

IMPROVING SECURITY VALIDATION IDENTITY BY USING FACE RECOGNITION AT  
ATMs

---

A Project  
Presented  
to the Faculty of  
California State University Dominguez Hills

---

In Partial Fulfillment  
of the Requirements for the Degree Master of Science  
in  
Computer Science

---

by  
Khalid Alghamdi

Fall 2019

This project is dedicated to my parent,  
For their endless love, support and encouragement.

## ACKNOWLEDGEMENTS

The author of this paper would like to thank Dr. Bin Tang for his invaluable guidance without which the creation of this paper would not be possible. The author would also like to thank my committee members, Dr. Beheshti and Dr. Celly.

## TABLE OF CONTENTS

	PAGE
DEDICATION.....	ii
ACKNOWLEDGEMENTS .....	iii
TABLE OF CONTENTS.....	iv
LIST OF FIGURES .....	vii
ABSTRACT.....	viii
CHAPTER	
I.    INTRODUCTION.....	6
II.    NEED FOR FACE DETECTION AT ATMs.....	8
Main Process.....	9
Problematics.....	9
III.    FACE RECOGNITION ALGHRITHMS COMPARION.....	10
Face Recognition based on geometric characteristics.....	10
Face Recognition using template matching.....	10
Face Recognition using convolutional neural network CNN.....	10
IV.    FUNCTIONAL ANALYSIS APPLICATION.....	11
V.    FACE DETECT RECOGNITION ALGORITHM.....	14
Haar Cascade.....	14
Local Binary Pattern (LBP) .....	15
Image Processing with JavaCV.....	17
VI.    BUILD UP THE DEVEOPMENT PLATFORM.....	21
VII.    TEST THE SYSTEM .....	25
VIII.    CONCLUSION.....	26
REFERENCES.....	27
APPENDICES.....	30
Source Code of Database.....	31
Source Code of FaceDetController.....	34
Source Code of FaceIdentifier.....	43
Source Code of MotionDetector.....	48

## LIST OF FIGURES

	PAGE
1. Haar features Detection .....	8
2. Face recognition and Detection .....	10
3. Cascade Classifier.....	11
4. Applying the LBP operation.....	12
5. LBP histogram.....	14
6. Haar Cascade Insertion.....	14
7. Face detection and tracking .....	15
8. Face recognition .....	17
9. Motion detection.....	18
10. Application structure 1 .....	18
11. Application structure 2 .....	19
12. Person table.....	19
13. Credential Table .....	19
14. Entity–relationship.....	19
15. Application classes relationship.....	20
16. Use Case Diagram .....	21
17. Sequence Diagram for facial detection and recognition.....	23
18. Application .....	27
19. New Face.....	24
20. After recognition.....	24
21. Database.....	25

## ABSTRACT

The purpose of this application is to develop an ATM system application which is able to collect real-time face image and recognize the face by comparing with the datasets inside the system and getting the personal details if there is a match, also, the system will be able to detect eyes, face recognition as a type of biometric method has the features of non-contact, safety, and convenience. It is widely used in human-computer interaction, transaction authentication, security, and other fields. In recent years, with the development of mobile internet and an embedded computer, it becomes possible to run face recognition on an embedded system. This type of application has enormous potential in remote payment and personal information security. This application is running on any operating system such as windows, linux, or mac, the procedure of face recognition includes persistent storage for trained Faces Image Using Database, face, eyes, normalization, and Recognition. This paper aimed to develop a real-time face detection module that analyzes the picture taken by the camera recognizes if a person's face is pictured and determines whether it is covered or not.

## **Chapter 1**

### **INTRODUCTION**

Face recognition is one of the most critical areas of study in image analysis. The research of face recognition and Detection can be traced back to years 1970s. With decades of studies and researches, face recognition has improved a lot. There are famous international conferences such as Conference on Computer Vision and Pattern Recognition, International Conference on Image Processing and Automatic Face and Gesture Recognition. Face detection has become a hot research area because it has excellent potential and significant social needs, especially in domains like banks and ATMs authentication and security. Automatic Teller Machines (ATMs) are widely used in our daily lives due to their convenience, wide-spread availability, and time-independent operation. In this paper, we are going to review some mechanisms used in dealing with the security threat posed to ATM users and found that there are potential threats associated with using a card-based security system, so there is a need to add up another secondary security after the primary stage has been passed. That second stage is a facial recognition security system, as explained in an algorithm developed in this paper. The recommendations for the future biometric system have been suggested, like the smell from mouth breathing be considered as the future secondary security system even though it has got its challenges.

## Chapter 2

### NEED FOR FACE DETECTION AT ATMs

ATM as a cash dispenser that is designed to enable customers to enjoy banking service without coming into contact with Bank Tellers (Cashiers). The ATM, therefore, performs the traditional functions of bank cashiers and other counter staff. It is electronically operated, and as such response to a request by a customer is done instantly. On most modern ATMs, the customer is identified by inserting a plastic ATM card with a magnetic stripe or a plastic smartcard with a chip that contains a unique card number and some security information, such as an expiration date. Security is provided by the customer entering a personal identification number (PIN). Let us take an example; if we need to get money out of an ATM machine, we might have to provide the PIN (Personal Identification Number) and password. However, we might forget the password, or somebody can find out the password. If we are using human face recognition as additional protection, we might not need to worry too much if we lose our credit card.

Face recognition also has the following features:

- Compared to other biometrics, such as fingerprint and DNA, face recognition does not need the tester to cooperate.
- Image sample collection can be done in a natural and easy way, which does not require complicated equipment.
- A lot of neuroscience research has shown that face recognition is controlled by a specific part of the human brain and is very important to us.

Research on face recognition is essential for us to understand human brains and also crucial to artificial intelligence and machine learning.



