# Face Recognition Software Development

## (Prototype Version)

陈鹏 12330029 软件工程综合实验

# Content

## 1. Abstract

Face recognition is a hot research and development direction today, and it has a wide range of applications in security, finance, and tourism. This book comprehensively and systematically introduces the techniques behind “brushing face”, including algorithmic principles and implementation techniques related to face detection, face recognition, and face retrieval. Since different image shows one's different facial expression, the image let us know who this person is. So we also use the image to identity someone's status, and then we can use this to get access into some place or buy something in our life.

## 2. Introduction

We used a variety of methods to carry out simple face recognition experiments, and initially understood and mastered the main principles of face recognition, processing flow, main methods and some models. We used the training set in the data set to go through various methods. Train in the model and get the features of the face, and use these features to construct a dataset with feature representations to process the test set. Among them, we will explain and analyze some methods in the experiment, and we will also make a brief summary and analysis of the advantages and disadvantages of the method. Two main methods we use are OpenIMAJ and FaceNet. OpenIMAJ is an award-winning set of libraries and tools for multimedia content analysis and content generation. OpenIMAJ is very broad and contains everything from state-of-the-art computer vision (eg, SIFT descriptors, salient region detection, face detection, etc.) and advanced data clustering, through to software that performs analysis on the content, layout and structure. FaceNet is a versatile system that can be used for face verification (is it the same person?), identifying (who is this person?) and clustering (looking for similar people?). FaceNet's approach is to learn to map images to Euclidean space through convolutional neural networks. The spatial distance is directly related to the similarity of the picture: the different images of the same person have a small spatial distance, and the images of different people have a large distance in the space. As long as the mapping is determined, the associated face recognition task becomes very simple.

We describe the limitation of this development in 5 Vs of big data concept as followed.

For velocity, in FaceNet, because the number of iterations is large, and the training data set used by the model is specially downloaded, the data is relatively large, so the method training time is relatively long, reaching more than ten hours, and the other method OpenIMAJ uses the own data set. The training data is relatively small, so it took only two minutes to complete the training.

For veracity, our data set uses two different data sets. The datasets and code used are described in more detail below. We will also upload these. The experimental accuracy of the two methods will also be shown below.

For value, the methods we use are some more general and effective models, and the effect is quite remarkable in face recognition. The data sets, models and codes we use are very reusable after we have processed and optimized them. Run smoothly in the new environment.

For volume, because the image data is difficult to store in Hadoop or RapidMiner, we just store the data in binary form in the mysql database.

For variety, we used two different methods, and the dataset we used also contained two different datasets, one larger and the other smaller. The code language we use also contains two. In OpenIMAJ, the language we use is Python, but we are programming in OpenIMAJ using java.

## 3. Related Work

Face recognition is a research topic with both scientific research value and broad application prospects. A large number of researchers in the world have achieved fruitful research results for decades, and automatic face recognition technology has been successfully applied under certain limited conditions. These results have deepened our understanding of the issue of automatic face recognition, especially its challenging understanding.

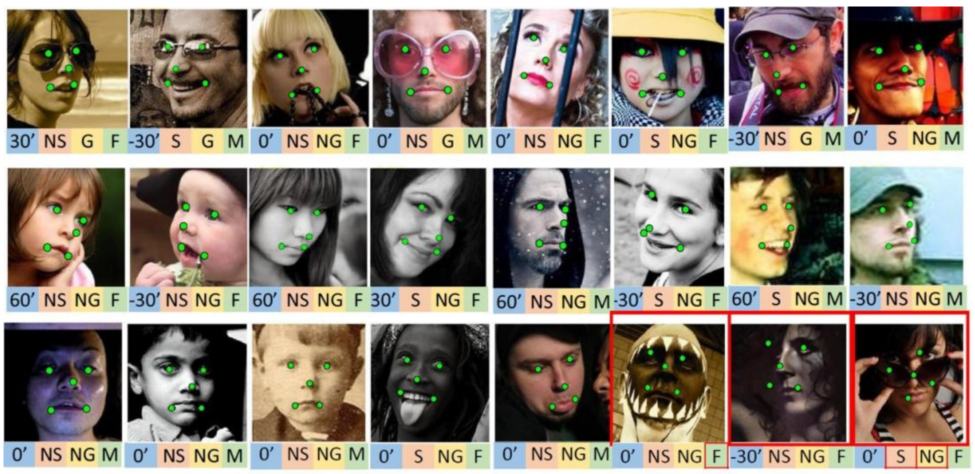
## 4. Dataset

In our project, three dataset was used here: MTFL(Multi-Task Facial Landmark dataset), LFW(Labeled Faces in the Wild Home), the ORL face database. And we have two ways to finding the head on the images: OpenIMAJ and MTCNN.

**LFW and MTFL**

Labeled Faces in the Wild is a database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. The only constraint on these faces is that they were detected by the Viola-Jones face detector.

And MTFL dataset is the extension of LFW dataset. This dataset contains 12,995 face images collected from the Internet. The images are annotated with five facial landmarks, attributes of gender, smiling, wearing glasses and head pose.



🟦 Pose estimation (0◦, ±30◦, ±60◦) 🟨 G/NG = with / without glasses

🟧 S/NS = smiling / not smiling 🟩 M/F = male / female

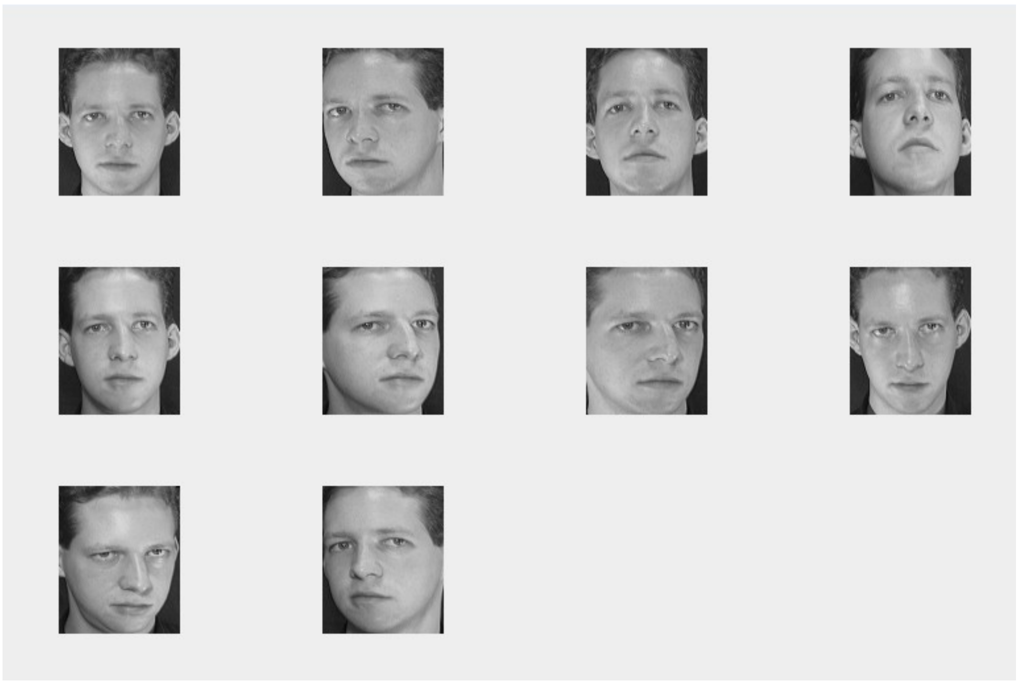
The dataset is divided into two parts: training and testing. The annotations are stored in according text files (training.txt and testing.txt). Each line in the text file is for one face image. According to these two text files, we extract the images and their annotations and store them in Mysql.

- Extract image path and other information(x1...x5,y1..y5, gender, smile, wearing, glasses, head pose) of each image based on the text files.

- Access images according to their path and save the images and their information.

