ANJUMAN - I - ISLAM'S KALSEKAR
TECHNICAL CAMPUS, NEW PANVEL
ENGINEERING & TECHNOLOGY
ARCHITECTURE
PHARMACY
AIKTC
NAVI MUMBAI - INDIA

A PROJECT REPORT

ON

"ATTENDANCE MANEGMENT APP USING FACE RECOGNITION"

Submitted to

UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN

ELECTRONIC & TELECOMMUNICATION ENGINEERING

BY

KHAN AMANULLAH VASIULAH	12ET24
NACHAN SHAINEELA SAHIR	17DET54
HANFI SAMEENA ASADULLAH	16ET03

UNDER THE GUIDANCE OF

Prof. Altaf Balsingh

DEPARTMENT OF ELECTRONIC & TELCOMM ENGINEERING

Anjuman-I-Islam's Kalsekar Technical Campus

SCHOOL OF ENGINEERING & TECHNOLOGY

Plot No. 2 3, Sector - 16, Near Thana Naka,

Khandagaon, New Panvel - 410206

CERTIFICATE

DEPARTMENT OF ELECTRONIC & TELCOMM ENGINEERING

Anjuman-I-Islam's Kalsekar Technical Campus
Sector 16, New Panvel , Navi Mumbai
University of Mumbai

This is to certify that the project entitled **ATTENDANCE MANEGMENT APP USING FACE RECOGNITION** is a bonafide work of **KHAN AMANULLAH VASIULLAH (12ET24), HANFI SAMEENA ASADULLAH (16ET03),NACHAN SHAINEELA SAHIR (17DET54)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Department of Electronics and Telecommunication Engineering.

Supervisor	Examiner
Head of Department	Director

Project Report Approval for Bachelor of Engineering

This project entitled **" ATTENDANCE MANEGMENT APP USING FACE RECOGNITION"** by Khan AmanullahVasiullah is approved for the degree of **Bachelor of Engineeringin Electronics and Telecommunication** .

Examiner

.....

Supervisor

.....

Date;

Place:

Declaration

I declare that this written submission represents our ideas in words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

……

KHAN AMANULLAH VASIULLAH 12ET24

……

NACHAN SHAINEELA SAHIR 17DET54

……

HANFI SAMEENA ASADULLAH 16ET03

Date:

Acknowledgments

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals . I would like to extend my sincere thanks to all of them.

We am highly indebted to (Name of your Guide) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express my gratitude towards my parents& Staff of AnjumanI-Islam's Kalsekar Technical Campus for their kind co-operation and encouragement which help me in completion of this project.

.

We My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Khan AMANULLAH VASIULLAH (12ET24)
NACHAN SHAINEELA SAHIR (17DET54)
HANFI SAMEENA ASADULLAH (16ET03)

ABSTRACT

Commencing from the last decade several different methods have been planned and developed in the prospect of face recognition that is one of the chief stimulating zone in the area of image processing. Face recognitions processes have various applications in the prospect of security systems and crime investigation systems. The study is basically comprised of three phases, i.e., face detection, facial features extraction and face recognition. The first phase is the face detection process where region of interest i.e., features region is extracted. The 2nd phase is features extraction. Here face features i.e., eyes, nose and lips are extracted out commencing the extracted face area. The last module is the face recognition phase which makes use of the extracted left eye for the recognition purpose by combining features of Eigenfeatures and Fisher features.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Various operations make use of computer based procedures and methods in order to execute processing on digital images. These operations are comprised of methods such as image reconstruction, enhancement, compression, registration, rendering, analysis, segmentation, feature extraction, face detection and recognition, respectively all these processes are component of a huge field called digital image processing. Face recognition is an enormous field comprised of huge processes of investigation and examination based methods.

Each year the attempted solutions grow in complexity and execution times. If we examine the history of face recognition filed, then we can see that the work on this field has its initialization roots in the times almost 35 years back from the present time. Most demanding field in the image processing is face detection and recognition system, numerous methods and techniques have been proposed and developed in this regard, but still there is a huge room for new and effective work. If we talk in general about the face recognition process, then we can say that it is a process comprised of examining a person's face and then relating and matching the scanned face next to a data base consisting of acknowledged faces. It identifies individuals by features (Hiremath et al., 2007).

Pattern recognition is a dynamic subject in prospect of human face recognition. It has huge range of applications such as recognition, driving certification and identification portrait confirmation checks mechanical safety methods, image dispensation, visualization centered human device communication, face detection and recognition and in the prospect of entertainment. There exist two main types in regard of human face identification centered on holist or examination of facial features. One type makes use of all the characteristics of a pattern or facial features.

The main methods under this field are neural network based face recognition (Khanfir and Jemaa, 2006), elastic graph match method for face recognition (Wiskott et al., 1999), Eigenface (Turk and Pentland, 1991), fisher faces and face recognition through density-isocline examination face equivalent. The second type in this regard works on specific features extracted from the face. The main methods under this type include face recognition through facial feature extraction, template matching approach (Hsu et al., 2002) and boundary tracing approach.

Face recognition base on human face geometry was also an area under consideration. Face recognition in many techniques seems like three phase process i.e., face detection, features extraction and lastly the face recognition phase (Zhao et al., 2003). In this study face detection, features extraction and recognition technique is presented which detects the face from the frontal view face images based on skin color information. After that face region containing the face is extracted. After having the detected face, the eyes are extracted using eye maps. extracted using eye maps.

Then nose and lips are extracted on the basis of distance between the two eyes. After the features extraction, a small face region is picked in order to improve the computational time. The picked region is the left eye from the extracted features. For the recognition purpose combination of Eigenfeatures and Fisher features is used.