## Main Process

Firstly, once running the application, it uses the camera of the system to capture real-time person is face then detect the target person is face based on detection algorithms. After that done, the program will then do an image preprocessing algorithm to the image to catch the face image using a face detect algorithm and draw a rectangle on it and save the image sample to the data folder. Moreover, here, where it comes the role of JavaCV libraries (wrapper of Open CV) class libraries with Java Core and JavaFX environment and use the principal component analysis algorithm for face image feature extraction. After extraction and save the personal data to MySQL database, the next step is to use comparison algorithms for matching the person with saved data and check for an identical match.

## Problematics

The application focuses on solving the following issues:

- Preprocessing and face Detection

This application will preprocess the real-time faces and use existing algorithms and functions that will help us to locate the face image. There are also functions in JavaCV (Open CV Wrapper), which helps us to locate the face image.

- Face Recognition

Different algorithms are developed in doing face recognition. However, some of them are not useful in running on a real-time system. In this application, I choose principal component analysis in doing face recognition. In the face recognition part, several types of decisions can be made. For face recognition, we Identify if the target person is one of the people in the database.

## Chapter 3

### FACE RECOGNITION ALGORITHMS COMPARISON

There are different face recognition methods:

- Face Recognition based on geometric characteristics:

Firstly, by use of geometric feature vector to represent the human face. Using hierarchical clustering-based pattern recognition to do human face recognition. The facial feature vector is a geometric shape and geometry of organs based on feature vectors, whose components usually include a custom face the Euclidean distance between two points, curvature or angle.

- Face Recognition using template matching:

The template matching method is mostly used normalized cross-correlation, direct calculation of the degree of matching between two images. Since this method requires the target of two images have the same scale, orientation, and illumination conditions, the pre-normalization becomes very important for the work. There are some typical matching functions that we can use to make a comparison with the images in the database.

- Face Recognition using hidden Markov model:

Hidden Markov Model is a signaling system used to describe the characteristics of a standard statistical model which is widely used in voice recognition. Facial features are in accordance with the distribution of a natural sequence, i.e., from the top to the small, left to right, even if the human face in the plane and the vertical direction is rotated, this order is not going to be changed; however, it becomes difficult to implement.

- Face Recognition using convolutional neural network CNN:

With the development of deep learning, face recognition technology based on CNN (Convolutional Neural Network) has become the primary method adopted in the field of face recognition. It has an outstanding ability to deal with damaged facial images and different illumination conditions.

- Principle component analysis method:

One of the simplest and most effective PCA approaches used in face recognition systems is the so-called eigenface approach. This approach transforms faces into a small set of essential characteristics, eigenfaces, which are the main components of the initial set of learning images (training set). Recognition is done by projecting a new image in the eigenface subspace, after which the person is classified by comparing its position in eigenface space with the position of known individuals. Local Binary Pattern (LBP) is the algorithm I am using in this application. Compared to the algorithm above, it is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

- Haar Cascade classifier:

It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images (where positive images are those where the object to be detected is present, contrary are those where it is not). It is then used to detect objects in other images. OpenCV offers pre-trained Haar cascade algorithms, organized into categories (faces, eyes), depending on the images they have been trained on.

## Chapter 4

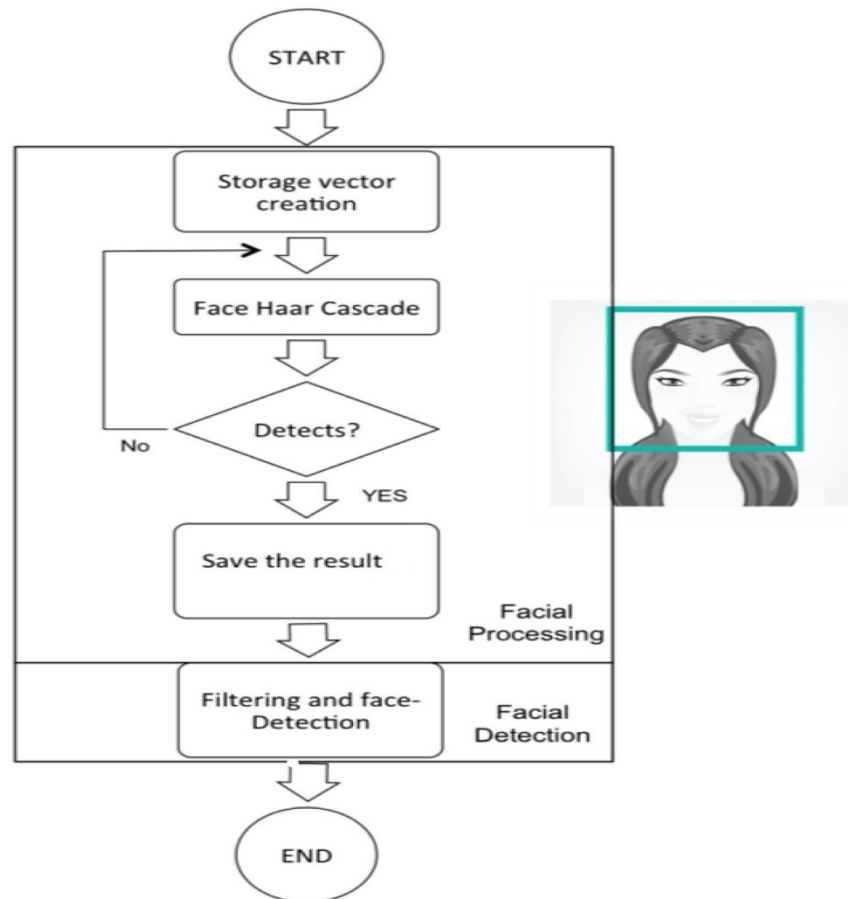
### FUNCTIONAL ANALYSIS APPLICATION

To do face recognition, firstly, we use to launch the application and use the camera of the system to get a real-time view of the objects around. We do a preprocessing algorithm to detect the face image of the target person. After that, we need to train a new face, and we have to capture at least ten pictures of a single person from a different angle. Furthermore, store the person related info in the database. Then it comes the role of JAVACV as a wrapper of OpenCV and use of Local Binary Pattern (LBP) as well as Haar Cascade classifier for face recognition and matches result with the database. Other features are included in the application; the application is capable of OCR (Optical Character Reader) with the use of tesseract. Generally, OCR tesseract works as follow:

1. Preprocess image data convert to grayscale, smooth, de-skew, filter.
2. Detect lines, words, and characters.
3. Produce a ranked list of candidate characters based on a trained data set. (here the `setDataPath()` method is used for setting a path of trainer data)
4. Post-process recognized characters choose the best characters based on confidence from the previous step and language data. Language data includes a dictionary, grammar rules.