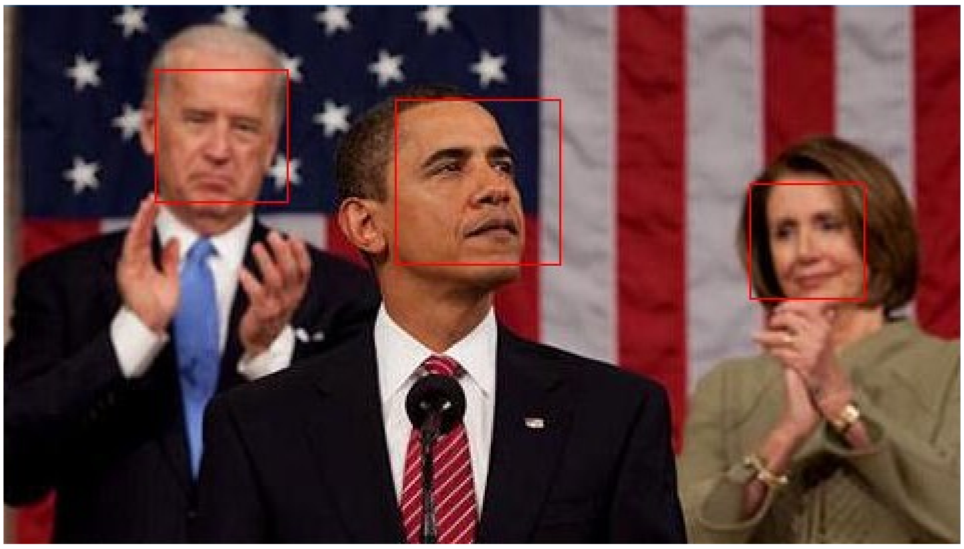
## The ORL face database

This dataset contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, UK.There are 10 different images of 40 distinct subjects. For some of the subjects, the images were taken at different times, varying lighting slightly, facial expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All the images are taken against a dark homogeneous background and the subjects are in up-right, frontal position (with tolerance for some side movement). The files are in PGM format and can be conveniently viewed using the 'xv' program. The size of each image is 92x112, 8-bit grey levels. The images are organized in 40 directories (one for each subject) named as “sX”, where X indicates the subject number (between 1 and 40). In each directory, there are 10 different images of the selected subject named as “Y.pgm”, where Y indicates which image for the specific subject (between 1 and 10).



## Finding head by OpenIMAJ

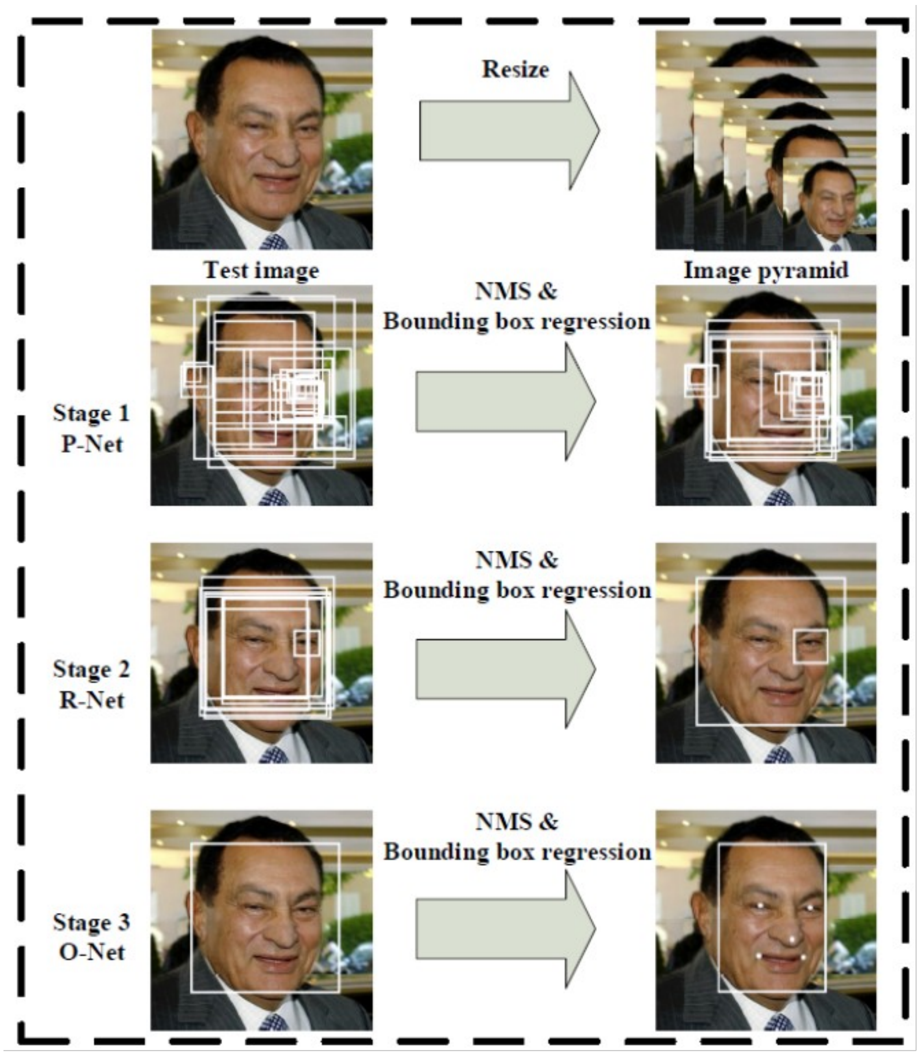
We use the function “HaarCascadeDetector” in OpenIMAJ to find person’s face in a large image. It efficiently get the face in a large image.



However, this function can only get the face, but can’t get the hair and ears. In some way it lost some feature. Which make the accuracy only 0.782. So we need to find another way to find the head.

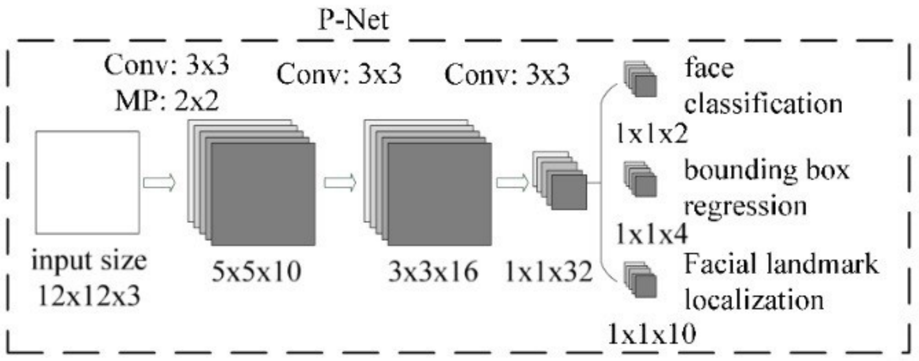
## Finding head by MTCNN

We used MTCNN model for face detection. MTCNN is a deep learning model of multi-task cascaded CNN for face detection, which takes into account both face border regression and key point detection. The overall network architecture of MTCNN is shown in the following figure:



First, the pictures will be scaled to different sizes according to different scaling ratios, forming a feature pyramid of the pictures. PNet mainly obtains the regression vectors of candidate windows and boundary boxes of face regions. The candidate window is calibrated by regression with the boundary box, and then the highly overlapping candidate frames are merged by non-maximum suppression (NMS). RNet will be trained by PNet candidate box in RNet network, then fine-tune candidate form by using the regression value of boundary box, and then remove overlapping form by NMS. ONet function is similar to RNet function. It only removes overlapping candidate windows and displays five key point locations at the same time.

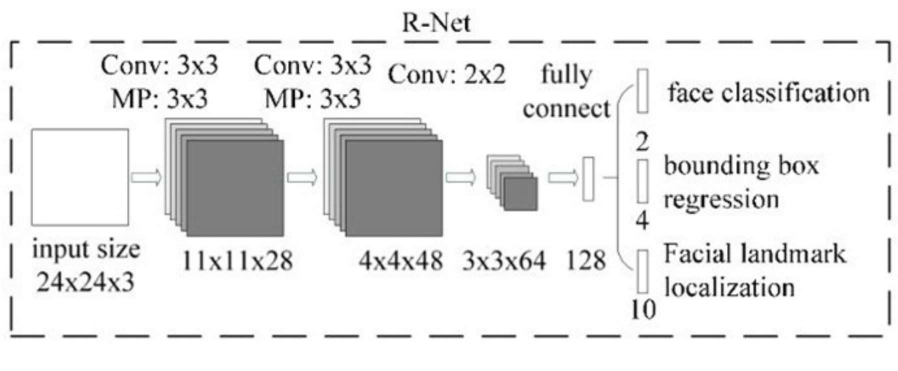
PNet's network structure is a fully convoluted neural network structure, as shown in the following figure:



The input of the training network is a 12*12 image, so the training data of PNet network need to be generated before training. The training data can generate a series of bounding boxes by computing with the IOU of Guarantee True Box. The training data can be obtained by sliding window or random sampling. The training data can be divided into three positive samples, negative samples and intermediate samples. The IOU of positive-positive sliding window and Guarantee True Box is greater than 0.65, the IOU of negative sample is less than 0.3, and the IOU of intermediate sample is greater than 0.4 and less than 0.65.