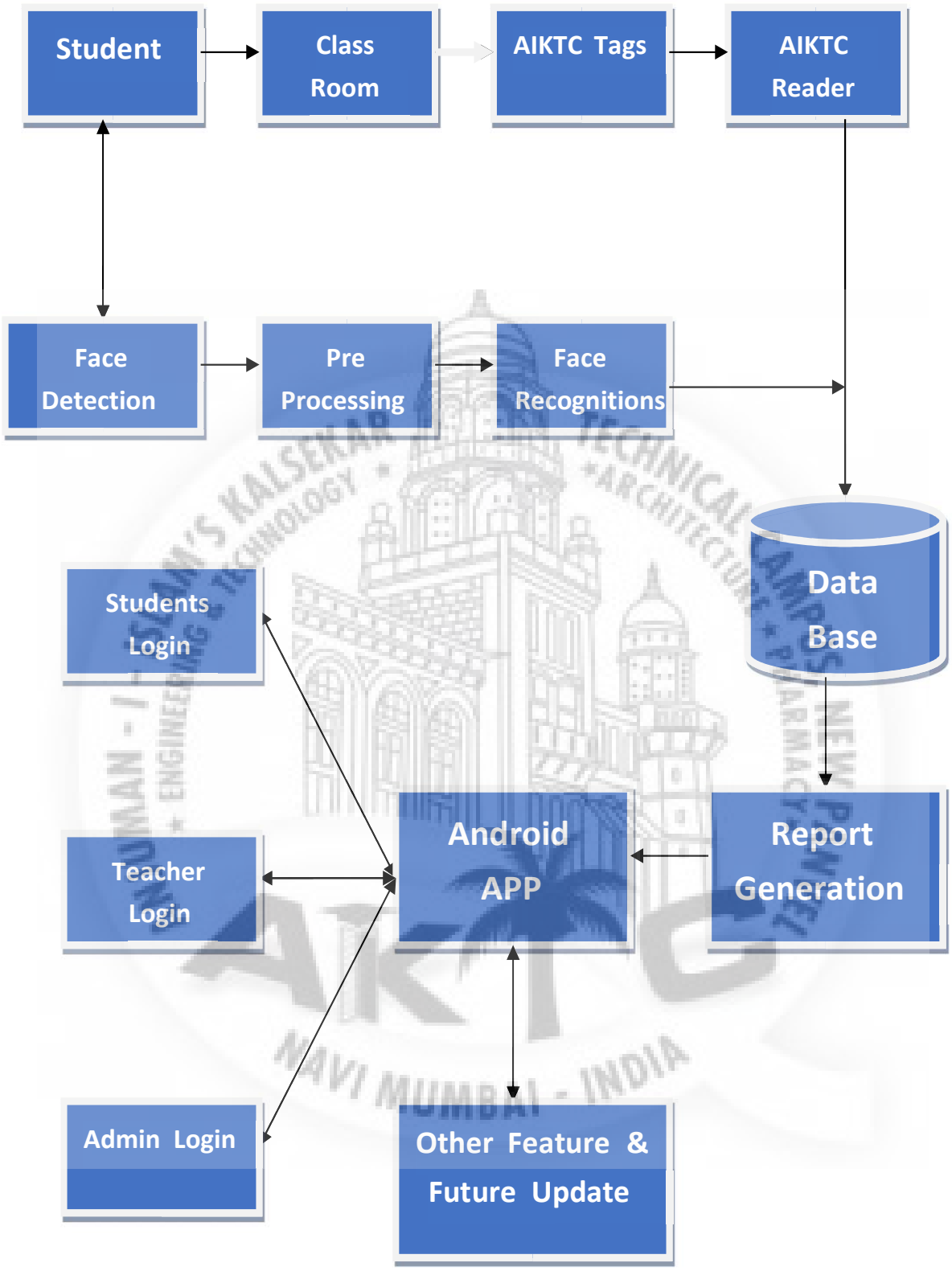


Fig. 1 Project Architecture

Chapter 2

Literature Survey

2.1

Existing work: There are various diverse methods proposed in the perspective of face recognition. Huge work has been done in this area and many different methods had been used for the face detection and recognition purpose. These methods are ranging from Eigen face approach (Turk and Pentland, 1991), Gabor wavelets (Duc et al., 1999; Zhang et al., 2005) to 2D discrete cosine transform for facial features extraction (Eickler et al., 2000), geometrical approach (Hiremath and Ajit, 2004; J'and Miroslav, 2008).

PCA based facial features extraction (BELHUMEUR et al., 1997 a, b) has a drawback that the method is sensitive to the illumination changes. Few most commonly used methods are PCA (Turk and Pentland, 1991), LDA (Belhumeur et al., 1997 a, b), ICA (Bartlett et al., 2002), kernel methods, Gabor wavelets (Yousra and Sana, 2008; Zhang et al., 2004), neural networks etc. Many researchers used the facial features for the recognition purpose. Some extracted the face features values in the form of Eigen values or Fisher values and some used the features of face i.e., eyes, nose and lips to recognize the face images.

Accurate facial features extraction is important for accurate face recognition. Features are extracted in many different ways that provide different types of issues to be considered. An earlier approach to face features extraction was proposed by Yuille et al. (1989). He mainly used templates of deformable parameters to extract eyes and mouth. But the method was computationally expensive and accuracy was not guaranteed.

An existing method that extracts facial features and performs recognition based on these extracted features is developed by Wiskott et al. (1999). In this technique a feature extraction (iris, mouth) and face recognition approach is presented. Iris and mouth are extracted and the template matching approach is used for the recognition purpose. The algorithm first locates the face area by means of skin color subsequent to the extraction of applicants of iris centered calculated costs. After that the mouth region is extracted by making use of color space called RGB. After that associated component process is utilized to accurately extract out the lips region. In order to assure that detected lips are mouth, mouth corner points are extracted using a method called SUSAN approach. Lastly face identification is carried out using template matching process.

In proposed work, both methods were used for the recognition purpose. The features of the face are detected commencing the detected face and then left eye is picked whose Eigen and Fisher values are calculated for the recognition purpose