Other features added, like Eye Detection, Rectangle Shape Detection, Gesture Control.

Local Binary Pattern (LBP) and Haar Cascade classifier algorithms are the main algorithms used for detecting and recognitions.



*Figure 1 Haar Cascade Detection*

If we are looking for one person among a significant number of people, how many real-time images of that person we should have in our database to balance the relationship between computation speed, hardware space, and correct rate? Moreover, we might face some other problems such as a difference in the environment, the difference of the person's facial angle, and distance of the person's face to the camera. In this application, the algorithm used makes detection and recognition at a very high rate, but the person needs to have to be trained with many pictures for different angles and views.

The Face recognition and Detection of the system work as follow:

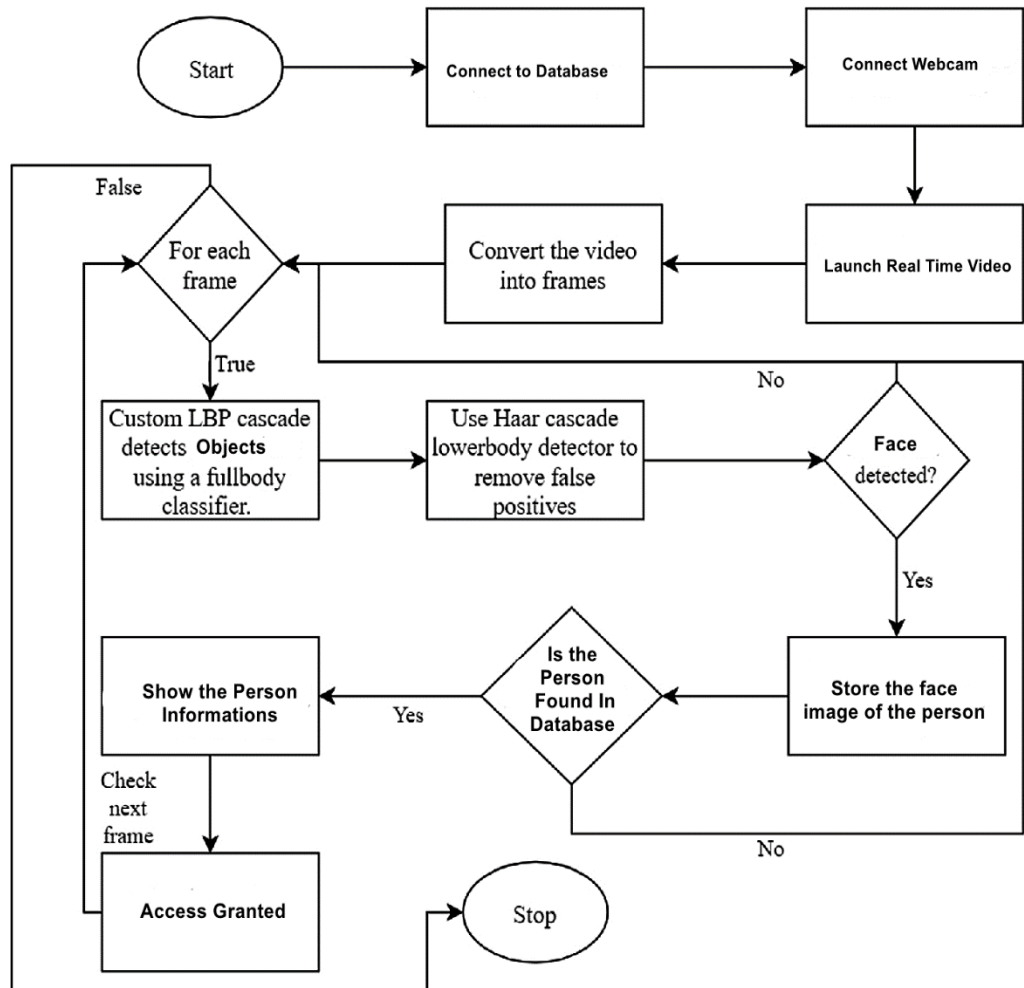


Figure 2 Face Recognition and Detection

## Chapter 5

### FACE DETECT RECOGNITION ALGORITHM

Two main algorithms have been used in the application:

- Haar Cascade

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video or camera. Initially, the algorithm needs a lot of positive images of faces from the camera and negative images without faces to train the classifier. Then we need to extract features from it. So basically, once running the camera, the first step is to collect the Haar Features. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region, and calculates the difference between these sums.

In  $a \times m$  sub-window, we only need to determine the top-left corner  $A(x1,y1)$  and lower right rectangle  $B(x2,y2)$  to find out a feature rectangle

1.  $x2-x1$  can be divided into  $s$  segments.
2.  $y2-y1$  can be divided into  $t$  segments. The minimum size of this rectangle is  $s \times t$  or  $t \times s$ , the maximum size is  $[m/s] \cdot s \times [m/t] \cdot t$  or  $[m/t] \cdot t \times [m/s] \cdot s$

During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing), a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into cascade classifiers to form a strong classifier.