Then the bounding box resize is transformed into a 12*12 size picture, which is transformed into a 12*12*3 structure, and the training data of PNet network is generated. The training data are processed by 10 3*3*3 convolution cores and 2*2 Max Pooling (stride = 2) operations to generate 10 5*5 feature maps. Then 16 3*3 feature maps are generated by 16 3*3*10 convolution cores. Thirty-two 1*1 feature maps are generated through 32 3*3*16 convolution kernels. Finally, for 32 1*1 feature maps, two 1*1 feature maps can be generated through two 1*1*32 convolution kernels for classification; four 1*1*32 convolution kernels can generate four 1*1 feature maps for regression frame judgment; and 10 1*1*32 convolution kernels can generate 10 1*1 feature maps for face contour point judgment.

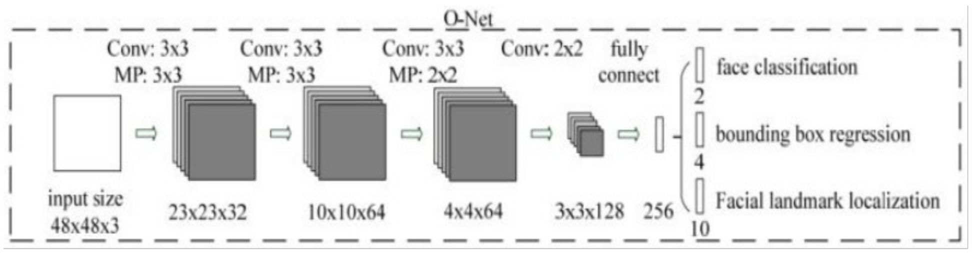
The model structure of RNet is as follows:



The model inputs 24*24-size images and generates 28 11*11 feature maps through 28 3*3*3 convolution cores and 3*3 (stride=2) Max pooling;

generates 48 4*4 feature maps through 48 3*3*28 convolution cores and 3*3 (stride=2) Max pooling; generates 64 3*3 feature maps through 64 2*48 convolution cores; and transforms 3*3*64 feature maps into 128-size full-connection layer. The problem of classification of regression box is converted to the full-connection layer of size 2; the problem of location regression of bounding box is converted to the full-connection layer of size 4; and the key points of face contour are converted to the full-connection layer of size 10.

ONet is the last network in MTCNN, which is used to make the final output of the network. ONet's training data generation is similar to RNet, and the detection data is the bounding boxes detected by pictures after passing through PNet and RNet networks, including positive samples, negative samples and intermediate samples. The model structure of ONet is as follows:



The input of the model is a 48*48*3 image, which is converted to 32 23*23 feature maps by 32 3*3*3 convolution cores and 3*3(stride=2) Max pooling, 64 10*10 feature maps by 64 3*3*32 convolution cores and 3*3(stride=2) Max pooling, 64 3*3*64 convolution cores and 3*3(stride=2) Max pooling, 64 4*4 feature maps by 64 3*3*64 convolution cores and 3*3(stride=2) Max pooling. 128 convolution kernels of 2*2*64 are converted to 128 feature maps of 3*3; the full-link layer of 256 size is converted by full-link operation; it is better to generate regression box classification features of size 2; regression features of regression box location of size 4; regression features of face contour location of size 10.

## 5. Face Detection With OpenIMAJ

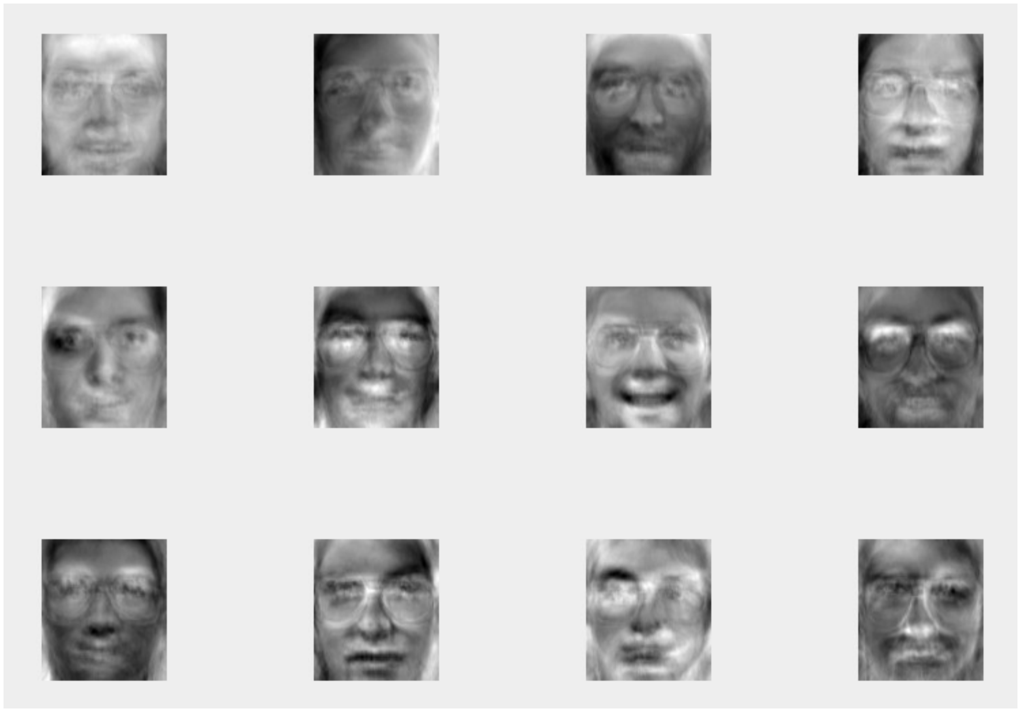
### Algorithm Implementation

We use OpenIMAJ as our main tools to do the face detecting. We divide the ORL face database into two part. 5 images at the training set and other 5 images as the testing set.

We implement the earliest face recognition algorithms called “Eigenfaces”. The basic idea behind the Eigenfaces algorithm is that face images are "projected" into a low dimensional space in which they can be compared efficiently. The hope is that intra-face distances (i.e. distances between images of the same person) are smaller than inter-face distances (the distance between pictures of different people) within the projected space (although there is no algorithmic guarantee of this).

The lower dimensional space used by the Eigenfaces algorithm is actually learned through a process called Principle Component Analysis (PCA).

We use a process called Principle Component Analysis (PCA) to get the lower dimensional space. Eigenfaces will really only work well on (near) full-frontal face images. In addition, because of the way Eigenfaces works, the face images we use must all be the same size, and must be aligned (typically such that the eyes of each subject must be in the same pixel locations). That is why we need to find a better head-finding algorithm before. One way of thinking about how we use the basis is that any face image can literally be decomposed as weighted summation of the basis vectors, and thus each element of the feature we'll extract represents the weight of the corresponding basis vector. This of course implies that it should be possible to visualize the basis vectors as meaningful images.



After we build a basis set, using a Map of Strings (the person identifier) to an array of features (corresponding to all the features of all the training instances of the respective person), we can start to use the model. We input a new image into the model, using the model to extract the feature. Then calculate the distant of the feature with all the feature in the basis set. Finding the closest one to estimate which person they belong to.

At last we get an accuracy 0.782 with OpenIMAJ’s finding head algorithm and 0.975 with the finding head algorithm which has been optimize.

## Discussion

However, this algorithm can not recognize the face never be learned or extracted features and save into the basis set.

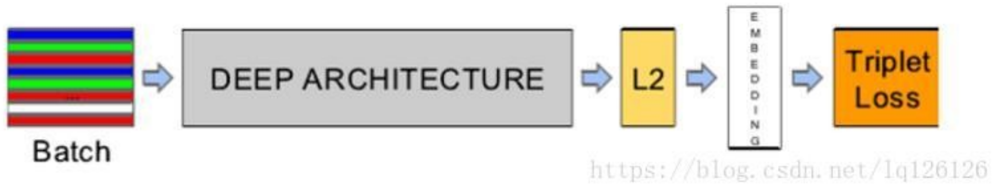
It’s time required is larger and larger as the basis set bigger and bigger. If we want to recognize more face, it is inevitable that the basis set would be bigger.

