



INTERNSHIP REPORT

Face Recognition For Door Locking System

Advisor : Pao-Ann Hsiung
Department of Computer Science

Report By :
Gourav Wadhwa
B.Tech 2nd year
Department of Electrical Engineering
Indian Institute of Technology, (IIT) Ropar



OUTLINE

1.	Introduction to Face Recognition	3
1.1	What is Face Recognition?	3
1.2	Techniques used in Face Recognition	3
1.3	Face Recognition using Face Encodings	4
1.4	Libraries Required for Face Recognition	4
2.	Testing Face Recognition	6
2.1	Introduction	6
2.2	Extracting Data	7
2.3	Testing	7
2.4	Improvements	8
2.5	Results	9
3.	Real Time Face Recognition	11
3.1	Introduction	11
3.2	Saving the Encoding Images	12
3.3	Testing and Results	12
4.	Server End Face Recognition	14
4.1	Introduction	14
4.2	Registering Images	15
4.3	Updating a Profile	16
4.4	Recognizing Images	16
5.	Face Recognition On MicroContollers	18
5.1	Introduction to Raspberry Pi 3B	18
5.2	Introduction to Google Coral	19
5.3	Introduction to Nvidia Jetson Nano	20
5.4	Face Recognition Using Raspberry Pi 3B	21
5.5	Circuit For Door Locking System	23
5.6	Results	25
6.	Conclusion	27
7.	References	28



Introduction To Face Recognition

What is Face Recognition?

Face recognition is the process of putting a label to a known Face (If the face is Unknown then we can put an Unknown label on it). Just like humans learn to recognize their family, friends and celebrities just by seeing their face, there are many techniques for a computer to learn to recognize a known face. These generally contain four important steps :

1. *Face Detection* : It is the process of locating a face region in an image (a large rectangle can be plotted around the face of the human being). This step does not care who the person is, just that it is a human face.
2. *Face Preprocessing* : It is the process of adjusting the face image to look more clear and similar to the other faces in the database.
3. *Collecting and Learning Faces* : It is the process of saving many preprocessed faces in the database (a specified location). From here we can choose a path to learn how to recognize these faces.
4. *Face Recognition* : It is the process of putting a label to a new image from all the images in the dataset. If the face is not stored in the database then the label to the new image should be "Unknown".

Techniques used in Face Recognition

There are many methods to learn how to recognize a face. Following are some of the Face Recognition Techniques :

1. *Holistic Matching Methods*: In holistic approach, the complete face region is taken into account as input data into face catching system. One of the best examples is Eigenfaces.
2. *Feature-based (structural) Methods*: In this methods local features such as eyes, nose and mouth are first of all extracted and their locations and local statistics (geometric and/or appearance) are fed into a structural classifier.
3. *Hybrid Methods*: Hybrid face recognition systems use a combination of both holistic and feature extraction methods. Generally 3D Images are used in hybrid methods. The image of a person's face is caught in 3D, allowing the system to note the curves of the eye sockets, for example, or the shapes of the chin or forehead.



Face Recognition using Face Encodings

There are many techniques to learn how to recognize a face but for a door locking system we need to register 100's of people at a time. We may also need to add (or remove) faces from the system when the person is new (or removed) to the company. We also need that the Door Locking system is able to recognize a person from only one photograph and the person does not need to submit a whole dataset of his face. So for this purpose we choose to make Face Encodings of each and every face in the database and then match the new face with all the Face Encodings and with some tolerance the new face was given a label (If the face is not in the database then it was given a label of "Unknown").

For making the Face Encodings we needed to use the `face_recognition` of python which can easily make Face Encoding in a few milliseconds. After we got the Face Encodings of the new test face we can compare all the faces in the database and then we can give a label. This process can take a lot of time when the total faces in the database is more than 1000 but for a company who is operating on people less than 500 this process will just take a few seconds.

Libraries Required For Face Recognition

1. **Dlib** : Before using the Face Recognition Library we need to install Dlib. Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.
2. **Face_recognition** : This is the main library used for recognizing faces in this project. It is built using dlib's state of the art deep learning concepts.
3. **Open CV** : This library is mainly used for opening Cameras and for making a rectangle around the face of the new person when he/she is recognized.
4. **Numpy** : This library is used for array manipulation. As all the images are a 3 dimensional arrays (height * breadth * 3), so it is easy to manipulate the images using numpy.
5. **Beautiful Soup 4** : It would be required for fetching images from the websites. It is mostly used for web scraping.
6. **Flask** : It is used for Server Based Face Recognition. It is used to render different templates on the website. We made a local host server using Flask.
7. **Flask_SocketIO** : It is used to make a pipeline from client to the server. It is used to fetch the images uploaded by the user on the website.