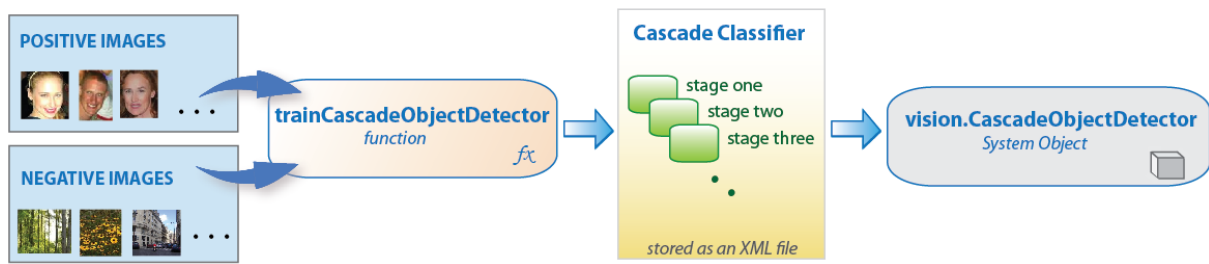


Figure 3 Cascade Classifier

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called *decision stumps*. Each stage is trained using a technique called *boosting*. *Boosting* provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Cascade classifier training requires a set of positive samples and a set of negative images. We must provide a set of positive images with regions of interest specified to be used as positive samples.

We can use the Image Labeler to label objects of interest with bounding boxes. The Image Labeler outputs a table to use for positive samples. We also must provide a set of negative images from which the function generates negative samples automatically. To achieve acceptable detector accuracy, set the number of stages, feature type, and other functional parameters.

- Local Binary Pattern (LBP)

Local Binary Patterns (LBP) is a non-parametric descriptor whose aim is to summarize the local structures of images efficiently. In recent years, it has aroused increasing interest in many areas of image processing and computer vision. It has shown its effectiveness in several applications, in particular for facial image analysis, including tasks as diverse as face detection, faces Recognition, facial expression analysis, demographic classification.



Basically, to detect and recognize faces, LBP uses four parameters:

- Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the persons we want to recognize. We also need to set an for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID with the training set already constructed.

Then, the first computational step of the LBP is to create an intermediate image that describes the original image in a better way by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window based on the parameters: radius and neighbors.

The procedure of that is like the following:

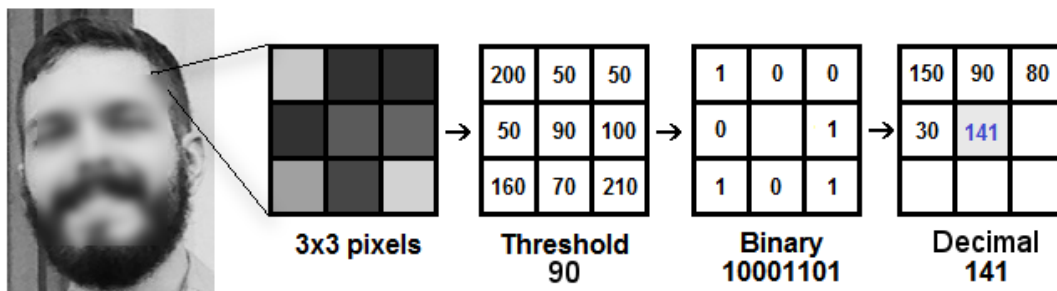


Figure 4 Applying the LBP operation

So basically, the steps for getting the end image from the original image is as follow:

- First, we get a real-time face imager of the person in front of the camera.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel.
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the eight neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g., 10001101).
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

The next step is to extract the Histograms: by using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids:

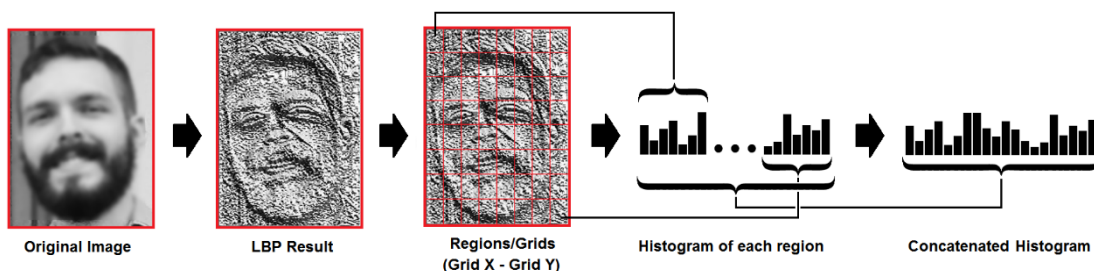


Figure 5 LBP histogram

Extraction is as follow:

- each histogram will contain only positions representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and more significant histogram. Supposing we have 8x8 grids, we will have  $8 \times 8 \times 256 = 16.384$  positions in the final histogram. The final histogram represents the characteristics of the image's original image.

The final step is to perform Recognition: the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram that represents the image. So, to find the image that matches the input image, we need to compare two histograms and return the image with the closest histogram.

### Image Processing with JavaCV

After we know how algorithms work, we need to use prebuild library tools that can aid to do Detection and Recognition of faces at ATMs. JavaCV is an excellent choice for that, JavaCV is a collection of wrappers for many famous libraries used for image processing and computer vision-related applications. JavaCV includes wrappers for OpenCV.