What’s more, the algorithm is easy to get wrong result is the image which we use to extracted the features and save into the basis set is not diverse and comprehensive enough.

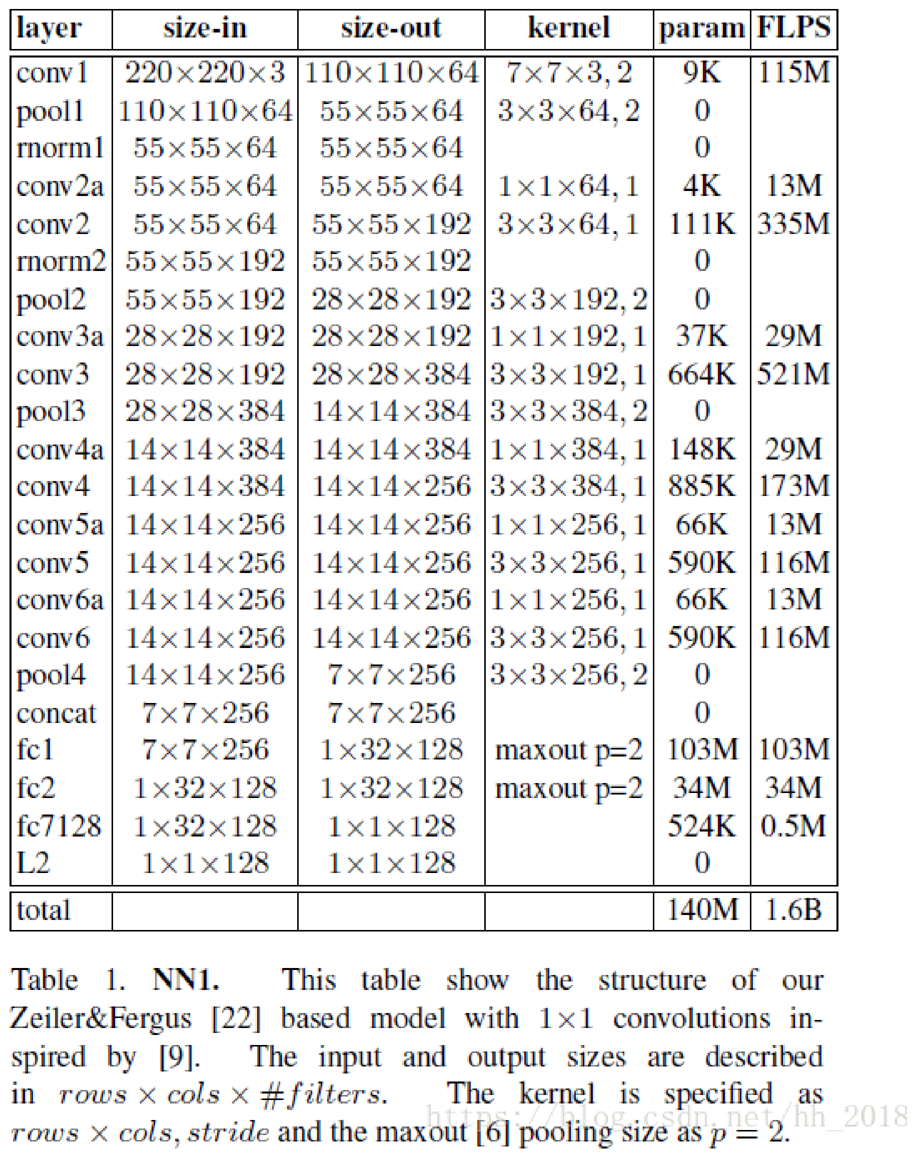
So we are going to try other algorithm. For example, face recognizing with deep learning.

## 6. Face Detection using FaceNet

We used a framework called FaceNet. FaceNet is different from the traditional CNN method. Traditional CNN is processed by network, and then the processed structure is classified by SVM method. In this method, the feature is transformed into a point on the Euclidean plane by learning directly, and then judged by comparing the distances between points.
The execution of faceNet is shown in the figure:



From the figure above, we can see that the network structure is a black box process. At present, there are mainly two kinds of network structures. One is the network structure proposed by Zelier &amp; Fefgus, which has 22 layers and 140 M parameters. The other is based on Google's Inception model. The parameters of the model are small. The network structure of NNS1 and NNS2 is generated by reducing the parameters and size of Inception model. Because the parameters of NNS1 and NNS2 are small, they can be used in mobile terminals. The first network structure is shown in the following figure.

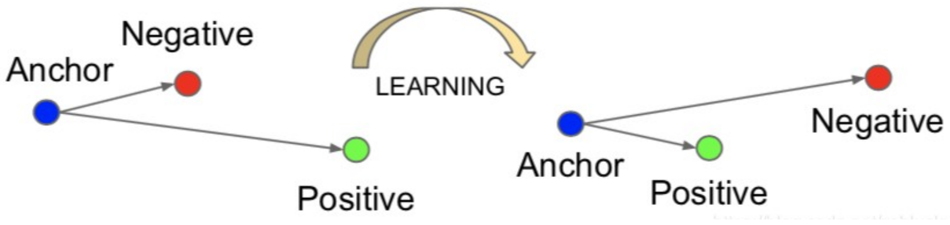


| layer | size-in | size-out | kernel | param | FLPS |
|---|---|---|---|---|---|
| conv1 | 220×220×3 | 110×110×64 | 7×7×3, 2 | 9K | 115M |
| pool1 | 110×110×64 | 55×55×64 | 3×3×64, 2 | 0 | |
| rnorm1 | 55×55×64 | 55×55×64 | | 0 | |
| conv2a | 55×55×64 | 55×55×64 | 1×1×64, 1 | 4K | 13M |
| conv2 | 55×55×64 | 55×55×192 | 3×3×64, 1 | 111K | 335M |
| rnorm2 | 55×55×192 | 55×55×192 | | 0 | |
| pool2 | 55×55×192 | 28×28×192 | 3×3×192, 2 | 0 | |
| conv3a | 28×28×192 | 28×28×192 | 1×1×192, 1 | 37K | 29M |
| conv3 | 28×28×192 | 28×28×384 | 3×3×192, 1 | 664K | 521M |
| pool3 | 28×28×384 | 14×14×384 | 3×3×384, 2 | 0 | |
| conv4a | 14×14×384 | 14×14×384 | 1×1×384, 1 | 148K | 29M |
| conv4 | 14×14×384 | 14×14×256 | 3×3×384, 1 | 885K | 173M |
| conv5a | 14×14×256 | 14×14×256 | 1×1×256, 1 | 66K | 13M |
| conv5 | 14×14×256 | 14×14×256 | 3×3×256, 1 | 590K | 116M |
| conv6a | 14×14×256 | 14×14×256 | 1×1×256, 1 | 66K | 13M |
| conv6 | 14×14×256 | 14×14×256 | 3×3×256, 1 | 590K | 116M |
| pool4 | 14×14×256 | 7×7×256 | 3×3×256, 2 | 0 | |
| concat | 7×7×256 | 7×7×256 | | 0 | |
| fc1 | 7×7×256 | 1×32×128 | maxout p=2 | 103M | 103M |
| fc2 | 1×32×128 | 1×32×128 | maxout p=2 | 34M | 34M |
| fc7128 | 1×32×128 | 1×1×128 | | 524K | 0.5M |
| L2 | 1×1×128 | 1×1×128 | | 0 | |
| total | | | | 140M | 1.6B |

Table 1. **NN1.** This table show the structure of our Zeiler&Fergus [22] based model with 1×1 convolutions inspired by [9]. The input and output sizes are described in $rows \times cols \times \#filters$. The kernel is specified as $rows \times cols, stride$ and the maxout [6] pooling size as $p = 2$.

In the training of the model, a method called triple loss is adopted. Its principle is as follows: the distance between groups should be as small as possible, and the distance between groups should be as large as possible. Its greatest advantage is that it is in place in one step, directly by quantifying the distance between the 128D vectors mapped by the two faces through the network to perform tasks such as verification, recognition and so on. At the same time, the face alignment operation in other face recognition algorithms is omitted. For example, backbone network is a classical classification network such as VGG, Google Net, etc. When doing face verification, we need to extract fc_layer or feature map of a convolution layer as face features and then compare them to prove that "you" is "you", which is very indirect and inefficient (there are many features), but faceNet does not have this problem.