8. *Flask_wtforms* : It is used to make the user register to the Server Side. In this the website asks you for an image in jpg format and a username.
9. *Os* : This library is mostly pre installed in Linux and it is used to give path to the saved images in the local system. It can also be used to make (or remove) a directory.
10. *Urllib* : It is a python module for fetching URLs. It offers a very simple interface, in the form of *urlopen* function. This is capable of fetching URLs using a variety of different protocols.
11. *Regex* : It can be downloaded in python using command - `pip install re`. It is used to regular expression matching operation similar to those found in perl.



Testing Face Recognition

Introduction

We needed to test `face_recognition` library on Indian Faces so we Extracted around 30,000 images from the internet to find the accuracy on the indian faces. We downloaded around 30 people images with 15 out of them were men and 15 were women. Each person contained a total of 1,000 images, each of them with a different pose and different surroundings. Before beginning to download these images we needed to design a tree diagram for Testing Face Recognition. Following is the Tree diagram :

Testing Face Recognition

```
| ----- ExtractImages.py
| ----- Numbering.py
| ----- FaceRecognition.py
| ----- Testing Images
|   | ----- Encoding Images
|   |   | ----- Image1.jpg
|   |   | ----- Image2.jpg
|   | ----- Testing Images
|   |   | ----- Image Type 1
|   |   |   | ----- Image1.jpg
|   |   | ----- Image Type 2
|   |   |   | ----- Image1.jpg
```

So inside the Encoding image we can keep one image of the person whom we want to recognize. Please ensure that in the encoding image there should only be one face in the image and it should be clear enough so that the face is identified. We had 30 different people in the project so there were 30 different Images in the Encoding Images directory and 30 directories in Testing Images with each of them containing 1,000 images. So there were 3 different kinds of python codes used for Extracting Images, Numbering Images and then finding out the Accuracy of each type of face.



Extracting Images

So for extracting images from the web using a search query asked from the user while extracting images. For one search query we find exactly 100 links to download the images (Some of these would be having some downloading issues so the total images downloaded for each search query are less than or equal to 100). While running this code it also ask the Image Type of the face we are downloading. So it saves all the downloaded images inside the folder Testing Images / Testing Images / Image Type 1 (If we have given 1 as the Image Type). When the images are downloading the code itself counts the number of images in the folder and gives the new image it's name according to it. After all the images are downloaded we need to delete all the images that are not containing the face that we aimed for. So we need to number all the images again therefore for that we can use Numbering.py code that numbers all the images consistently.

After downloading all the images we had almost 30,000 images out of which 15,000 were of 15 different men and 15,000 were of 15 different women. Each person had 1,000 images. We need to keep one image of each of the 30 people in the encoding image (Please keep in mind that the encoding image should only contain one face and it should be clear enough so that we are able to identify a face in the image).

Testing

We tested the face_recognition on the dataset of 30,000 images without any preprocessing of the image and we found the following results :

Men : 87.24 %
Women : 93.13 %

The main reason for the low accuracy on the Men was because most of the men we chose in our database were quite old and the dataset contained both young and old image of these people, due to which the accuracy was decreased for men.

The time taken for recognizing a photo was :

Unrecognized Image : 0.41 seconds
Recognized Image : 0.32 seconds



So the time taken for recognizing an image was less than 1 seconds with 30 encoded images, so up to 500 encoding images the code can run for less than 1 second which is very good for Door Locking System.

There were many reasons because of which we were not able to recognize a face and following are some of the most important reasons :

1. The face_recognition library was not able to identify a face if the orientation of a face is changed from vertical to horizontal. So we needed to rotate the image by 90 degrees to get the face in Vertical position (The main problem was that we did not know in which case we needed to rotate the face and in which face we did not needed to rotate the face).
2. If a person is drenched then it was really difficult for the library to recognize a face. In this case we were not even able to identify a face so it was really difficult to solve this problem.
3. If the image of the person is too dark or too bright then also the face_recognition library was not able to identify the face. In this case we needed to use Image enhancements to make the face more recognizable.
4. The code was not able to identify a face if the height breadth ratio is changed. It was not a major problem because at last we wanted to make Door Locking System so in that system we could always have a normal height breadth ratio.
5. If the hand is touching the face or covering some small portion of the face then also the library is not able to identify the face.

So for tackling these important issues we needed to do some improvements by which the accuracy of the system would improve and the Door Locking System would be much more accurate.

Improvements

To increase the accuracy of our Face Recognition system we needed to make some improvements, the 2 biggest causes because of which we were not able to get high accuracy were :

1. The Horizontal Face Problem
2. Surrounding Light Problem

Issue 1 : The Horizontal Face Problem

```
def rotate_image(img):  
    rotated_image = [[] for x in range(len(img))]  
    for i in range(len(img)):  
        for j in range(len(img[i])):  
            rotated_image[len(img) - j - 1].append(img[i][j])  
    return rotated_image
```




The function `rotate_image(image)` can rotate an image by 90 degrees in clockwise direction. So if we are not able to find any face in an image then we can rotate it and check whether there is a face in the image or not. As the image is only a 3 Dimensional array of numbers so we can easily do it using numpy arrays.