The image bellow read and set our haar cascade we will use for Detection in the application.

```

112 public String lname; //last name
113 public String sec; //section
114 public String name;
115
116 public void init() {
117     faceRecognizer.init();
118
119     setClassifier("haar/haarcascade_frontalface_alt.xml");
120     setClassifierEye("haar/haarcascade_eye.xml");
121     setClassifierEyeGlass("haar/haarcascade_eye_tree_eyeglasses.xml");
122     setClassifierSideFace("haar/haarcascade_profileface.xml");
123     setClassifierFullBody("haar/haarcascade_fullbody.xml");
124     setClassifierUpperBody("haar/haarcascade_upperbody.xml");
125     setClassifierSmile("haar/haarcascade_smile.xml");
126
127 }
128
129 public void start() {
130     try {

```

*Figure 3 Haar Cascade Insertion*

The code below is for face detection and tracking.

```

*/
private void detectAndDisplay(Mat frame)
{
    MatOfRect faces = new MatOfRect();
    Mat grayFrame = new Mat();

    // convert the frame in gray scale
    Imgproc.cvtColor(frame, grayFrame, Imgproc.COLOR_BGR2GRAY);
    // equalize the frame histogram to improve the result
    Imgproc.equalizeHist(grayFrame, grayFrame);

    // compute minimum face size (20% of the frame height, in our case)
    if (this.absoluteFaceSize == 0)
    {
        int height = grayFrame.rows();
        if (Math.round(height * 0.2f) > 0)
        {
            this.absoluteFaceSize = Math.round(height * 0.2f);
        }
    }

    // detect faces
    this.faceCascade.detectMultiScale(grayFrame, faces, 1.1, 2, 0 | Objdetect.CASCADE_SCALE_IMAGE,
        new Size(this.absoluteFaceSize, this.absoluteFaceSize), new Size());

    // each rectangle in faces is a face: draw them!
    Rect[] facesArray = faces.toArray();
    for (int i = 0; i < facesArray.length; i++)
    {
        Imgproc.rectangle(frame, facesArray[i].tl(), facesArray[i].br(), new Scalar(7, 255, 90), 4);
        System.out.println(facesArray[i].tl());
        System.out.println(facesArray[i].br());
    }
}
}

```

*Figure 4 face detection and tracking*

A screenshot of an IDE window titled 'story'. The window shows a Java method named 'recognize' that takes an 'IplImage faceData' parameter. The code performs several steps: it converts the image to a 'Mat' object, then to grayscale using 'CV\_BGR2GRAY'. It initializes an 'IntPointer' for the label and a 'DoublePointer' for confidence. The 'faceRecognizer' object's 'predict' method is called with the 'faces' Mat, the 'label' pointer, and the 'confidence' pointer. The predicted label is retrieved from the 'IntPointer'. A comment indicates that a confidence value less than 60 means the face is known, and greater than 60 means it is unknown. An 'if' statement checks if the confidence is greater than 60; if so, it prints '-1' and returns -1. Finally, it returns the predicted label.

```
public int recognize(IplImage faceData) {  
  
    Mat faces = cvarrToMat(faceData);  
  
    cvtColor(faces, faces, CV_BGR2GRAY);  
  
    IntPointer label = new IntPointer(1);  
    DoublePointer confidence = new DoublePointer(0);  
  
    this.faceRecognizer.predict(faces, label, confidence);  
  
    int predictedLabel = label.get(0);  
  
    //System.out.println(confidence.get(0));  
  
    //Confidence value less than 60 means face is known  
    //Confidence value greater than 60 means face is unknown  
    if(confidence.get(0) > 60)  
    {  
        //System.out.println("-1");  
        return -1;  
    }  
  
    return predictedLabel;  
}  
  
}
```

*Figure 5 face recognition*

## Chapter 6

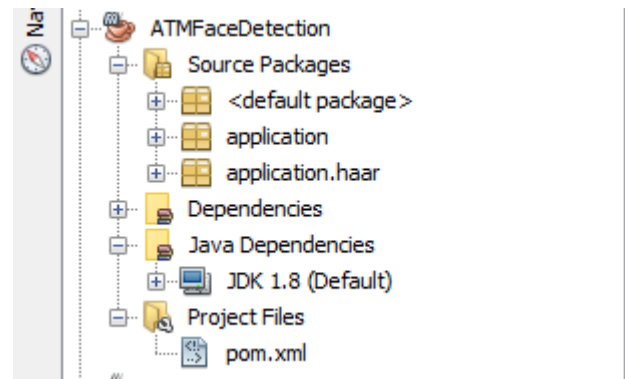
### BUILD UP THE DEVELOPMENT PLATFORM

- Core Java platform is a collection of programs that help to develop and run programs written in the Java programming language. Java platform includes an execution engine, a compiler, and a set of libraries. JAVA is a platform-independent language. It is not specific to any processor or operating system.
- JavaFX is a software platform for creating and delivering desktop applications, as well as rich Internet applications that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE.
- JavaCV is a collection of wrappers for many famous libraries used for image processing and computer vision-related applications. JavaCV includes wrappers for OpenCV, FFmpeg, OpenKinect, and much more. JavaCV has everything required for computer vision and image processing in one place.
- MySQL is a freely available open-source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).
- Maven is a powerful project management tool that is based on POM (project object model). It is used for projects to build, dependency, and documentation. It simplifies the build process like ANT. Nevertheless, it is too much advanced than ANT.
- Tesseract OCR Framework is an optical character recognition engine with open-source code; this is the most popular and qualitative OCR-library.

## Program Structure:

The application contains two main parts:

- Resources that contains all resources for the applications like images file.
- Application folder where we use to save source code.
- Application haar folder that is used to save haar cascade files.
- Pom.xml is where we place all dependencies for the application.