The diagram of triple loss is as follows:



According to the above principle, the corresponding loss function is defined as:

$$\sum_{i}^{N} \left[ \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \quad . \tag{2}$$

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2 \quad . \tag{3}$$

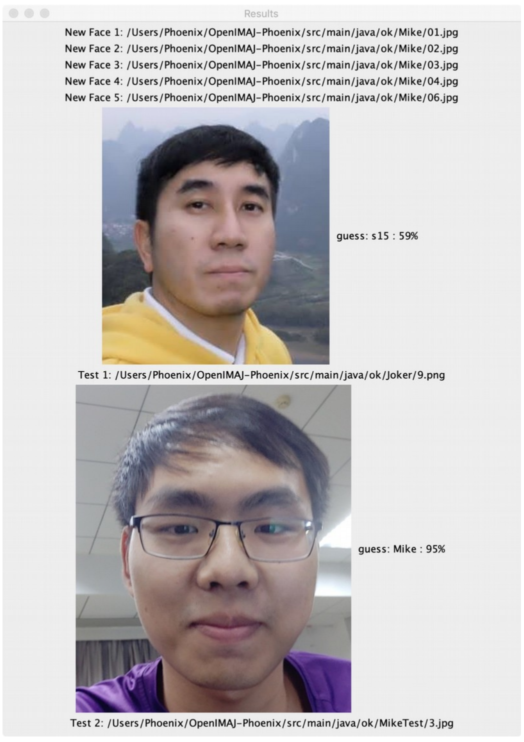
The goal of learning is to minimize the above loss functions, for which finding all ternary combinations is a huge workload. So in the process of implementation, we choose the most unlike one in position and the most similar one in negative. Because the loss distance calculated in this way is the largest distance, it is enough to optimize the loss function. When the value calculated in this way can meet the requirements, other smaller ones can also meet the requirements. There is also a disadvantage when using the above method to select triples. When selecting the nearest and farthest elements, it also needs to traverse all samples, and traverse all samples has a great workload. In this regard, batch search can be used. Because the execution of images is batch, we can find the corresponding positive and negative samples when processing each batch of images. In order to ensure that the data selected by this method are reasonable. When generating the corresponding batch of pictures, make sure that each person has an average of 40 pictures and randomly add counterexamples (I don't know why). At the same time, the semi-hard constraints are followed when selecting negative:

FaceNet performance:

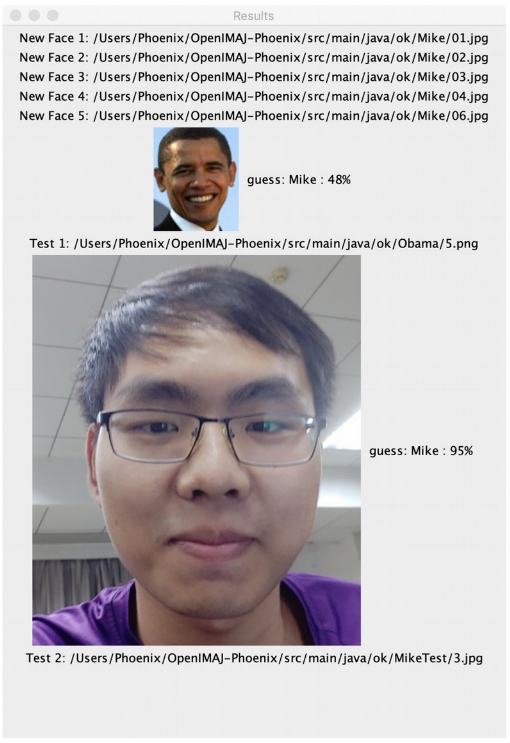
The accuracy of this method on LFW dataset has reached 99.6%, and it is the best record of detection on LFW dataset.

## 7. Software Development

Regarding to this studies we developed a prototype system as described in the way of software development to fulfill this research study. The concept of software development was described by Unified Modeling Language (UML). The objective is to implement face detection technologies studied in this research for face recognition purpose.



```
New Face 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/01.jpg
New Face 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/02.jpg
New Face 3: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/03.jpg
New Face 4: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/04.jpg
New Face 5: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/06.jpg
guess: s15 : 59%
Test 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Joker/9.png
guess: Mike : 95%
Test 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/MikeTest/3.jpg
```

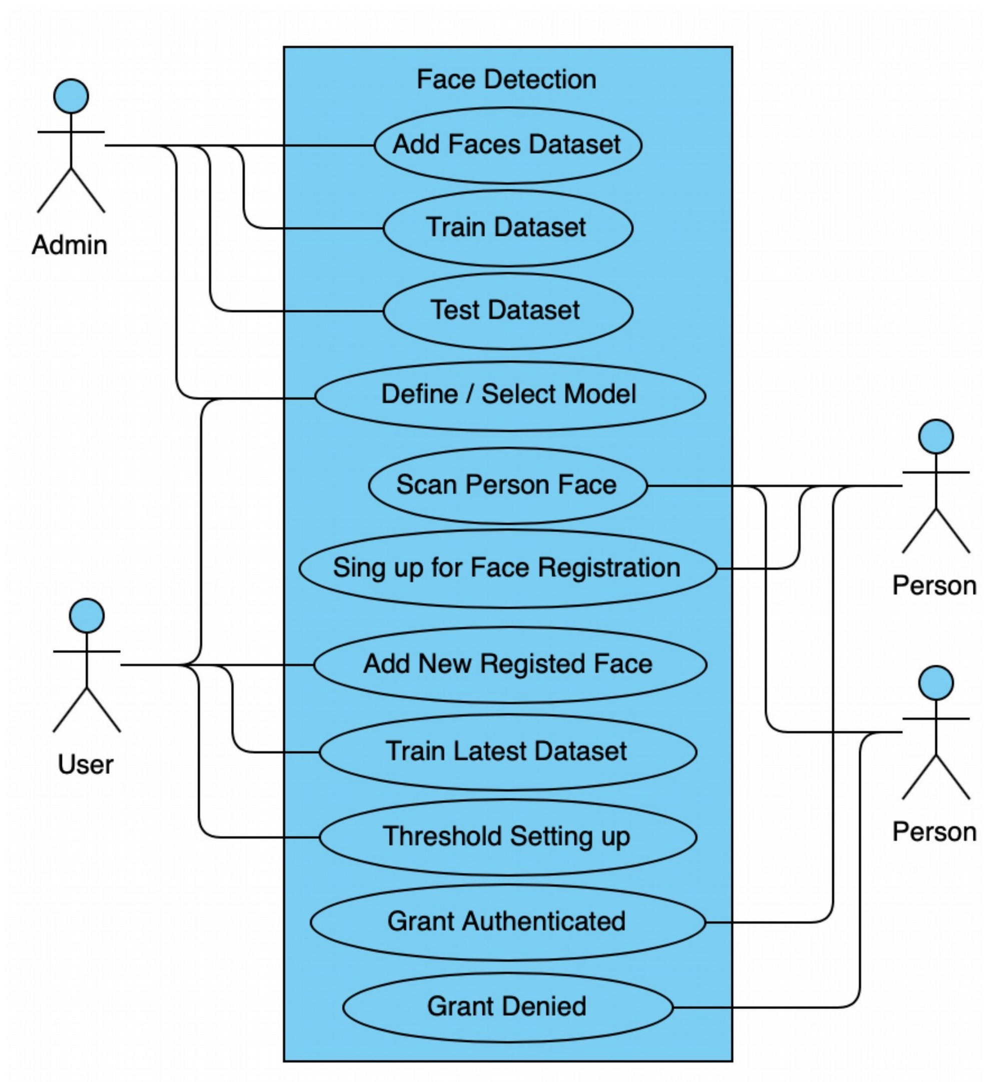


```
New Face 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/01.jpg
New Face 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/02.jpg
New Face 3: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/03.jpg
New Face 4: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/04.jpg
New Face 5: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/06.jpg
guess: Mike : 48%
Test 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Obama/5.png
guess: Mike : 95%
Test 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/MikeTest/3.jpg
```