Issue 2 : Surrounding Light Problem

For this problem we can use image enhancement techniques and the PIL library which is made for this specific reason.

```
bright_test_image = (((test_image / 255.0) ** 0.5) * 255.0).astype('uint8')
dark_test_image = (((test_image / 255.0) ** 2.0) * 255.0).astype('uint8')

test_face_encodings = face_recognition.face_encodings(bright_test_image)
test_face_encodings.extend(face_recognition.face_encodings(dark_test_image))
```

This code is used to make an image brighter or darker in appearance. The brightness can the darkness factor can be controlled continuously so there is no problem in that.

In this case :

We have used 0.5 factor for the bright image, and

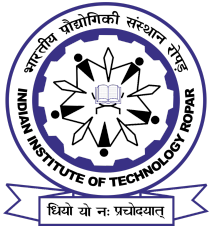
We have used 2.0 factor for the dark image.

Applying these improvements would make the code run a little slower but the accuracy for the code can increase drastically.

Results

After applying the improvements it was seen that the code was slower on the previously unrecognized images but for now it was able to correctly label the previously incorrectly labelled images. So the accuracy for the Face Recognition was found to be :

Men : 91.22 %
Women : 95.83 %



The time taken for recognizing a photo was :

Unrecognized Image : 1.94 seconds

Recognized Image : 0.44 seconds

So from the above stats we can easily see that the accuracy has increased by 4 - 5 % but the running time for Unrecognized Images has gone up exponentially. We can see that these stats are for 30 face encodings in the dataset but for 200 images this technique can perform perfectly well for a Door Locking System.

So if we have more than 200 people registered in the dataset then we have to sacrifice on the accuracy and we need to use the previous algorithm but if we have less than 200 images in the dataset then we use this algorithm with high accuracy.



Real Time Face Recognition

Introduction

To make a Real Time Face Recognition we needed 2 things. Firstly, we needed to register new people and for that we need to store the face into the Dataset. Secondly, we would need to make a Real Time Face Recognizer System which would load the images from the Dataset and save it in the form of encodings and match it with the real time test faces. For making this system we would need a Proper Tree Diagram of Real Time Face Recognition. So following is the Real Time Face Recognition Tree Diagram :

Real Time Face Recognition

```
| ----- SavingImages.py
| ----- RealTimeRecognition.py
| ----- Real Time Images
|   | ----- Encoding Images
|   |   | ----- Image1.jpg
|   |   | ----- Image2.jpg
|   | ----- Saving Real Time Images
|   |   | ----- Image Type 1
|   |   |   | ----- Image1.jpg
|   |   | ----- Image Type 2
|   |   |   | ----- Image1.jpg
|   | ----- names.txt
```

So in this the SavingImages.py can be used to save the images in the Folder Encoding Images from where we can fetch the images to match with real time camera feed. At the time of registering a person you will also be prompted to write your username which will be written inside the file named names.txt from which it will be fetched at the time of Real Time Face Recognition.

RealTimeRecognition.py code is used to recognize a face in real time on running the code we can see a screen popped up on which there will be a rectangular box around the face with the recognized label (If the image is not Recognized then it would show “Unknown” tag under the face)



Saving the Encoding Images

For recognizing a person firstly we would need to store the person image in the Database. For that we have the code Saving Images.py, when we run this code it will be stuck on the first frame for going to next frame we need to press escape key and when we want to save the frame face into the database we need to press spacebar after which it will prompt you to enter a username after entering the username you will be registered into the database.

The code have some restriction on saving the image as an encoding image, Following are the cases where you cannot save a frame into the Encoding Images

Case 1 : If there is no face in the frame then we cannot save the frame in the database

Case 2 : If there are two or more faces in the frame then we cannot save the frame in the database

Case 3 : If the face that is in the frame is already in the database then also that frame cannot be stored in the database

If it frame clears all the 3 Cases then we save the frame into the database which will be then further used to recognize a new image.

While saving the image we need to verify that the person face is completely visible and he/she is sitting with a clear straight face so that it would be easy to recognize the face at the time of Real Time Face Recognition.

Results

At the time of testing we tested with 15 known face encodings and following were the accuracies on women and men :

*Men : 98.45 %
Women : 97.89 %*

These were tested with real time data and these stats were on around 10,000 frames. Out of 15 people 6 were women and 9 were men. We were able to recognize the real time face feed in both dark and bright surroundings. We were also able to recognize horizontal faces which earlier were not being recognized.

The time required to recognize a face was :

*Unrecognized Image : 1.68 seconds
Recognized Image : 0.46 seconds*



So it was clear that it was taking too much time on unrecognized Faces so we needed to delete some of the improvements so that it took a realistic time for the Real Time Face Recognition.

As we know that most of the people keep their faces vertical and not horizontal at any point of time, so it is a useless thing to try and rotate image again and again to check whether there is a face in front of the screen or not. So by removing this we can save the time on Unrecognized Image but for Recognized Image there will be no change.