2.2 General Flow Chart

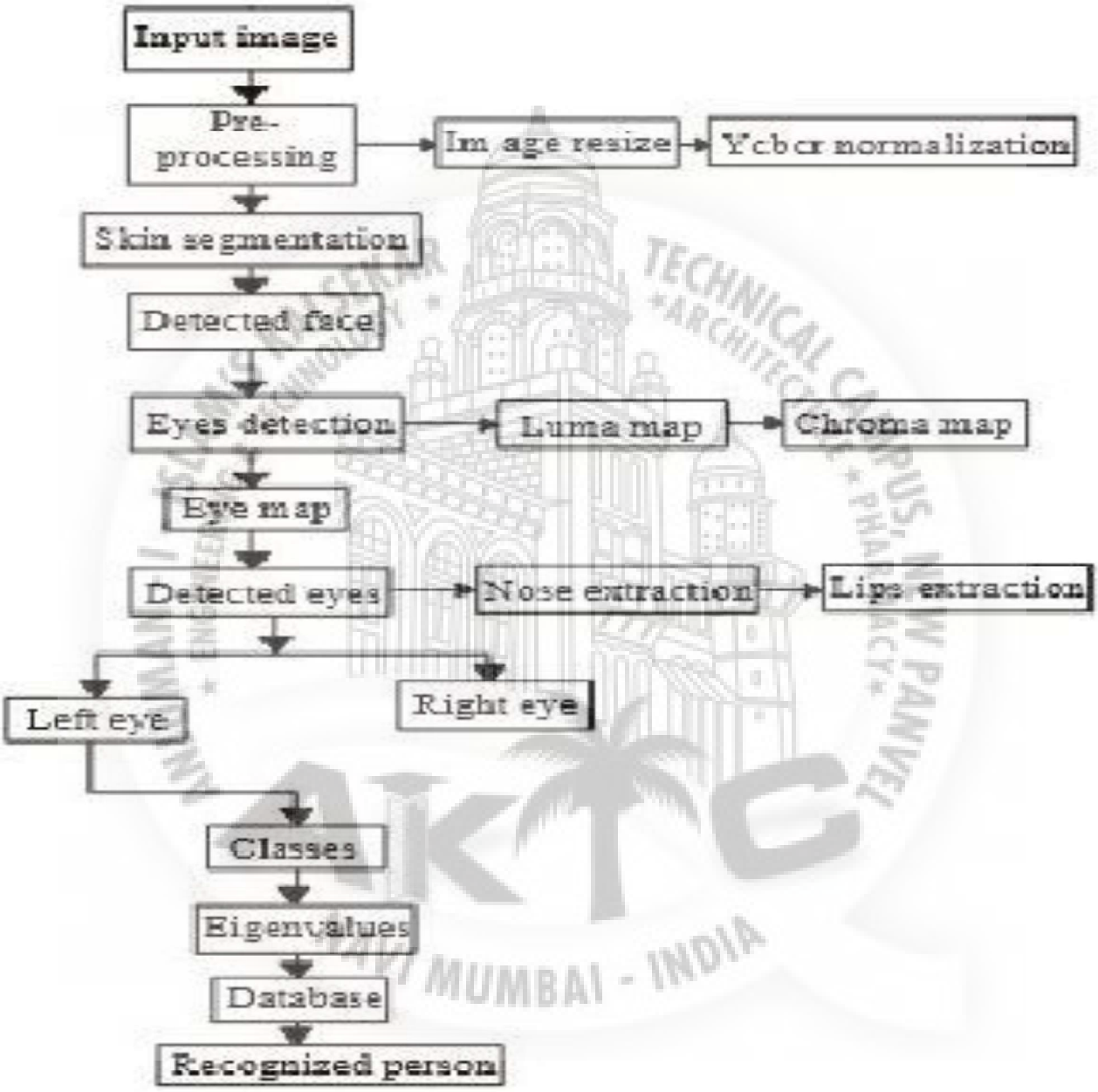


Fig2.System block diagram

Chapter 3

MATERIALS AND METHODS

The proposed technique is basically face detection features extraction and recognition technique. The system contains their main phases which are:

1. Face Detection
2. Features Extraction
3. Face Recognition

The basic working of the system is as follows (Fig. 1):

3.1.1 Face detection

Face detection: in short is: given an input image, to decide if there are people's faces present in that image. And for each present face, to know where each face is located (e. g. a

bounding box that encloses it) and possibly, also to know the position of the eyes, the nose, the mouth (known as face landmarks).

The face detection phase further involves steps which are

- Pre-processing
- Image resize
- YCBCR conversion and normalization
- Skin Segmentation
- Face detection

3.2 Pre-processing:

First an input image is presented to the system Fig. 2 represents an example input image

Fig3.2 Input Image

3.3 Image resizes:

When the method is offered with an input image the image is resized to 200×180 pixels in order to apply further processing.

Fig3.3 Image Resize

3.4 Ycbcr conversion and normalization:

The objective of this step is to compensate the lightening effects in the input image. The basic purpose is to attenuate noise from the image; this step works by converting the image to YCbCr color space and finding the maximum and minimum values of luminance Y. Next average value of Y is calculated. Then according to that average value the values of red, green and blue components are adjusted. The process works as

Y = Ycbcr conversion of the input image.

$$Y = 0.25\ R+0.504\ G+0.098B+16 \quad (1)$$

$$Cb = 0.439\ R-0.368\ G-0.071B+128 \quad (2)$$

$$Cr = 0.148\ R-0.291+0.439B+128 \quad (3)$$

First maximum and minimum values of Y are calculated using:

$$Ymin = min(min(Y) \quad (4)$$

$$Ymax = max(max(Y) \quad (5)$$

where Y=Ycbcr conversion of the input image.

After calculating the values, the subsequent step is to normalize the value of Y (luminance) which is done by:

$$Y = 255.0*(Y-Ymin)/(Ymax-Ymin) \quad (6)$$

After that the average value of Y is calculated that is used for lightning compensation purpose:

$$Yavg = sum\ (sum(Y))/(W*H) \quad (7)$$

where,

W = Width of the input image

H = Height of the input image

3.5 Skin segmentation:

In the next step the image after preprocessing is used as input and skin region is extracted using the YCbCr color space. This process basically utilizes the skin color in order to separate the face region. For this process the value of Cr (red difference Chroma component) is fixed in a range and only pixels in that particular range are picked as skin pixels of the face. The range defined here is 10<Cr & Cr<45. Figure 4 represents the segmented skin region.

The process of skin extraction works as follows:

$$\sum_{i=1}^{j} R(i).C(j) = 1 \qquad (8)$$

where,

j = Length (R, C)

R = Skin index row

C = Skin index column

3.6 Face detection:

After having the skin region, the face is isolated from that extracted skin region. Extra information is eliminated to obtain the region of concern i.e. the facial features region. Figure 5 represents the segmented face region.

3.7 Features extraction phase:

The second phase is the features extraction phase. This phase receives the detected face as the input. Once the face is detected from the image, subsequently step is to find the features from the face region being detected. The extracted features are:

- Eye
- Nose
- Lips

First of all, eyes are detected using chrome and luminance maps Fig 6 and 7. After eyes nose and lips are extracted using a formula centered on the space existing among both eyes.

For the features extraction purpose 7 points are extracted resting on the face which are:

- Left eye center
- Right eye center
- Eyes midpoint
- Nose tip
- Lips center
- Lips left corner
- Lips right corner

3.8 Eye extraction:

To extract the eyes from a human, face the information of dark pixels on the face are needed. As the eyes are different from the skin color, so using a color space and separating the colors could be a good way to locate the eyes. The YCbCr color space provides good information about the dark and light pixels present in an image. The color space called YCbCr has the tendency to demonstrate the eye area with a high Cb and low Cr values. To

detect the eye region 2 eye maps are constructed in the YCbCr color space which show the pixels with high Cb and Low Cr values for the eye region.

Fig3.8: Eye extraction

$$\text{Chroma Map} = 1/3\ (Cb\ /Cr + Cb\text{^}2 + (1\text{-}Cr)\text{^}2) \quad (9)$$

The luma chart is built by applying the morphological operations of erosion-dilation on the luminance component of the YCbCr image. These operations are applied to enhance the brighter and the darker pixels around the eye region in the luminance component of the image.

$$\text{Luma Map} = Ydialte\ /\ (Yerode\ -1) \quad (10)$$

Fig 3.8.1 Lumamap

The Y image is dilated and eroded by a structuring element. Then both ChromaMap and LumaMap (Fig. 3.8.1) are added together to make a Map. This map is further dilated to brighten the eye pixels. The eye region is detected by applying thresholding and erosion on the Map.

The detected eye region is then marked and the eye is extracted from the image. Figure 8 represents the whole extraction process.

3.9 Nose and lips detection:

The nose and lips are extracted on the basis of distance between the two eyes. The nose and lips are extracted by assuming that they lie at a specific ratio of the distance between the two eyes.

3.9.1Nose detection:

For nose extraction the distance between the two eyes is calculated as:

$$D=\sqrt{(L_x - R_x)^2 + \left(L_y - R_y\right)^2} \quad (11)$$

where,

D = Distance between the center points of two eyes

Lx = Left eye x coordinate

Rx = Right eye x coordinate

Ly = Left eye y coordinate

Ry = Right eye y coordinate

After having the eyes distance a formula is generated on the basis of that distance to extract the nose. It is assumed that nose lies at a distance of 0.55 of the eyes distance. This is done as:

$$N = (My+D)*0.55 \quad (12)$$

where,

N = Nose tip point

My = Eyes midpoint

D = Distance between the center points of two eyes

3..9.2 Lips extraction:

For lips extraction the three lips points extracted are:

- Lips center point
- Lips left corner point
- Lips right corner point

For lips extraction the distance between the two eyes is calculated as:

$$D=\sqrt{(L_x - R_x)^2 + (L_y - R_y)^2} \quad (13)$$

where,

D = Distance between the center points of two eyes

Lx = Left eye x coordinate

Rx = Right eye x coordinate

Ly = Left eye y coordinate

Ry = Right eye y coordinate

After having the eyes distance a formula is generated on the basis of that distance to extract the lips. For the lips center point, it is assumed that lips lie at a distance of 0.78 of the eyes distance. This is done as:

$$L = (My+D)*0.78 \quad (14)$$

Fig.3.9.2 Features extraction

where,

L = Center point of lips

My = The midpoint of the space among both eyes

D = Space among the center points of two eyes

Now for lips right and left corner point's extraction it is analyzed that the lips corners are almost located at a distance 0.78 of the eyes center points. These points are extracted as:

$$Ll = (Ly+D)*0.78 \quad (15)$$

where,

Ll = Left corner point of lips

Ly = Left eye y coordinate

D = Distance between the center points of two eyes

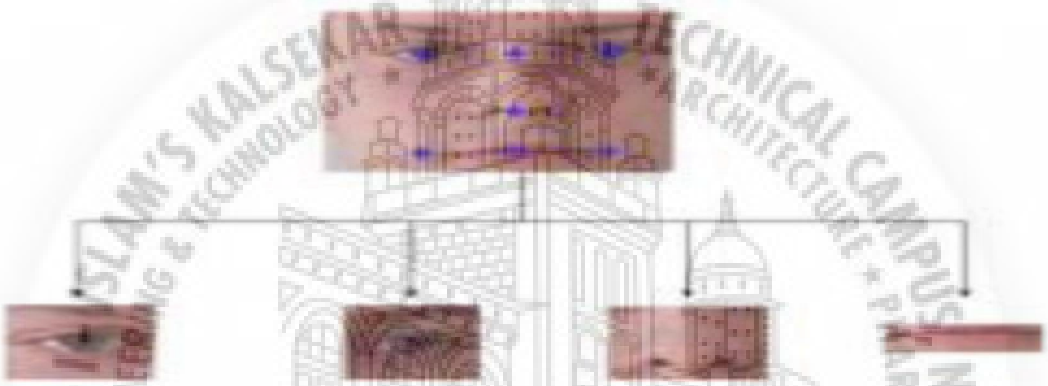


Fig.3.9.3 Output of features extraction phase

Similarly, the right corner point of mouth is extracted as:

$$Lr = (Ry+D)*0.78 \quad (16)$$

where,

Lr = Right corner point of mouth

Ry = Right eye y coordinate

D = Distance between the center points of two eyes

3.10 Face recognition phase:

The last phase of this system is the face recognition phase. The phase makes use of two algorithms for the recognition purpose which are:

- Eigen faces
- Fisher Faces

The recognition stage makes use of extracted eyes for the recognition process. Here a small portion of the face is chosen to perform the recognition so that it can save the computational time and to increase the accuracy. For this purpose, the chosen part from the face is the extracted left eye in order to recognize individuals.

The proposed approach to face recognition takes into account the fact that faces can be recognized using a smaller portion of the face. For this purpose, the region of left eye is picked and a face eye space is defined. The eye area is professed through the Eigen faces that are the Eigen-vectors of set of eyes of different individuals.

Most of the earlier exertion in the prospect of face identification using Eigen faces focus on the fact that the faces are recognized by calculating the Eigen faces of set of face features i.e., the whole face was considered and used for computing the Eigenvalues of different individuals. The proposed approach focuses only on the smaller region pf the face to recognize different individuals. Due to which the computational cost and time is efficient and reliable.

3.11 Calculating the eigenvalues:

As there is a figure of diverse persons in a library called database, there is ensemble of different individual images creating different points in the eye space. The eye images that are much alike cannot be randomly distributed in the eye space because the purpose is to differentiate different individuals despite the fact that they may have features that are quite alike. So in order to describe such images a slight measurement area is needed. The chief objective of PCA is to discover such vectors that can effectively distribute the images in the whole area. These vectors basically describe the eye space of different individuals.

Creation of Eigenvalues is the main thing in this process. Once these are created the next step seems like a pattern recognition process. There are M different training images of eyes of different individuals whose Eigenvalues are created and are distributed in the eye space. We have to pick out M1 Eigenvectors which have largest Eigenvalues associated with them. The process of choosing these Eigenvectors is entirely dependent on the Eigenvalues being calculated of different individuals. Whenever an input image is received, its Eigenvalue components are computed. A weight vector is used which is used to define the involvement of every Eigen face in the process of demonstrating the image to be judged.

	Detection rate (%)		Processing time (Sec)	
Facial features	Existing (Yuen *et al.*, 2010)	Proposed	Existing (Yuen *et al.*, 2010)	Proposed
Eyes	93.08	99	2.04	0.592
Mouth	95.83	99	0.46	0.712

Table.1: Features extraction rate comparison

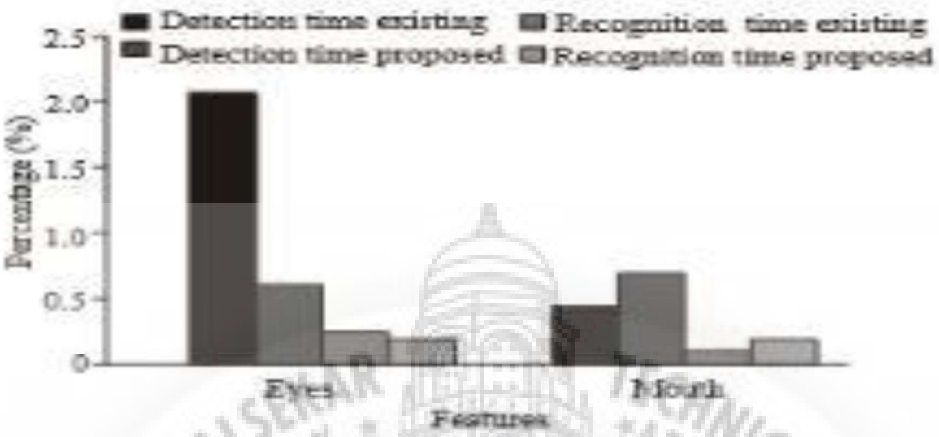


Fig.3.11: Features and recognition rate comparison through bar charts

Fig.3.11.1: Processing time comparison

The main purpose is to determine which included class most closely resembles to the input image. The concept of classes of Fisher faces is used to differentiate different individuals. For this purpose, classes equal to the figure of diverse persons present inside the database being used with the system. The best suited class to the input image is selected centered on the minimum Euclidian distance of the class from the input image.

Chapter 4

RESULTS AND DISCUSSION

In direction to check the performance, the projected system the database is tested against three databases which are:

- Self-selected AR database
- C Self-selected CVL database
- C Database containing 300 images of different individuals with illumination change

The results are compared with the existing technique results proposed by Yuen et al. (2010) which is based on template matching.

4.1 Results comparison:

4.1.1Results on the self-selected AR database:

For the comparison purpose system is tested with the same number of images i.e., 72 images, 3 images per person are used in the existing paper. 12 males and 12 females' images are used for this purpose and also with 30 males and 30 females each having 3 images. Table 1, Fig. 3.11 (for detection and recognition rate) and Fig. 3.11.1 (for processing time) shows that the features extraction rates and the processing time of proposed technique is far better than that of existing technique.

Similarly, Table 2, Fig. 3.11 (for detection and recognition rate) and Fig.3.11.1 (for processing time) show that recognition rate and recognition time of proposed technique is far better than that of existing technique.