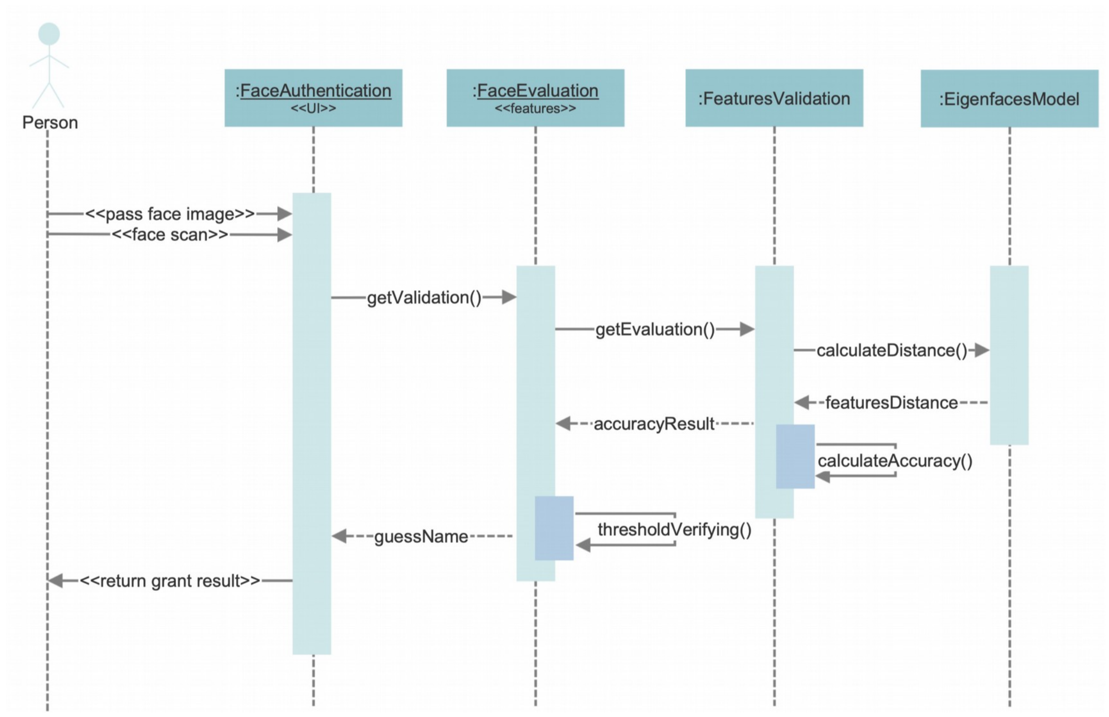
For example the face that has been registered to the system could be detected and informed use the face owner by looking at the percentage of detection accuracy. If the detection accuracy is higher than the threshold then the name shows beside the accuracy told us the face owner. As in the figures above, this prototype system has recorded Mike's faces but has never recorded Joker's and Obama's faces yet. Hence, when we pass the input face image into the system (prototype system), or scan a person face into the system (real purpose system, not prototype system), then Mike face would be perfectly detected with high accuracy higher than those unregistered faces, for instance, Joker and Obama faces. Look at the figure on the left hand side found that Joker's face was predicted as a face of s15 which is a person's faces used to create the detection model, found in the training dataset, named s15's face with the detection accuracy is lower than 65%. That means Joker's faces have never been registered to the system yet. Besides look at the right hand side figure also found that, Obama faces also have never been registered to the system yet. The detection result shows that Obama's face was predicted as Mike's face with accuracy value is 48% which is lower than the threshold (< 65%), that means Obama faces have also never been recorded to the dataset.

## System Design

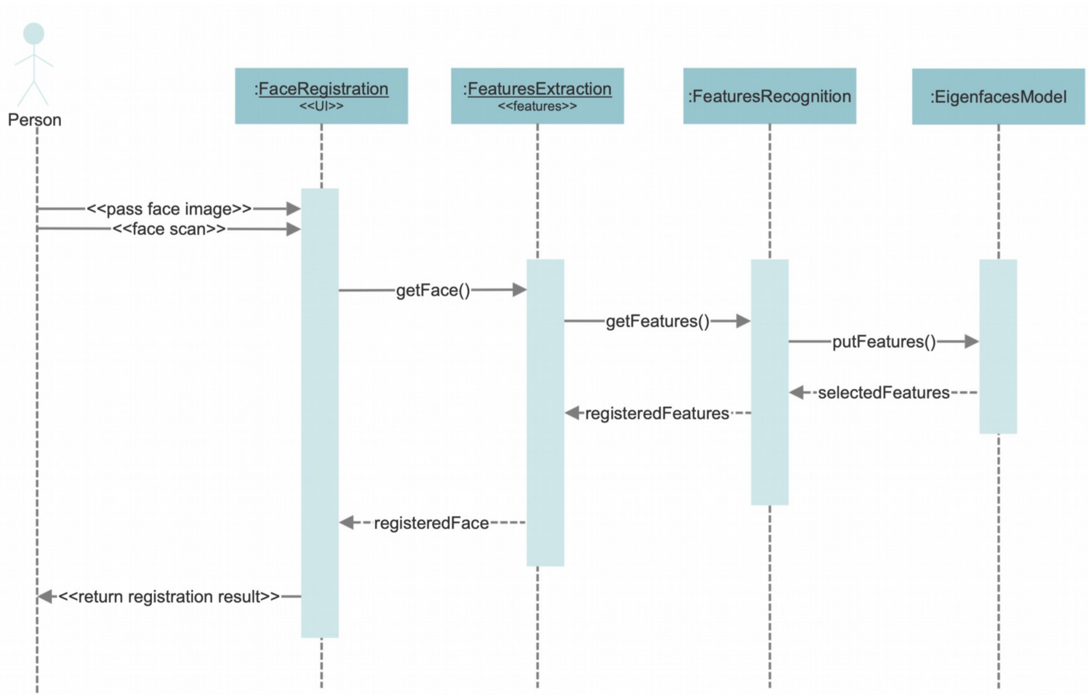
We proposed a simple and generic system use case as in the figure below. The use case diagram is used to describe the software designed we did to fulfill this research study. We also created a UML to describe the face detection process next to the use case and sequence diagram.