So after applying the Improvements following was the time taken for Recognizing a person :

Unrecognized Image : 1.03 seconds

Recognized Image : 0.45 seconds

So now we have saved more than half a second for recognizing an Unknown person. We could further improve the timings but then there would be a drop in the accuracy. If we know that the surroundings will be dark or bright then we can remove the Image Enhancements, and after removing following were the time taken to Recognize an image :

Unrecognized Image : 0.62 seconds

Recognized Image : 0.45 seconds

So now for this version we had a good accuracy and really good timings too but the condition would be, we should know what is lighting condition where we are going to use this face recognition technique.



Server Side Face Recognition

Introduction

For making the Server side Face Recognition system most important part was to make the Tree diagram of the system, so following is the Tree Diagram :

Server Side Face Recognition

```
| ----- FaceRecognition.py
| ----- server.py
| ----- forms.py
| ----- names.txt
| ----- Dataset
|   | ----- Registered
|   |   | ----- Image1.jpg
|   |   | ----- Image2.jpg
|   | ----- Recognized
|   |   | ----- Image Type 1
|   |   |   | ----- Image1.jpg
|   |   | ----- Image Type 2
|   |   |   | ----- Image1.jpg
| ----- static
|   | ----- Server To Client
|   |   | ----- Image1.jpg
|   |   | ----- Image2.jpg
|   | ----- main.css
| ----- templates
|   | ----- home.html
|   | ----- layout.html
|   | ----- register.html
|   | ----- recognize.html
|   | ----- recognized_image.html
|   | ----- update.html
```



In this the html files are all in the templates folder and all the css files are in the static folder. For running the server end version of Face recognition we just need to execute the server.py file and then you could be able to see the website hosted on the website localhost:8000. The Registered images will be stored inside the folder Dataset / Registered and the names of the users will be stored in the file names.txt. The images that are being recognized will be saved in the folder static / ServerToClient. The Last image in here will be shown recognized_image tab on the webpage. When a person is recognized his / her face is stored in the folder Dataset / Recognizing / Image Type (x). The number “x” will be defined from the number by which it is saved into the Registered images. The Image Type folder will be made itself so we don’t need to make those folders. If the person is unknown then the face will be stored Image Type 0.

For running Server Side Version we would need the following additional Libraries :

1. *Flask*
2. *Flask SocketIO*
3. *Flask_wtf*
4. *wtfforms*

FaceRecognition.py code is used to make a recognized image from the raw image which is then stored in the folder static / Server To Client / Image1.jpg. It will also store all the faces in the image in the folder Dataset / Recognizing / Image Type 1 / Image1.jpg.

Server.py code starts a server and render the home page in the starting, you would be able to register, update and recognize using this server. On running this code the website will be hosted on localhost:8000.

Forms.py code is used to create the forms for registering a user and updating the profile of the user in the database.

Registering a User

To register a user we just need to click on the “Register” Tab on the website and then you will be rendered to the webpage. On the webpage you need to enter your Username and upload a photo in the form and if all the following conditions are satisfied then the user will be registered otherwise an error message will be displayed

1. *If there is no face in the image then the user will not be saved into the database.*
2. *If there are two or more faces in the image then the user will not be saved into the database.*



3. *If the face in the image is already in the database then also that image will not be saved into the database.*
4. *If the username entered is of length less than 4 alphanumeric digits then the user will not be registered into the database*
5. *If the username is already taken by another user then the user will not be registered into the database.*

So if the username and the photo uploaded by the user clears all the above criteria then the user will be registered in the database otherwise an error message will be shown. On completion of registration you would see a flash message saying “Successfully Registered”. After the registration process you would be able to see your image inside the folder Dataset / Registered and your username inside the text file name.txt.

Updating a Profile

Sometimes we would need to update the face of the user after some time, for this update feature has been added into the website. Whenever you need to update the face of the person you just need to click on “Update” option on the website and then enter your username and add your new photo in the form. Following are the cases when the profile will not be updated :

1. *If there is no face in the image then the user will not be updated into the database.*
2. *If there are two or more faces in the image then the user will not be updated into the database.*
3. *If the face in the image is already in the database (other than previous face user uploaded) then also that image will not be updated into the database.*
4. *If the username does not exist in the database then the profile will not be updated in the database.*

If there is no problem in the profile then the user will be updated in the database and you will see a message saying “Successfully Updated” on the website.

Recognizing Images

You can recognize a frame too that will be streaming on the web page. The video stream will start whenever you give permission for the webcam. When the video stream is going on you can click on the button below the video stream at anytime to send the frame from client to the server from their it will be processed and returned to the webpage. You can see your Recognized image in the tab



recognized image. Your face will be stored inside the folder *dataset / Recognized / Image Type (x)*, where *x* is registration id. The whole frame after the processing will be saved in the folder *static / ServerToClient / Image1.jpg*. The processing will draw a rectangle around the face and write the username of the user below it.