*Figure 10 Application structure1*

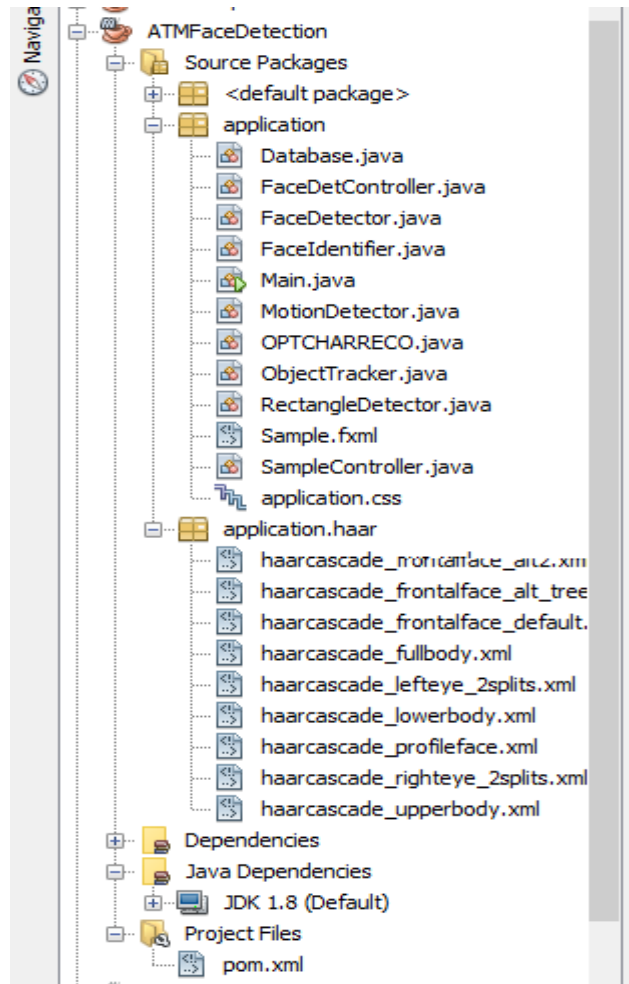


Figure 11 Application structure2

The application stores the person image inside the application folder image, also it stores its information in MySQL database:

The database contains a table person that has the following structure:

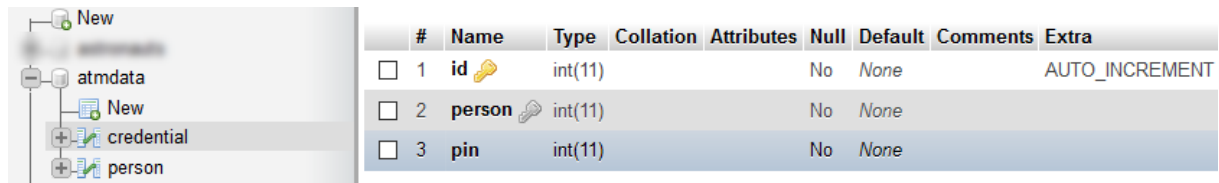
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/> 1	<b>id</b>	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/> 2	<b>code</b>	int(10)			No	None		
<input type="checkbox"/> 3	<b>first_name</b>	varchar(30)	latin1_swedish_ci		No	None		
<input type="checkbox"/> 4	<b>last_name</b>	varchar(20)	latin1_swedish_ci		No	None		
<input type="checkbox"/> 5	<b>reg</b>	int(10)			No	None		
<input type="checkbox"/> 6	<b>age</b>	int(10)			No	None		
<input type="checkbox"/> 7	<b>section</b>	varchar(20)	latin1_swedish_ci		No	None		

Figure 12 Person Table



Person Table contains the person id, code, first and last name, registration, code, the section along with its faces that are stored in the application image folder.

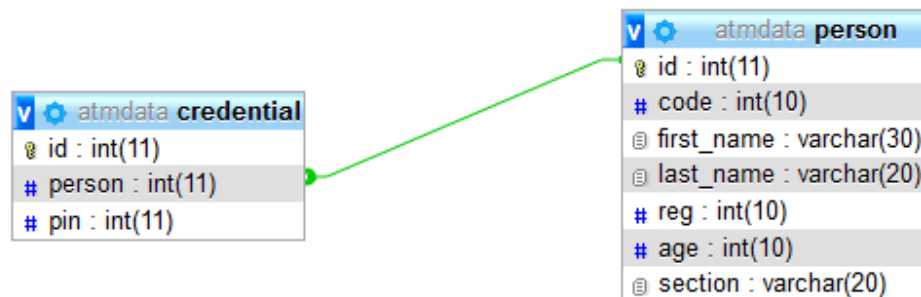
And also it has a credential table that store pin for ATM card for a specific person.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	person	int(11)			No	None		
3	pin	int(11)			No	None		

*Figure 13 Credential Table*

Entity–relationship model:



