02-29 19:41:34,049 table usergroups created
WARNING: 2020-02-29 19:41:35,743 insert into database unknown Unknown20-02-29-19:41:35
WARNING: 2020-02-29 19:41:35,743 insert into database unknown Unknown20-02-29-19:41:35
INFO: 2020-02-29 19:45:09,259 A new face added to table face Modae Kong
INFO: 2020-02-29 19:45:09,617 A new face added to table face Jun Wang
INFO: 2020-02-29 19:45:10,026 A new face added to table face Vincent Saban
INFO: 2020-02-29 19:45:10,336 A new face added to table face Donald Trump
INFO: 2020-02-29 19:45:10,884 insert into database check_table Jun.Wang
```

Figure 11. database.log

```
INFO: 2020-02-29 19:47:10,190 Found a friend!Jun.Wang
INFO: 2020-02-29 19:47:10,340 Found a friend!Jun.Wang
WARNING: 2020-02-29 19:47:10,535 found a unknown Unknown20-02-29-19:47:10
WARNING: 2020-02-29 19:47:10,687 found a unknown Unknown20-02-29-19:47:10
CRITICAL: 2020-02-29 19:47:10,842 Found a foe!Donald.Trump
CRITICAL: 2020-02-29 19:47:11,015 Found a foe!Donald.Trump
```

Figure 12. camera.log

• Saving Data into the Database

Based on the result of face recognition, the information of the person exposed to the camera will be stored into the database. If the person's face matches one of the faces in our face table, his information will be saved to the check_table. The information includes the person's first name, last name, accessing date and time. If it could not find a match, this person's information will be stored into the table unknown. The unknown person will be named by a string in the form of "unknown" + date +time to label for the uniqueness. The frame containing the unknown person's face will be also captured and saved into an image file ".jpg" under directory "./unknown", at the same time.

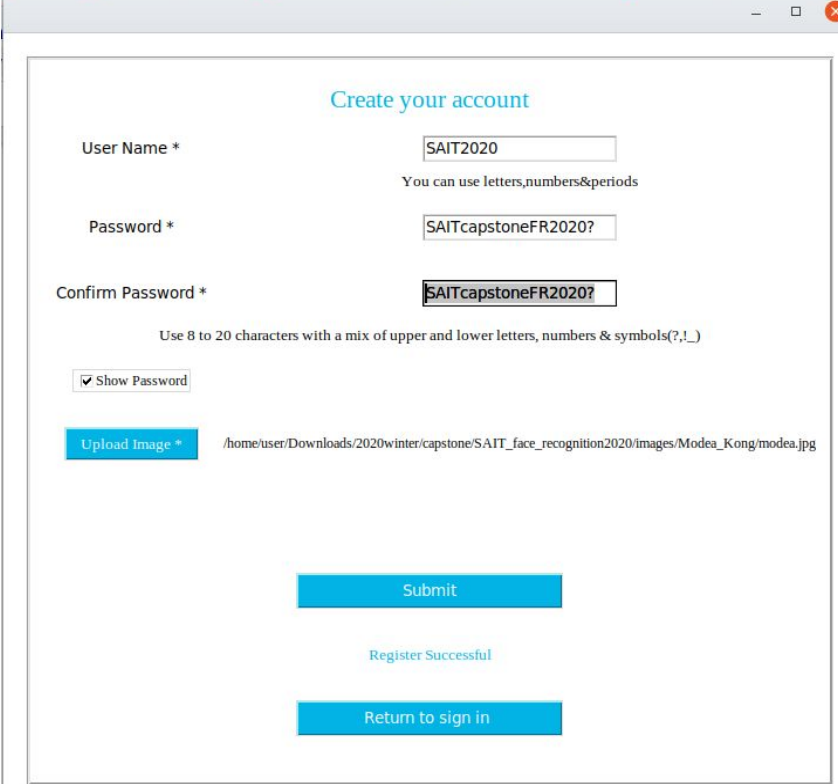
• Saving known faces into database

When the program face_surveillance.py is initiated in a new environment, it reads the images under directory "./images", extracts the face features and saves them into the face table as known faces.

The add_new.py is another way to add a known face into the database. By running this python code in the command line, it can insert a record for a new user into the existing face table.

② login.py