Following was the time taken for Recognizing a Face :

Unrecognized : 1.76 seconds
Recognized : 0.44 seconds

As this is not the real time Face Recognition system we can bear with a second delay but if we want faster results we need to delete some of the optimizations.

The time taken when there were following number of recognized faces in the image :

1 Face : 0.44 seconds
2 Faces : 0.73 seconds
3 Faces : 0.97 seconds
4 Faces : 1.11 seconds

So on the increase of number of recognized faces in the frame the time approximately increases linearly.

Note : These all stats were taken with 30 known encodings.

Face Recognition on Micro Controllers

Introduction to Raspberry Pi 3B



The Raspberry Pi 3B is the latest microcontroller of all the Raspberry pi's and now it has a Quad-Core 64Bit CPU. The processing power of Raspberry Pi 3B has been increased by 10x compared to the first model of this company. Following are the steps to install Raspberry Pi OS into a Memory card and then starting the work in Raspberry Pi :

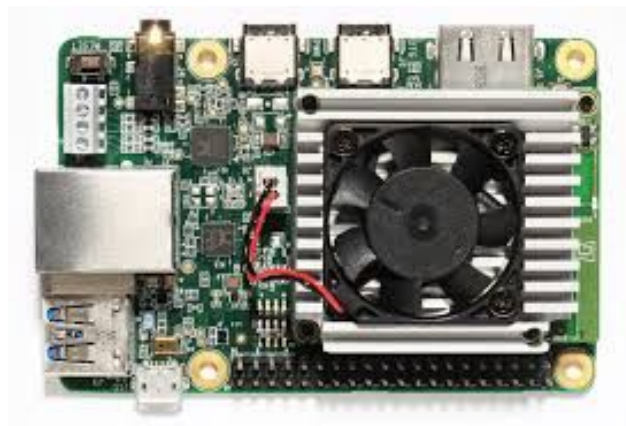
- 1. Download the Raspberry Pi Stretch OS from [this](#) website.*
- 2. Download Balena Etcher into your laptop from [this](#) website*
- 3. Mount the SD card into your laptop and Flash the Raspberry Pi Stretch OS into the memory card using balena Etcher.*
- 4. Now insert the memory card into the Raspberry Pi and then you can see Raspberry Pi booting into the OS.*

In Raspberry Pi 3B you would need to install the following libraries :

1. *Dlib : For installing Dlib you can refer to [this](#) website*
2. *Face Recognition : Face Recognition can be installed using the command :
pip3 install face_recognition*
3. *OpenCV : For installing Open you can refer to [this](#) website*
4. *Pygame : Pygame can be installed using the command :
pip3 install pygame*
5. *Re : Regex can be installed using the command :
pip3 install re*

After installing all the libraries we are all ready to run the Face Recognition in Raspberry Pi 3B.

Introduction To Google Coral



Google Coral is most recently released microcontroller, it has released in march 2019. It is a single board computer supporting Tensorflow Processing Unit. So it is the fastest of all the microcontrollers. It supports Mendel Linux (derivative of Debian). It has a quad-core processor with 1GB of RAM and it support Google Accelerator too.

For booting into the Google Coral you need to follow the steps in [this](#) website

When you will be booted into the mendel linux you need to install the following libraries for making Face Recognition work :



6. Dlib : For installing Dlib you can refer to [this](#) website
7. Face Recognition : Face Recognition can be installed using the command :
`pip3 install face_recognition`
8. OpenCV : For installing Open you can refer to [this](#) website
9. Pygame : Pygame can be installed using the command :
`pip3 install pygame`
10. Re : Regex can be installed using the command :
`pip3 install re`

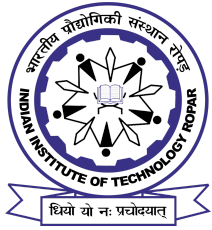
As the OS is Linux all the commands you use normally will work and you can easily install all the libraries without any problem.

Introduction to Nvidia Jetson Nano



Nvidia Jetson nano is a small, powerful computer that lets you run multiple neural networks. It has a quad-core processor with 1GB of RAM. It is one of the most powerful single board computers which is currently being used for many big projects. Following is the process to download the Nvidia Jetson Nano OS :

1. Download the Nvidia Jetson Nano OS from [this](#) website.
2. Download Balena Etcher into your laptop from [this](#) website
3. Mount the SD card into your laptop and Flash the Nvidia Jetson Nano OS into the memory



card using balena Etcher.

4. *Now insert the memory card into the Nvidia Jetson Nano and connect it to a screen, keyboard and mouse, then you can see Nvidia jetson Nano booting into the OS.*

As the Nvidia Jetson Nano is basically Ubuntu then we download the following libraries easily :

1. *Dlib : For installing Dlib you can refer to [this](#) website*
2. *Face Recognition : Face Recognition can be installed using the command :*
`pip3 install face_recognition`
3. *OpenCV : For installing Open you can refer to [this](#) website*
4. *Pygame : Pygame can be installed using the command :*
`pip3 install pygame`
5. *Re : Regex can be installed using the command :*
`pip3 install re`

After installing these libraries we can test all the Face Recognition codes.