Table 2: Recognition rate comparison

	Detection rate (%)		Processing time (Sec)	
Facial features	Existing (Yuen et al., 2010)	Proposed	Existing (Yuen et al., 2010)	Proposed
Eyes	79.17	97	0.24	0.1754
Mouth	81.39	75	0.10	0.194

Table 3: Results on 72 images tested with nose of individuals

Feature	Detection rate (%)
Nose	98

Table 4: Results on 72 images tested with right eye of individuals

Feature	Recognition rate (%)
Right eye	96

Table 5: Features results + processing time comparison

	Detection rate (%)	
Facial features	Existing (Yuen et al., 2010)	Proposed
Eyes	93.08	98
Mouth	95.83	97

In the features extraction process nose is also extracted of all the tested images. The extraction result of nose images (Detection rate) at 72 persons is 98 % (Table 3): The results on 72 images are also tested with right eye of individuals. The results (Detection rate) using right eye are 96% (Table 4):

4.1.2 Results on self-selected CVL database:

The self-selected CVL database contains 70 individuals with three images per person. The images selected were frontal faces with eyes visible and open. Table 5 shows that the detection rate of facial features using proposed technique is better than the existing technique.

Table 6: Recognition rate comparison

	Recognition rate (%)	
Facial features	Existing (Yuen *et al.*, 2010)	Proposed
Eyes 79.17	95	

Table 7: Nose extraction on CVL database

Feature	Detection rate (%)
Nose	97

Table 8: Face features extraction results (%)

	Detection rate (%)	
Facial features	Existing (Yuen *et al.*, 2010)	Proposed
Eyes	93.08	99

Table 9: Face recognition rate %

Facial features	Recognition rate (%)	
	Existing (Yuen *et al.*, 2010)	Proposed
Eyes	79.17	94

Chapter 5

Different Approach's

5.1 Adrian's approach

In this great article [5], Adrian Rosebrock solves the problem in Python using of **OpenCV's face_recognition** library, with the nn4.small2 pre-trained model from the **OpenFace** project and he achieves around **14 FPS** throughput rate in his MacBook Pro. The performance reported for this model is around **58.9 ms/frame** in an 8 core 3.70 GHz CPU. The published accuracy for this model claims to be around **93% LFW** on this "deep funneled" dataset.

This could possibly be an approach for our mobile application, using the OpenCV SDK for Android, but:

- Would it be **fast enough** on mobile?
- Will it **fit in the smartphone RAM**? (the **nn4.small2** model file size is more than 30 MB)
- What about **accuracy**? Adrian himself says on the post that he has some limitations and drawbacks with the accuracy of his implementation.

- If I take this way, I'm not sure how different could **Android** and **iOS** implementations be.

These are all big questions … **so let's see if there is another approach available …**

5.2 The FaceNet approach

FaceNet: A Unified Embedding for Face Recognition and Clustering.[2] **FaceNet** is a face recognition system developed in 2015 by researchers at Google that achieved the state-of-the-art results on a range of face recognition benchmark datasets **(99.63% on the LFW)**. This work introduces the novel concept of triplet loss.

In this great article [6], Jason Brownlee describes how to develop a Face Recognition System Using FaceNet in Keras. Although the model used is heavy, **its high accuracy is tempting** to try using it. Also, as **FaceNet is a very relevant work**, **there are available many very good implementations**, as well as pre-trained models. Perhaps, by applying post-training quantization, the model could be reduced and its speed would be good enough on mobile**…**

So, I decided to give it a chance and I converted **David Sandberg's** FaceNet implementation to TensorFlow Lite. I've chosen this implementation because is very well done and has become a facto-standard for FaceNet. I thought that the it was going to be an easy task, but I ran into several difficulties. I explain how I did it in this post.

Once I had my FaceNet model on TensorFlow Lite, **I did some tests with Python** to verify that it works. I took some images of faces, crop them out and computed their embeddings. **The embeedings matched their counterparts** from the original models. **I also noticed much lighter and faster execution** with the Lite version on my laptop's CPU.

They finally imported the Lite model in my Android Studio project to see what happened. What I found is that **the model works** fine, but **it takes around 3.5 seconds to make the inference** on my Google Pixel 3. The answers to the questions from the beginning, begin to be revealed.

Using the FaceNet, with TFLite we can:

- **Compare faces offline on mobile**
- **The comparison is guaranteed to be accurate**
- **As a baseline, the execution time takes around 3.5 seconds.**

Although not in real time, there are many useful applications that this way could be done, If the user is willing to wait a bit. But… nowadays as users, we want it all and we want it now, don't we? So is there any other alternative?

Well, if we want speed and lightness **we should give a try to a Mobile DNN Architecture**!

Chapter 6

Implementation

6.1 Using the MobileFaceNet :

MobileFaceNets [1] is a great work by researchers at Watchdata Inc. in Beijing, China. They presented a very efficient CNN model specifically designed for high-precision real-time face verification on mobile devices. They achieved **impressive speeds** with **very high accuracy** with a **model of just 4.0 MB**. The accuracy they obtained is very similar to that of other heavier models (such as FaceNet).

They looked for **some MobileFaceNet implementation to bring it to TensorFlow Lite**. Most available implementations are for PyTorch, which could be converted using the ONNX conversion tool. But since this tool is still in early stages of development, I opted for this excelent MobileFaceNet implementation on TensorFlow, from **sirius-ai**. Note: To convert the model the answers from this thread were very helpful.

The resulting file is very light-weight only 5.2 **MB**, really good for a mobile application.

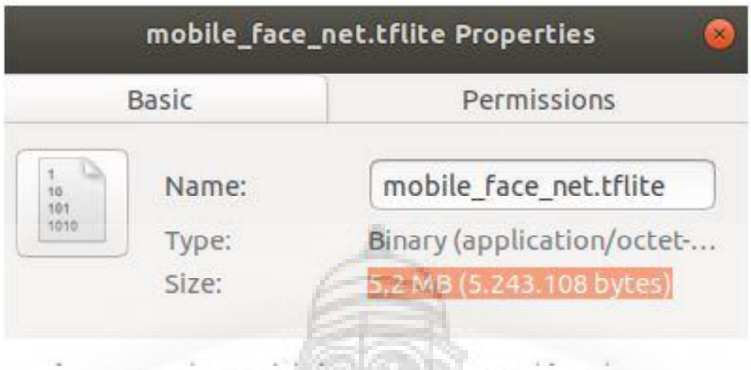


Fig.6 MobilefaceNet Model

Once I had my Lite Model They did some tests in Python to verify that the conversion worked correctly. And the results were good, so I was ready to get my hands on mobile code.

6.2 Creating the mobile application

We are going to modify the TensorFlow's object detection canonical example, to be used with the MobileFaceNet model. In that repository we can find the source code for Android, iOS and Raspberry Pi. Here we will focus on making it work on Android, but doing it on the other platforms would simply consist of doing the analogous procedure.

6.3 Adding the Face Recognition Step

The original sample comes with other DL model and it computes the results in one single step. For this app, we need to implement several steps process. Most of the work will consist in splitting the detection, first the face detection and second to the face recognition. For the face detection step we are going to use the Google ML kit.