This code is our 2 way authentication system with python tkinter interface. It has the functions of sign up, sign in, saving and comparing username in clear text and passwords in hash512, verifying user's identity by comparing the face feature from uploaded image with the face feature captured by camera. The username and passwords must comply with the sanitation rules. For the first time running this system, the mysql_functions.py must be run in advance which creates tables for the database.



Create your account

User Name *
You can use letters,numbers&periods

Password *

Confirm Password *
Use 8 to 20 characters with a mix of upper and lower letters, numbers & symbols(?,!,_)

☒ Show Password

Register Successful

Figure 13. sign up page

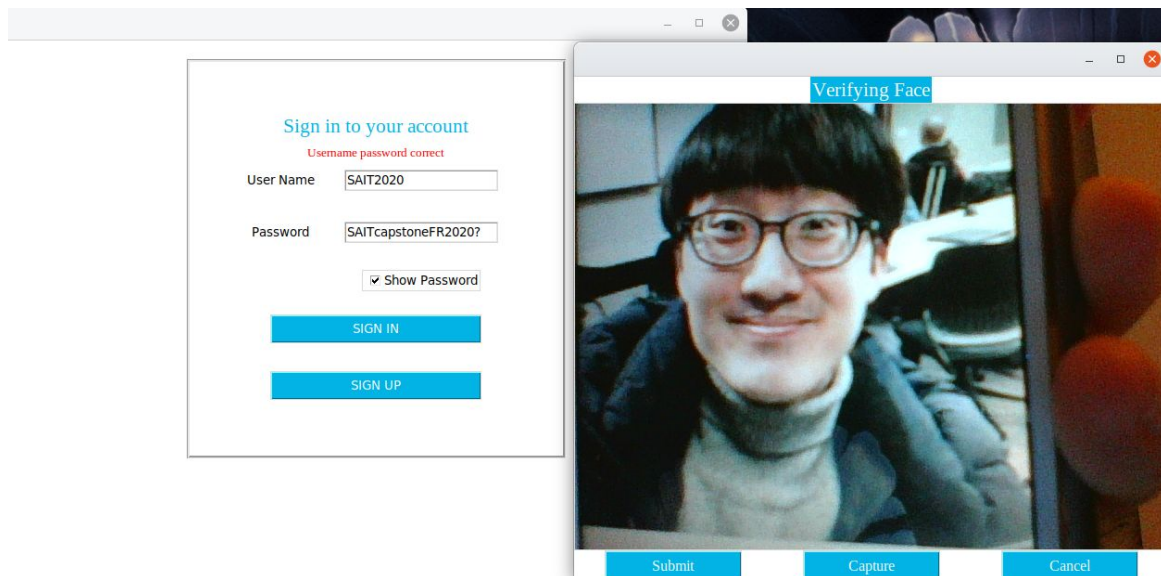


Figure 14. sign in page

The sign up process includes sanitizing user input (username, password and the image file to upload), it compares the username with the database table “users”, makes sure the username is unique in the database. If it does not find a match in the database, it will extract the face feature in the image file and hash the password. Then a record of users will be inserted into the users table.

The sign in function is composed by sanitizing user input, username and password comparing the respective data in the database table users. When the SIGN IN button is clicked, It queries the database by the username which is stored in the database in plain text. If found a match, it will continue to call the hash function to decode the password in the database, if the password and username all match with the record in the database, a new window will pop up to verify the user's face. At the same time, the camera will be turned on to capture the face exposed. Click on the button Capture, the feature of the face will be extracted. When a user clicks on submit, the code will compare the username, password again, and the captured face feature with the encoding of this user's face which was stored in the database in the signup process. Only when all three credentials are verified, the user is allowed to login the system.

4.4 How to Use

(Step 1) Install Mysql and setup password

① Install Mysql database:

```
sudo apt-get install mysql-server -y
```

② Start Mysql :

```
sudo service mysql start
```

③ Log in Mysql:

```
sudo mysql -u root -p
```

④ Change password:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'mplftw20';
```