Face Recognition using Raspberry Pi 3B

After downloading all the libraries in Raspberry Pi, we started testing Face Recognition library. We extracted 20,000 images of 20 different people. Out of these 20 people 10 were men and 10 were women. Each person had 1000 images where there faces were clear and the code should have recognized those faces. Following was the accuracy found on Raspberry Pi Model 3B :

*Men : 89.21 %
Women : 94.34 %*

The low accuracy for men was because the images of men which were downloaded were really old and their testing images had both there young age and old age images because of which the accuracy is low for men.



The time taken to recognize a images was :

Unrecognized Image : 23.41 seconds

Recognized Image : 5.43 seconds

So the time taken for recognizing any of the image is much greater than required for a Real Time Recognition. Therefore we needed to remove the improvements we applied in the Testing Face Recognition code. So after applying the changes following was the accuracy on the recognized image :

Men : 86.51 %

Women : 90.11 %

So the accuracy was decreased by 3 - 4 %.

But we found that the time taken for recognizing an image was decreased to :

Unrecognized Image : 4.47 seconds

Recognized Image : 1.23 seconds

Now the time taken to recognize was really less but still we cannot make a Real Time Recognition system using this. So we planned to make a button on after clicking that button we would get a signal and then we will process that frame and recognize all the faces in the frame.

We found that the time taken to recognize a known face changes with the number of known faces in the frame. Following is the time taken with different number of faces in front of the screen :

1 Face : 1.23 seconds

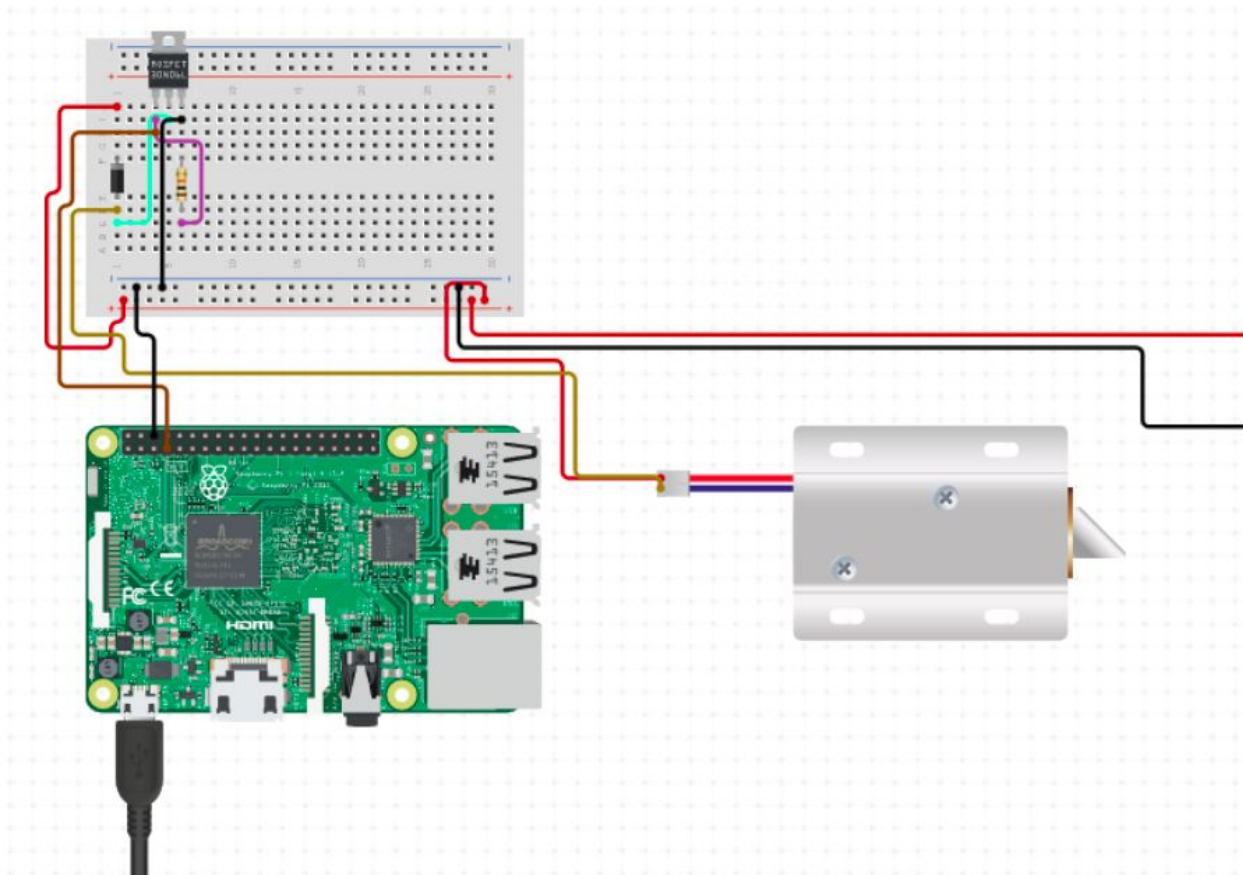
2 Faces : 2.36 seconds

3 Faces : 3.19 seconds

4Faces : 4.07 seconds





















So the more number of faces in front of screen will increase the time needed to recognize all the people in that frame.