Let's add the ML kit dependency to our project by adding the following line to the build.gradle file:

```
dependencies {

    ...

    // added MLKit dependencies for face detector
    implementation 'com.google.mlkit:face-detection:16.0.0'

}
```

Fig:

When the project finished sync, we are ready to use the FaceDetector into our DetectorActivity. The face detector is created with options that prioritize the performance over other features (e.g. we don't need landmarks for this application).

```
// Face detector
 private FaceDetector faceDetector;


 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);


   // Real-time contour detection of multiple faces
   FaceDetectorOptions options =
           new FaceDetectorOptions.Builder()
                   .setPerformanceMode(FaceDetectorOptions.PERFORMANCE_MODE_FAST)
                   .setContourMode(FaceDetectorOptions.LANDMARK_MODE_NONE)
                   .setClassificationMode(FaceDetectorOptions.CLASSIFICATION_MODE_NONE)
                   .build();


   FaceDetector detector = FaceDetection.getClient(options);

   faceDetector = detector;

 }
```

The original app defines two bitmaps (the rgbFrameBitmap where the preview frame is copied, and the croppedBitmap which is originally used to feed the inference model). We are going to define two additional bitmaps for processing, the portraitBmp and the faceBmp. The first is simply to rotate the input frame in portrait mode for devices that have the sensor in landscape orientation. And the faceBmp bitmap is used to draw every detected face, cropping its detected location, and re-scaling to 112 x 112 px to be used as input for our *MobileFaceNet* model. The frameToCropTransform converts coordinates from the original bitmap to the cropped bitmap space, and cropToFrameTransform does it in the opposite direction.

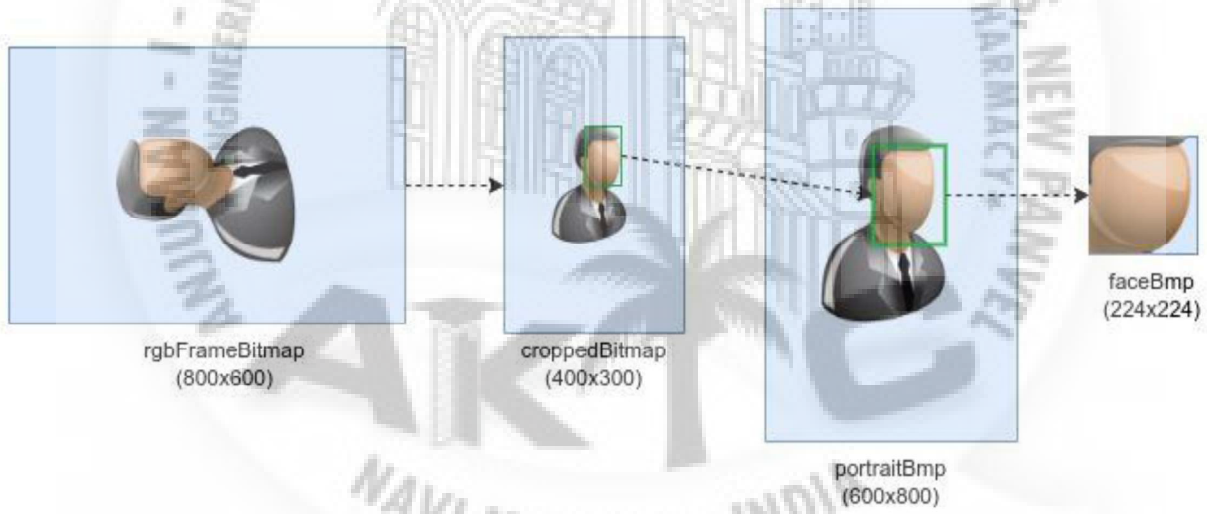


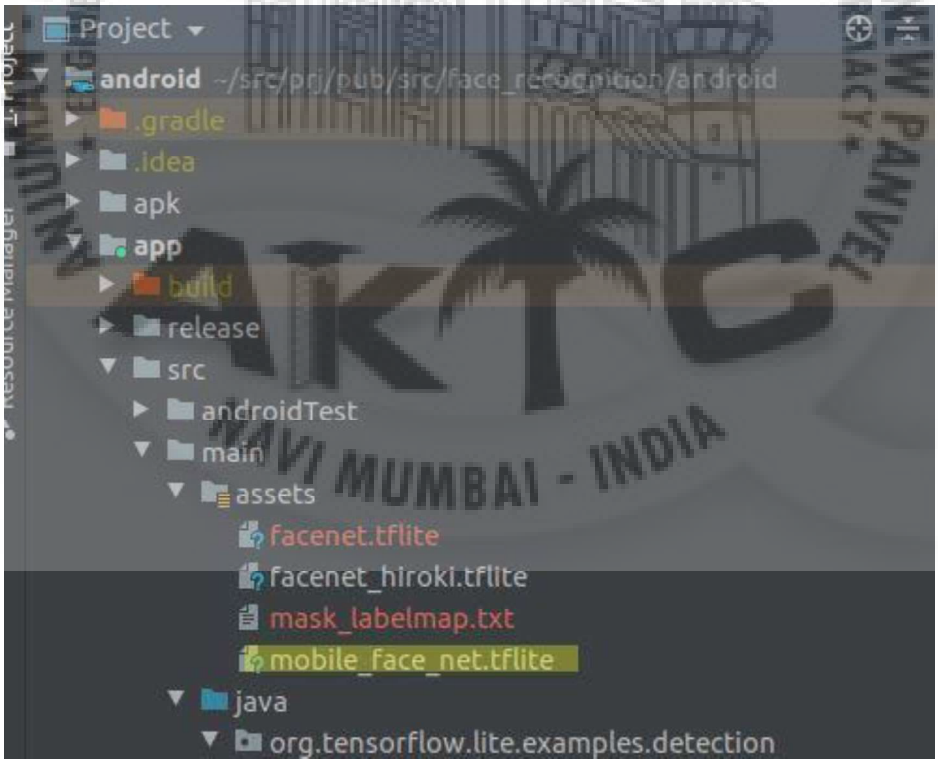
Fig:

When the frames arrive the face detector is used. Face detection is done on the **croppedBitmap**, since is smaller it can speed up the detection process.

When the faces are detected, the original frame is drawn in the **portraitBmp** bitmap. For each detected face, its bounding box is retrieved and mapped from the cropped space to portrait space. This way we can get a better resolution image to feed the recognition step. Face cropping is done by translating the portrait bitmap to the face's origin and scaling to match the DNN input size.

6.4 Adding the face recognition step:

First we need to add the TensorFlow Lite model file to the assets folder of the project:



And we adjust the required parameters to fit our model requirements in the **DetectorActivity** configuration section. We set the input size of the model to **TF_OD_API_INPUT_SIZE = 112**, and **TF_OD_IS_QUANTIZED = false**.