⑤ Create database:

```
create database iss;
```

(Step2) Install related libraries and modules

```
sudo apt-get -y install python3-pip python3-dev -y
sudo apt-get -y install cmake
sudo apt-get -y install python3-tk
sudo apt-get install python3-pil python3-pil.imagetk
pip3 install --upgrade pip
pip3 install opencv-contrib-python
pip3 install face_recognition
pip3 install mysql.connector
pip3 install playsound
```

(Step 3) Adapt to your needs

- a.) Use your own image files in the folder “./images” . Make sure one picture for each person and put it in the respective folder. The folders must follow the format of “fname_lname”
- b.) Change the email address and password to your own in face_surveillance.py

(Step 4) Run respective code

- a.) `python3 mysql_functions.py`
- b.) `python3 face_surveillance.py`
- c.) `python3 add_new.py imagefilename firstname lastname [0/1]`
Example: `python3 add_new.py Will_Smith.jpg Will Smith 0`
- d.) `python3 login.py`

5. Problems and Solutions

5.1 OpenCV works on a specific environment.

Opencv doesn't work with environments that are written in Qt. An example of that is the KDE plasma desktop environment. Only solution is to use an operating system that uses GTK environment (i.e Ubuntu gnome environment (during the time of the writing, might have changed since then.)

Opencv has two packages (opencv-python and opencv-contrib-python). Make sure to install the latter as that one has the contrib modules that we needed for our script to run.

5.2 Choosing the face recognition module.

We had two main choices when choosing the face recognition module that would suit our needs. It was opencv with the haar_cascade function and face_recognition module. Both have their pros and cons but ultimately, our choice was either if we wanted a faster program or a more accurate one and we went with the accurate route.

5.3 The face recognition process pauses when sending alerts.

Before we apply the solution, when a foe is found, the alerting system runs in the major process. The face recognition had to pause and wait for the alerting process to finish. To solve this problem we implemented a python multithreading module. By running an alert system in separate threads, the major face recognition process runs without interference. This measure makes sure the face recognition system is real-time.

6. Ways to improve the project

We have created a system that can perform real-time face recognition with CPU. Although it is only running at around 6 FPS, it is comparably much faster than using complex CNNs. The quality of the facial images extracted from video cameras cannot be controlled: light, distance and position of the person captured on the video footage limit the facial features. Therefore, live facial recognition technologies are more likely to result in false matches.

However, there are still many things we could do to improve the performance (both the accuracy and speed) of this system. Potentially, we could improve the accuracy using other machine learning methods. MTCNN which uses tensorflow on its backend, would increase the accuracy of the program and the range of how far it can detect faces.

Moreover, we can apply knowledge distillation to compress the current model and further reduce the model size using low bit quantization. Better hardware is another option to improve this project, while doing this project we were only using SAIT's Toshiba laptop, which only has an i5, an integrated graphics card and the laptop's integrated camera. Doing this project with better hardware would have completely changed on how we approached our project. Instead of finding modules that could barely work on our machines, we could have expanded our search and used some of the GPU extensive modules that use machine learning to detect and recognize faces.

7. Conclusion

In this project, a real-time face recognition system is developed by applying 2 python modules OpenCV and face_recognition. A reliable 2 way authentication system is developed based on our real-time face recognition system. They are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc.

Accuracy and timing are major requirements for the live face recognition. Limited by the hardware and software conditions, our project can detect and recognize frontal faces accurately in a certain distance. The recognition speed reaches 6 FPS. Our applications can work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. The 2 way authentication is secure and reliable.

8. Reference

(what is deep learning?)

<https://machinelearningmastery.com/what-is-deep-learning/>

(Face detection with OpenCV and Deep Learning)

<https://becominghuman.ai/face-detection-with-opencv-and-deep-learning-90b84735f421>

(Face Detection – OpenCV, Dlib and Deep Learning)

<https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/>

<https://towardsdatascience.com/mtcnn-face-detection-cdcb20448ce0>

(python face_recognition module)

https://github.com/ageitgey/face_recognition

<https://www.pyimagesearch.com/2018/07/09/face-clustering-with-python/>

(python opencv tutorial)

<https://www.pyimagesearch.com/opencv-tutorials-resources-guides/>

<https://towardsdatascience.com/real-time-face-recognition-with-cpu-983d35cc3ec5>