Circuit For Door Locking System



This is the circuit diagram for connecting the Lock system to the Raspberry Pi 3B. We are using Mosfet because the Raspberry Pi 3B works at 5 volts whereas the Lock works only at 12 volts. We can connect any working pin of Raspberry pi as red wire but the black wire needs to be connected to the Ground. Following is the Pin diagram of Raspberry Pi :

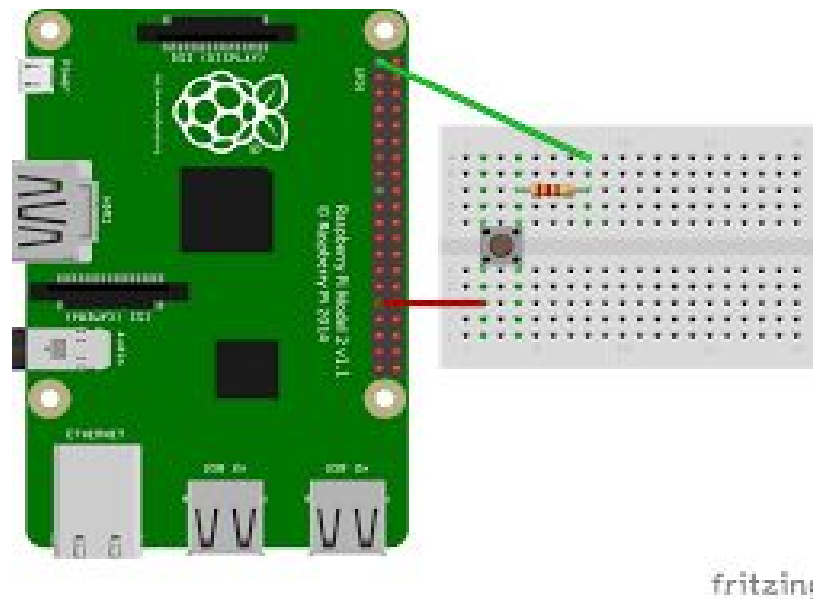
Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

For connecting the red wire try to use only the red wire and for the black wire always connect to any of the Ground in Raspberry Pi.



The above circuit is for connecting a button to Raspberry Pi 3B. For this we would need a resistor or the button / Raspberry Pi can get damaged. Be absolutely sure that all the circuits are joined correctly or the Raspberry Pi can get damaged.

Results

We tested the Door Locking System with around 30 faces and following were the time taken in the whole process :