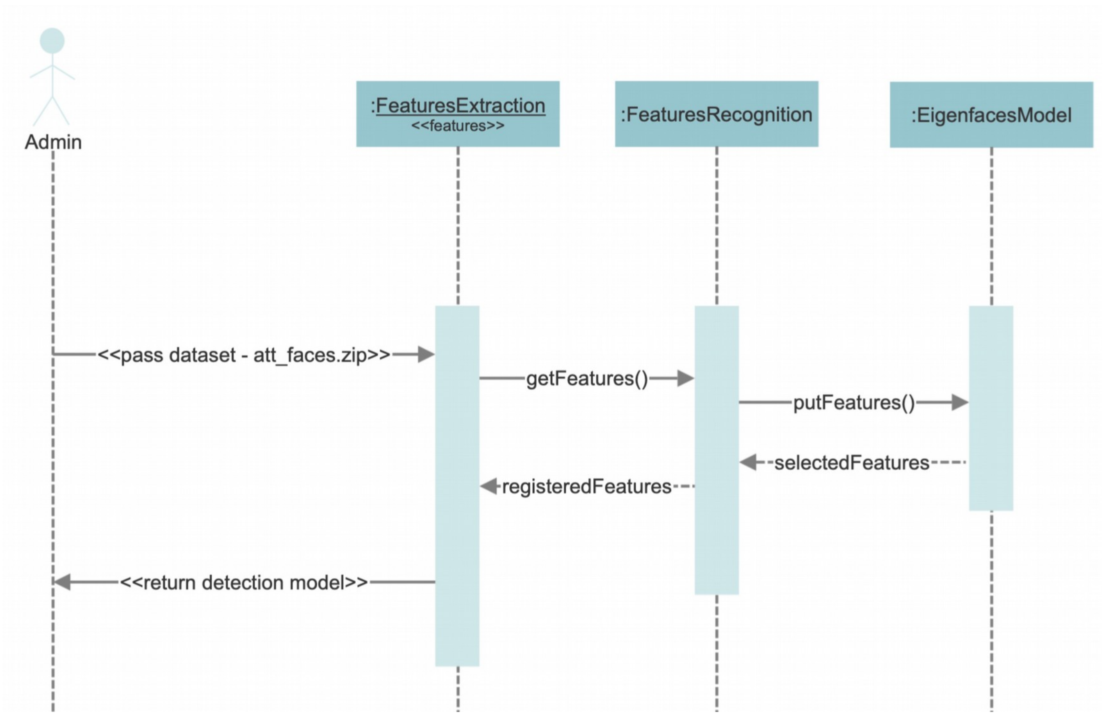
Use Case Diagram



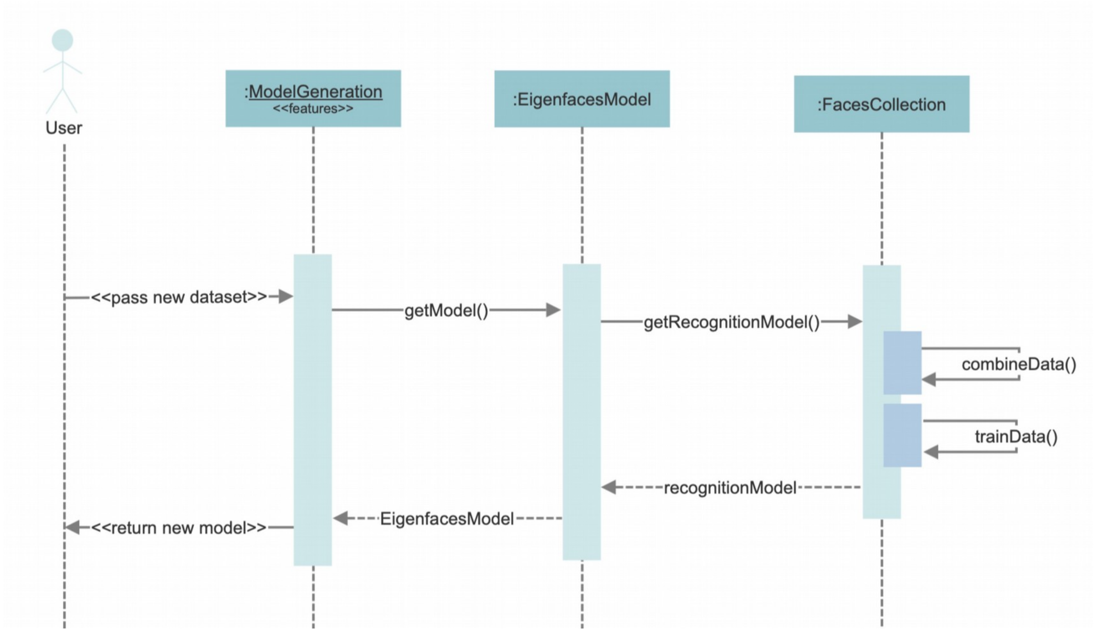
Sequence Diagram: Face Detection



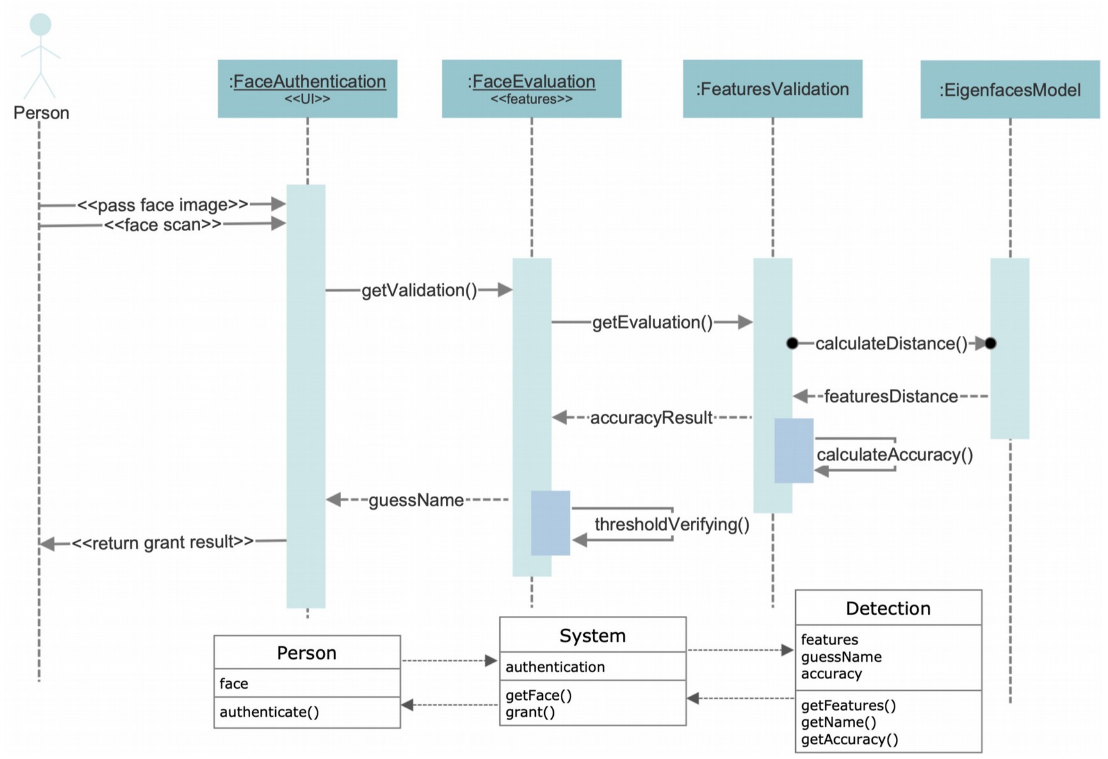
Sequence Diagram: Face Registration



Sequence Diagram: Model Building



Sequence Diagram: Model Generation

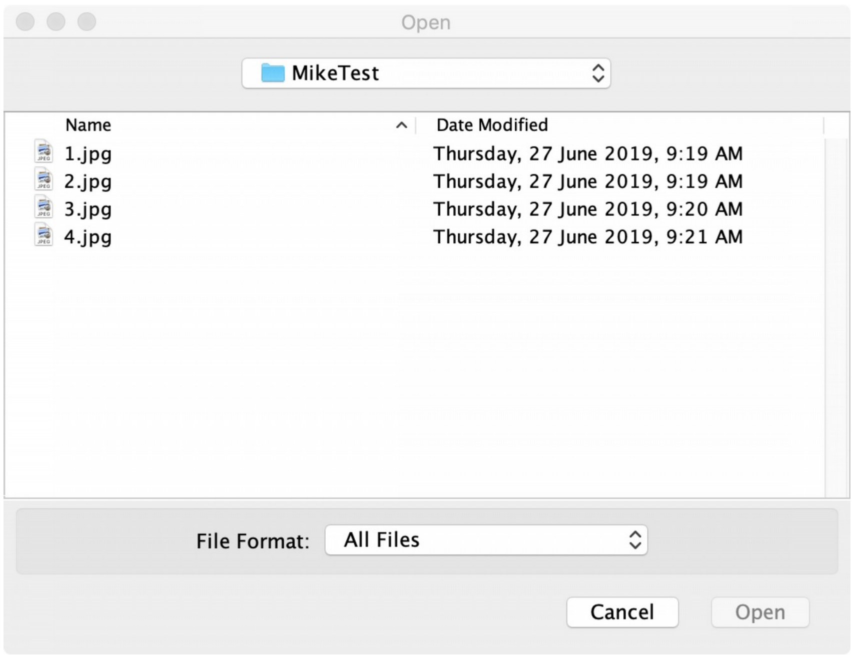


Face Detection UML and Sequence Diagram

Our prototype version was developed for face detection as described in the Face Detection Sequence Diagram. The prototype system can be run by ../ok/**facerecog.jar**. To prepare to run our prototype system, user needs to download:

- facerecog.jar
- facerecog_lib

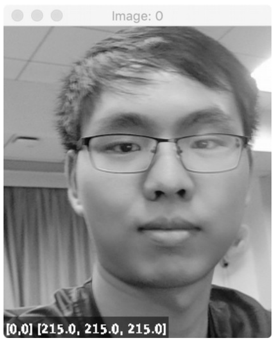
The execution file is facerecog.jar. The first UI will show the user a file selection window to pass the test face image. Suppose that our new registered user is Mike. Therefore, the new model generated by model generation process as described in Model Generation Sequence Diagram has been prepared for Mike's face detection. After run facerecog.jar we will get the window below. After that for testing, we will select the faces of Mike to do authentication test using his face image with different resolution from the training dataset.