Let's change the name of the Classifier interface to **SimilarityClassifier** since now what the model returns is similarity, its behavior is a little different. It allows us to register recognition items in the dataset. We rename the confidence field as **distance,** because having confidence on the Recognition definition would require do something extra stuff. By now, we are going to use just distance as a measure of similarity, in this case it is the opposite to confidence (the smaller the value, the more sure we are that the recognition is from the same person), for example, if value is zero it is because it is exactly the same image.

```
// Changes made to the model interface
// This interface was renamed from Classifier to SimilarityClassifier
public interface SimilarityClassifier {

  // added method for registering samples into the dataset
  void register(String name, Recognition recognition);
  ...

  public class Recognition {

    // now we use distance instead of confidence (lower is better, eg should be zero for the sam
    private final Float distance;

    // extra field to store any data (we are going to use it to store the embeedings)
    private Object extra;

    ...
  }

}
```

Now, let's change the model implementation, by now we implement our dataset in the simplest possible way, that is a dictionary that stores the name of the person and its recognition (which has the embeedings). The **recognizeImage** method, is modified to retrieve the embeedings, and if necessary store them into the recognition result, when we have the embeeding, we just look for the nearest neighbor embeeding into the dataset by perfirming a linear search.

```
// changes made to the model implementation
public class TFLiteObjectDetectionAPIModel
        implements SimilarityClassifier {

  ...

  // the output size is 192
  private static final int OUTPUT_SIZE = 192;

  // added embeedings array buffer for model output
  private float[][] embeedings;

  // this is our "dataset", here we register the recognitions
  private HashMap<String, Recognition> registered = new HashMap<>();

  public void register(String name, Recognition rec) {
      registered.put(name, rec);
  }

  ...

  // looks for the nearest embeeding in the dataset (using L2 norm)
  // and retrurns the pair <id, distance>
  private Pair<String, Float> findNearest(float[] emb) {

    Pair<String, Float> ret = null;
    for (Map.Entry<String, Recognition> entry : registered.entrySet()) {
```

```
        final String name = entry.getKey();
        final float[] knownEmb = ((float[][]) entry.getValue().getExtra())[0];

        float distance = 0;
        for (int i = 0; i < emb.length; i++) {
              float diff = emb[i] - knownEmb[i];
              distance += diff*diff;
        }
        distance = (float) Math.sqrt(distance);
        if (ret == null || distance < ret.second) {
            ret = new Pair<>(name, distance);
        }
    }

    return ret;

}

@Override
public List<Recognition> recognizeImage(final Bitmap bitmap, boolean storeExtra) {

    // pre-process input into imgData
    ...

    Object[] inputArray = {imgData};
    Map<Integer, Object> outputMap = new HashMap<>();
```

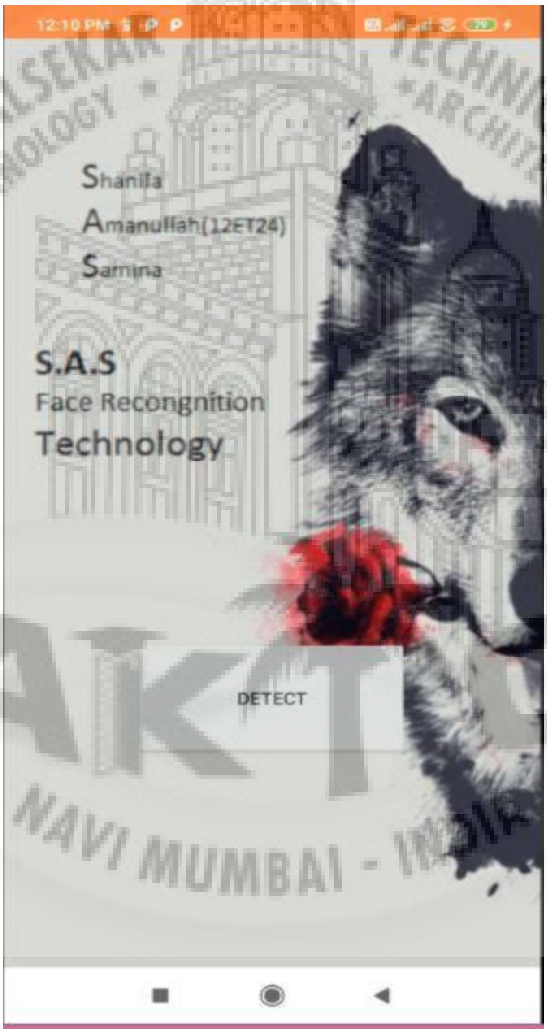
```
    embeedings = new float[1][OUTPUT_SIZE];
    outputMap.put(0, embeedings);

    // run the model
    tfLite.runForMultipleInputsOutputs(inputArray, outputMap);

    // looks for the nearest neighbor
    final Pair<String, Float> nearest = findNearest(embeedings[0]);
    final String name = nearest.first;
    final float distance = nearest.second;



    final ArrayList<Recognition> recognitions = new ArrayList<>(numDetectionsOutput);
    Recognition rec = new Recognition(
            id,
            label,
            distance,
            new RectF());

    recognitions.add( rec );

    if (storeExtra) {
        rec.setExtra(embeedings);
    }

    return recognitions;
}
```

The rest is pretty straightforward, all the code is provided and any additional details can be seen in the repository.