Time For Reading Known Images : 23.51 seconds

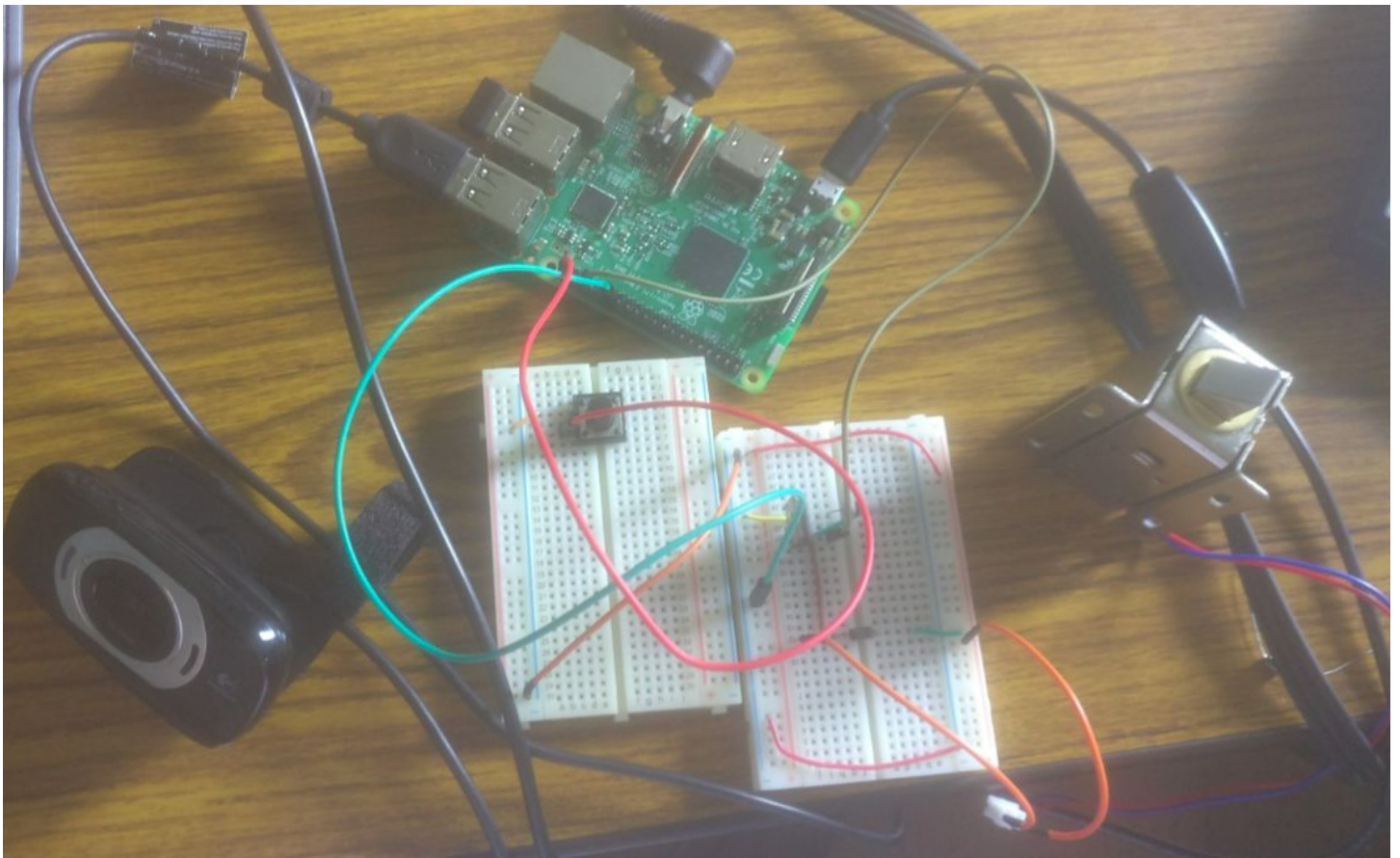
Time to Recognize a known Image : 4.57 seconds

Time to Recognize an Unknown Image : 6.92 seconds

So it can be finally used as Door Locking System but still we can make some improvements so that the time taken to recognize an image can be decreased.

We used a battery source to power the small monitor and we used it for giving power to circuit. We still needed an output source for powering the Raspberry Pi system but it can also be improved and we can take voltage from the battery source only after stepping down the power.

Following is the photo of the whole system when it is working





Conclusion

So we made the following progress during my internship

- 1. We made a Testing Face Recognition System which can be used to recognize all the images saved locally. We can easily measure the percentage accuracy of the system using it. We tested the system on around 30,000 images.*
- 2. We made a Real Time Face Recognition System which was used to save images locally and then by using those locally saved images as known Images we could easily do a real time face recognition, in which frames will be coming from the camera and after some processing we would output the recognized frames. It was a little laggy but we ensured that it worked correctly and the accuracy was above 90 percent.*
- 3. After this we made a Server Side Face Recognition System which had the features of registering a person, updating the profile of a person, and recognizing a person. It also did some test before registering and updating the profile of the person.*
- 4. Next we worked on the software side of Raspberry Pi Door Locking System and we attached the camera, the Lock, the button and the monitor to make it a working model. We connected the USB camera to Raspberry Pi.*
- 5.*

Following is the final Accuracy achieved by our model :

*Men : 91.22 %
Women : 95.83 %*

Following is the Time Taken on the laptop with 30 known Encodings :

*Unrecognized Image : 1.94 seconds
Recognized Image : 0.44 seconds*

Following is the Time Taken on the Raspberry Pi 3B with 30 known Encodings :

*Time For Reading Known Images : 23.51 seconds
Time to Recognize a known Image : 4.57 seconds
Time to Recognize an Unknown Image : 6.92 seconds*

You can find all the codes written by on [here](#)



References

1. <https://www.raspberrypi.org/downloads/raspbian/>
2. <https://www.balena.io/etcher/>
3. <https://gist.github.com/ageitgey/1ac8dbe8572f3f533df6269dab35df65>
4. https://docs.opencv.org/3.4.1/d2/de6/tutorial_py_setup_in_ubuntu.html
5. <https://coral.withgoogle.com/docs/dev-board/get-started/>
6. <https://developer.nvidia.com/jetson-nano-sd-card-image-r322>
7. <https://ourcodeworld.com/articles/read/841/how-to-install-and-use-the-python-face-recognition-and-detection-library-in-ubuntu-16-04>
8. <https://www.learnopencv.com/install-opencv3-on-ubuntu/>

You can find all codes on the following link :

1. <https://github.com/megatron-3/Face-Recognition-NCCU>