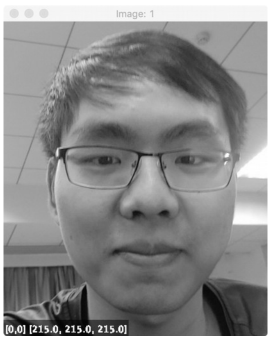


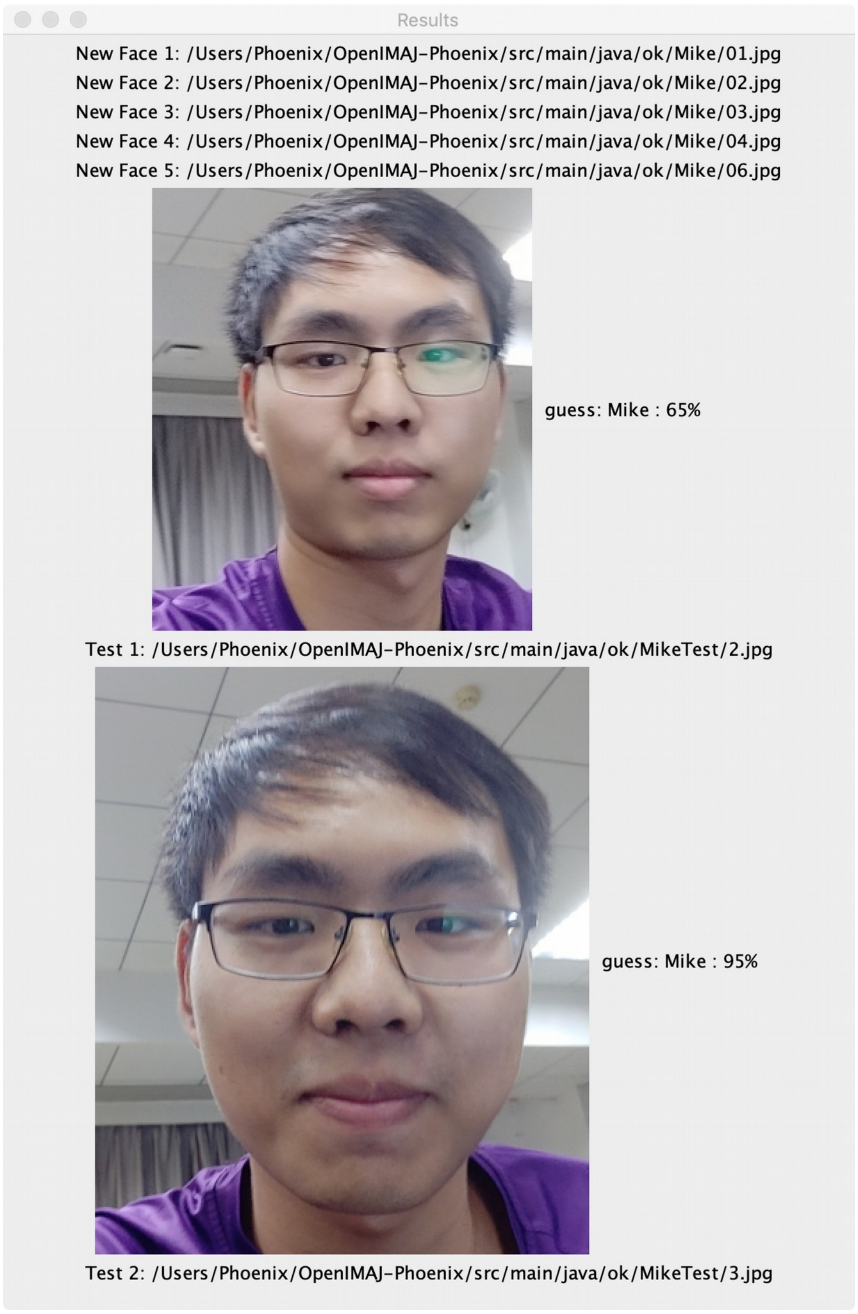
| Name | Date Modified |
|---|---|
| 1.jpg | Thursday, 27 June 2019, 9:19 AM |
| 2.jpg | Thursday, 27 June 2019, 9:19 AM |
| 3.jpg | Thursday, 27 June 2019, 9:20 AM |
| 4.jpg | Thursday, 27 June 2019, 9:21 AM |

File Format: All Files

For example, we choose some images of Mike in ../ok/MikeTest/ path as 2.jpg and 3.jpg, and the face selection (cropped) results are shown below respectively.



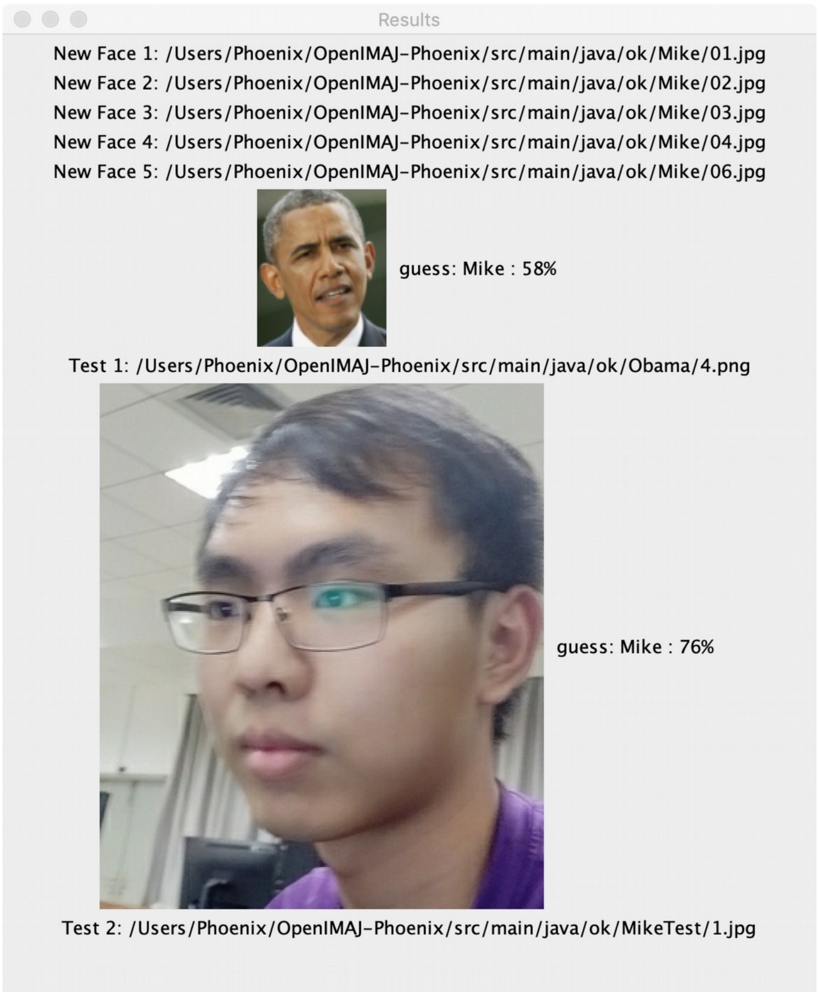




```
New Face 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/01.jpg
New Face 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/02.jpg
New Face 3: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/03.jpg
New Face 4: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/04.jpg
New Face 5: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/06.jpg
guess: Mike : 65%
Test 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/MikeTest/2.jpg
guess: Mike : 95%
Test 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/MikeTest/3.jpg
```

The detection results showed that both images were predicted as Mike's faces with the accuracy values: 65% and 95% respectively.

Suppose we selected other person's faces such as Obama's faces (the face 3.png and 6.png are not proper for use as some limitation, please use another Obama's faces) or Joker's faces. The results showed that the accuracy values were lower than 65% and the prediction could be wrong persons.



```
New Face 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/01.jpg
New Face 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/02.jpg
New Face 3: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/03.jpg
New Face 4: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/04.jpg
New Face 5: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Mike/06.jpg
guess: Mike : 58%
Test 1: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/Obama/4.png
guess: Mike : 76%
Test 2: /Users/Phoenix/OpenIMAJ-Phoenix/src/main/java/ok/MikeTest/1.jpg
```

The detection results showed that Obama’s face was predicted as Mike’s face with the accuracy values: 59%, and Mike’s face was correctly predicted with 76% accuracy result.

## 8. Conclusion

In this work we presented two approaches for face recognition. Firstly, we have preprocessed the image using the OpenIMAJ and MTCNN so that we can get the full face of the image. And then we use the OpenIMAJ to detect the image first, which get an accuracy 0.782 with OpenIMAJ's finding head algorithm and 0.975 with the finding head algorithm which has been optimize. After that, as a comparison, We used a framework called FaceNet to deal with the image, the accuracy of this method on LFW dataset has reached 99.6%, and it is the best record of detection on LFW dataset.