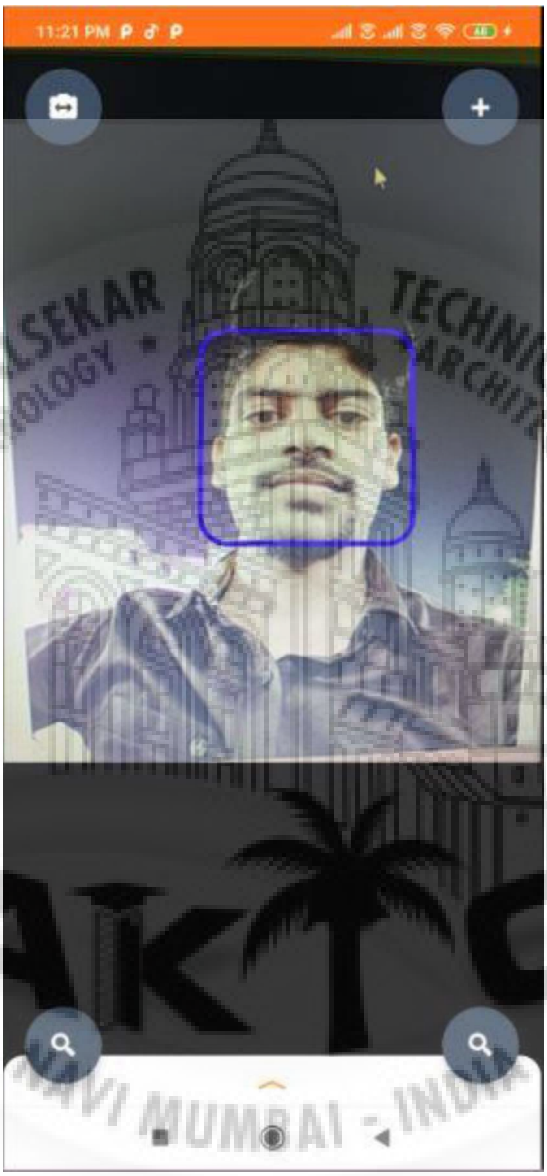
Chapter 7

Testing the App

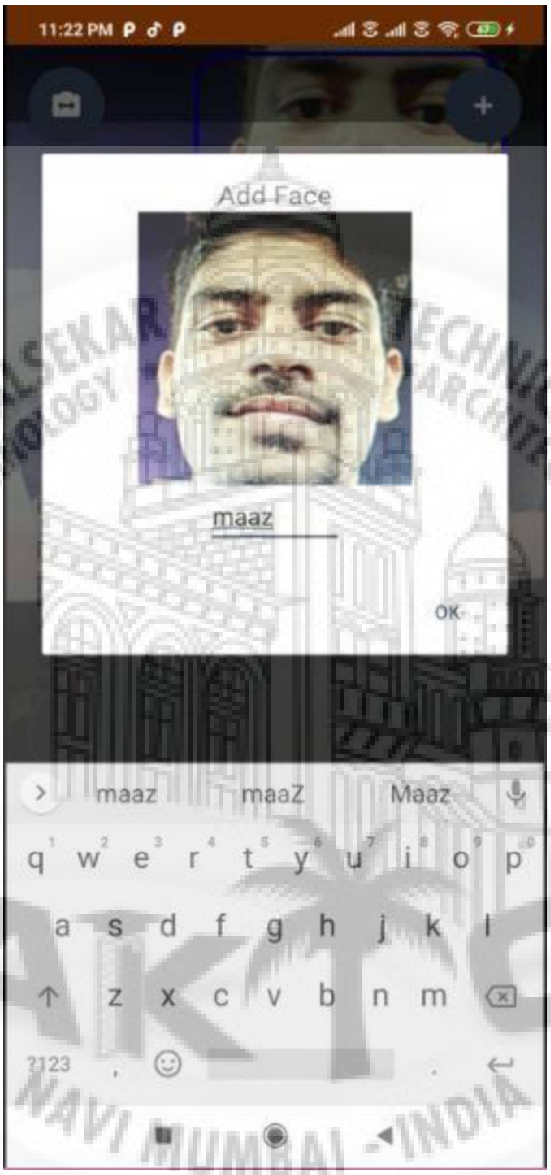
1.Home Screen



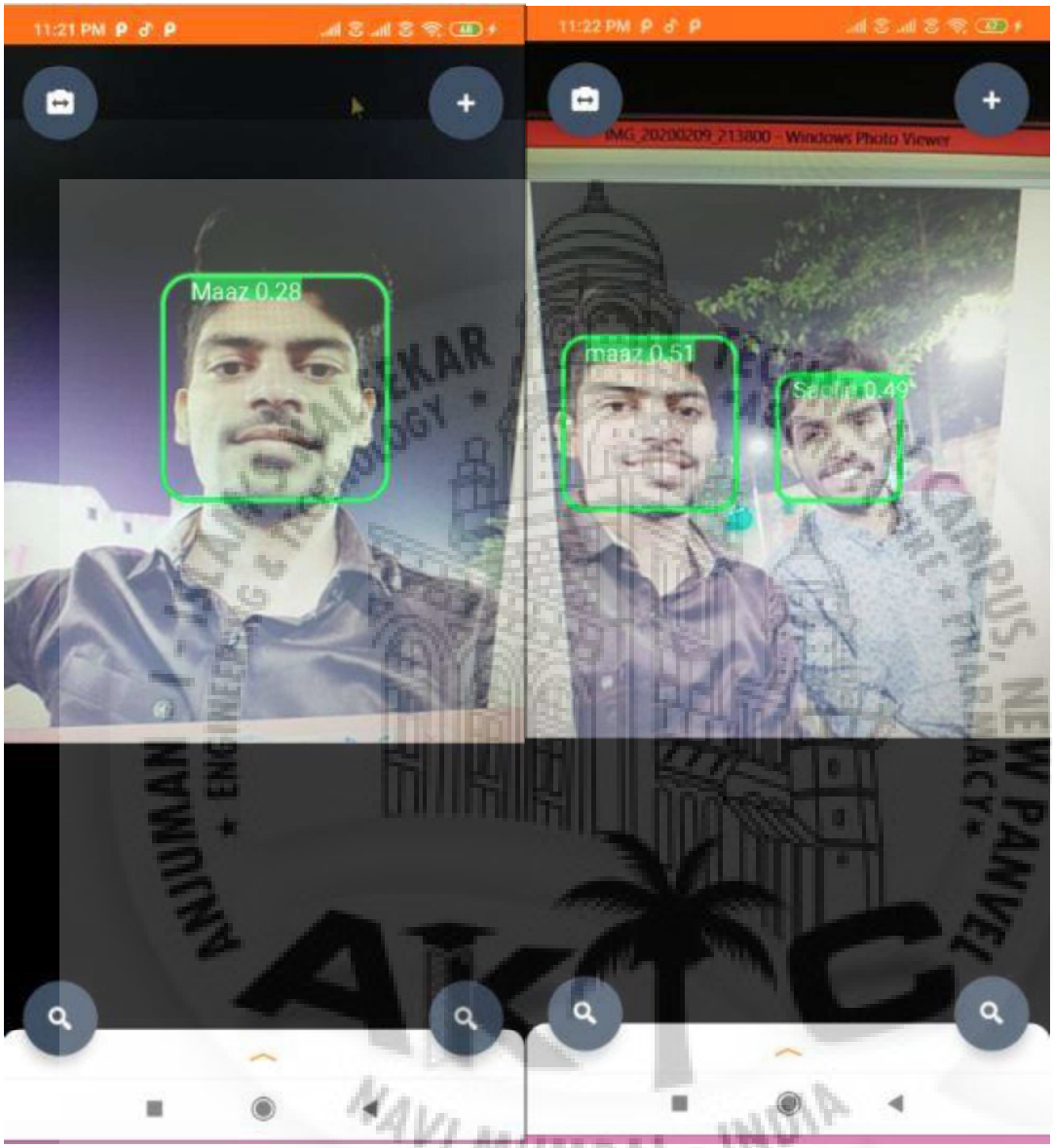
2.Fcae Detect:



4. Add Faces



5. Faces Recognitions:



Improvements and future work

Face pre-processing

Ideally the face should be aligned and whitened, before use. In my case I am using the result as it comes from ML Kit, just scaling to the required input size and that's it. This is a very important improvement point, but in Java or Kotlin it might be more laborious than in Python. Undoubtedly, this would allow improving the accuracy of the results (although even without aligning, the results are very good). This is something I will add for future work.

References

[1]: Chen, Sheng, et al.["MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices] — Chinese Conference on Biometric Recognition. — Apr, 2018

[2]: F. Schroff, et al. ["FaceNet: A unified embedding for face recognition and clustering,"] — 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 815–823 — Jun, 2015

[3]: Satya Mallick, at al. [Face Recognition: An Introduction for Beginners] — learnopencv — https://www.learnopencv.com/face-recognition-an-introduction-for-beginners/ — Apr, 2019

[4]: Adrian Rosebrock. [Face Alignment with OpenCV and Python] — pyimagesearch — https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/ — May, 2017

[5]: Adrian Rosebrock. [OpenCV Face Recognition] — pyimagesearch — https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/
pyimagesearch — Sep, 2018

[6]: Jason Brownlee. [How to Develop a Face Recognition System Using FaceNet in Keras] — machinelearningmastery — https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/ — June, 2019

ANJUMAN - I - ISLAM'S KALSEKAR
TECHNICAL CAMPUS, NEW PANVEL
ENGINEERING & TECHNOLOGY
ARCHITECTURE
PHARMACY
AIKTC
NAVI MUMBAI - INDIA