*Figure 14 Entity–relationship*

## Application classes relationship

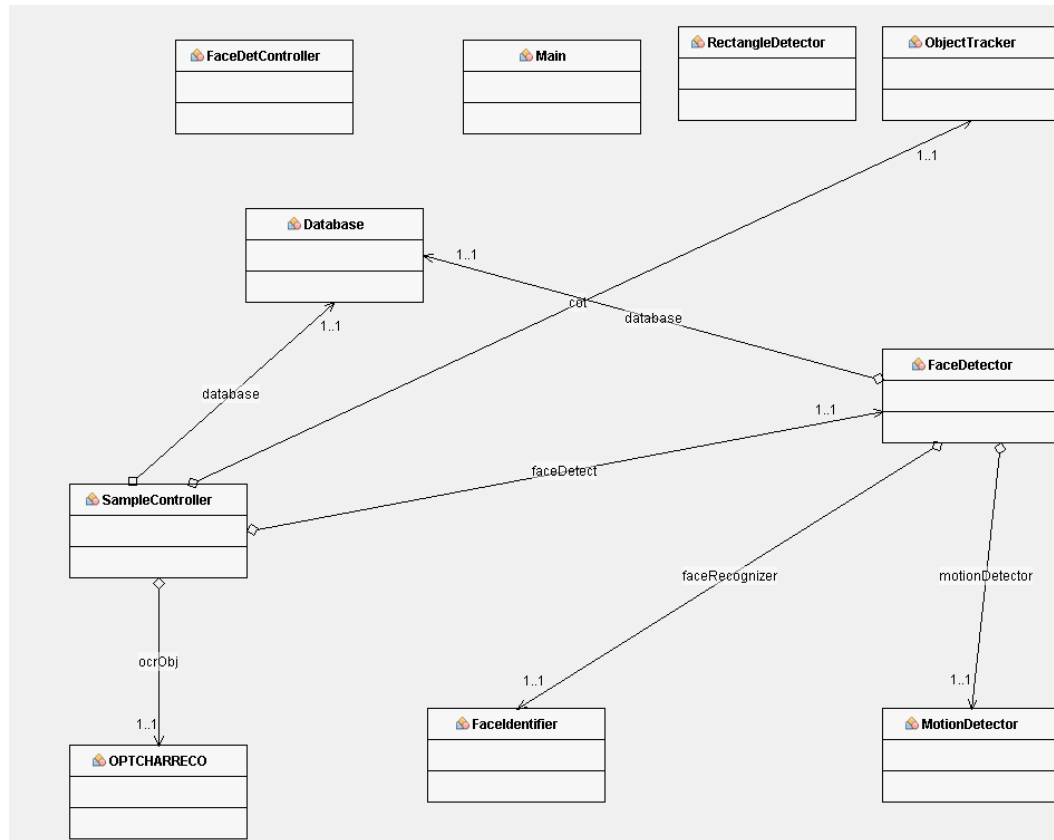


Figure 15 Application classes relationship

In our application there are two tables in the entity:

- Table person with fields: id, code, first\_name, last\_name, reg, age and section.

And the primary key is id.

- Table credentials with fields: id, person, pin.

With id is the primary key, and person if the foreign key.

## Use cases:

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.

Use case can be divided into three parts, a use case, actors and relationship.

- Actors are the users of a system:

In our application, there are two actors:

Card holder and bank customers are the main primary actors.

Atm system is the secondary actor.

- Use case represent the system's functions:

Card holder and bank customers can interact with the atm system and validate their cards using the face detection and recognition system.

- Illustrate relationships between an actor and a use case:

Bank customer extends all functionalities of card holder.

Face recognition system extends also all functionalities of system detection.

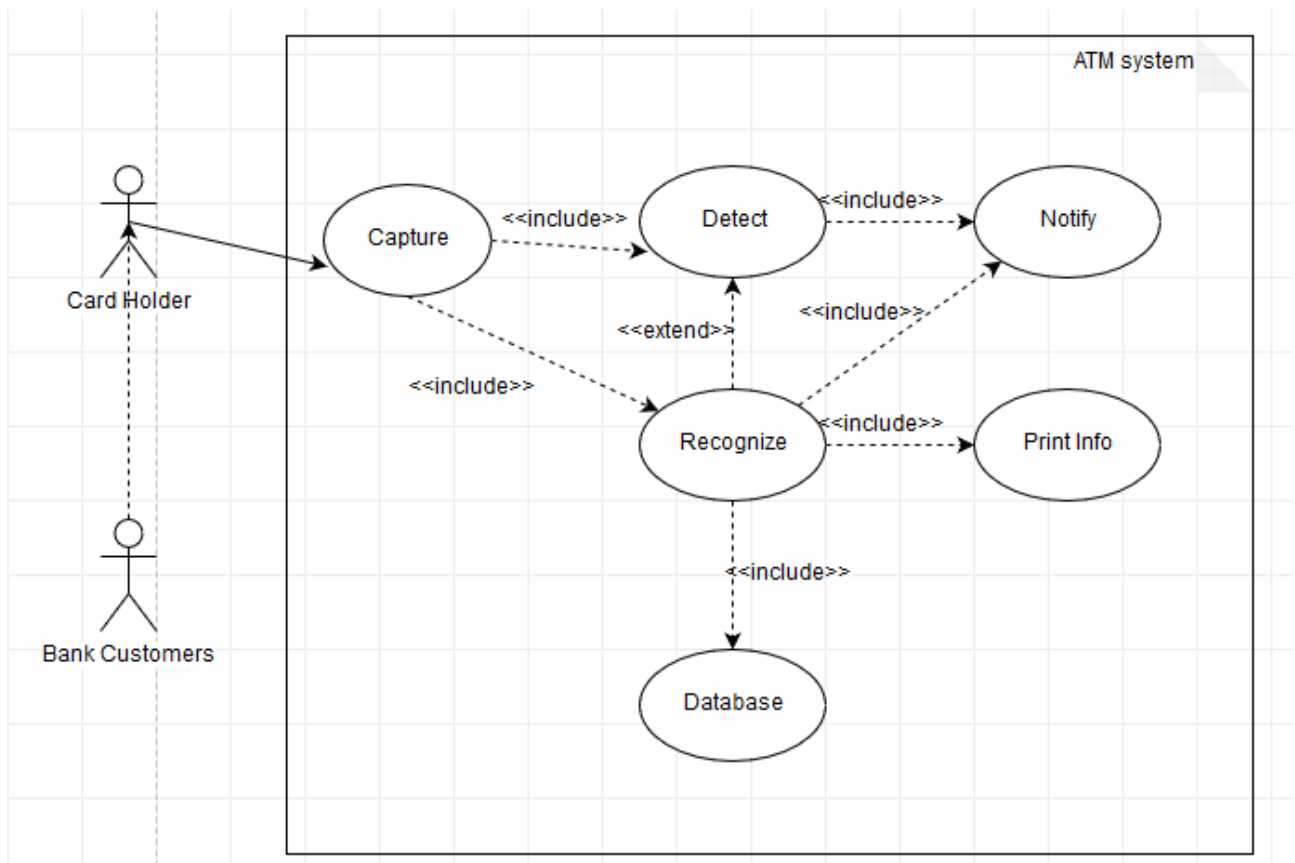


Figure 16 Use Case Diagram

### Sequence Diagram:

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

For our application the sequence diagram is:

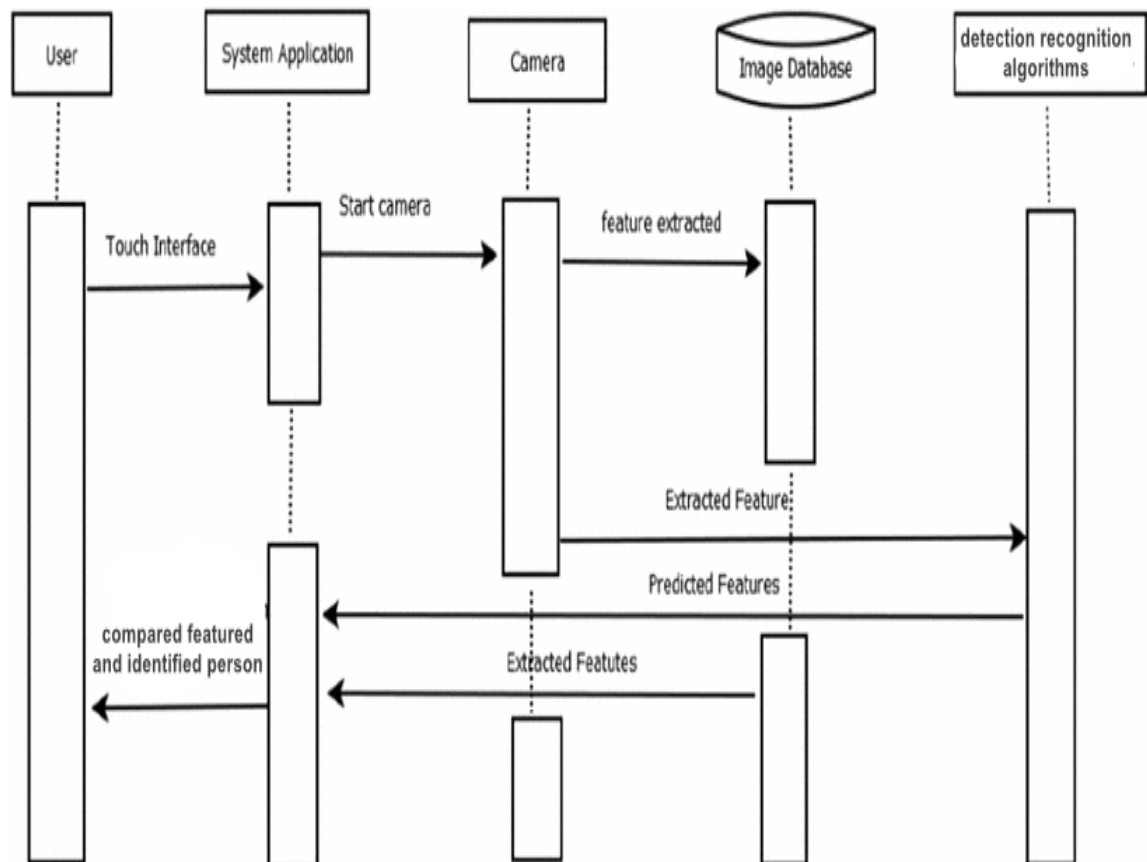
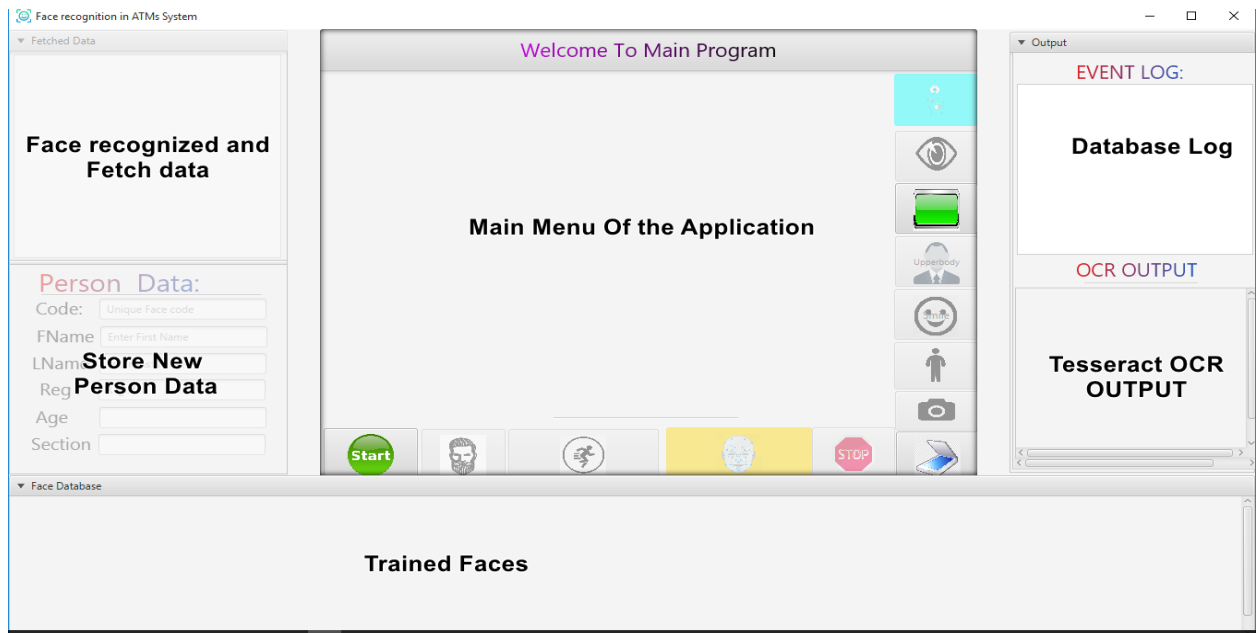


Figure 17 Sequence Diagram for facial detection and recognition

## Chapter 7

### TEST THE SYSTEM

Once we run the application main Interface is:

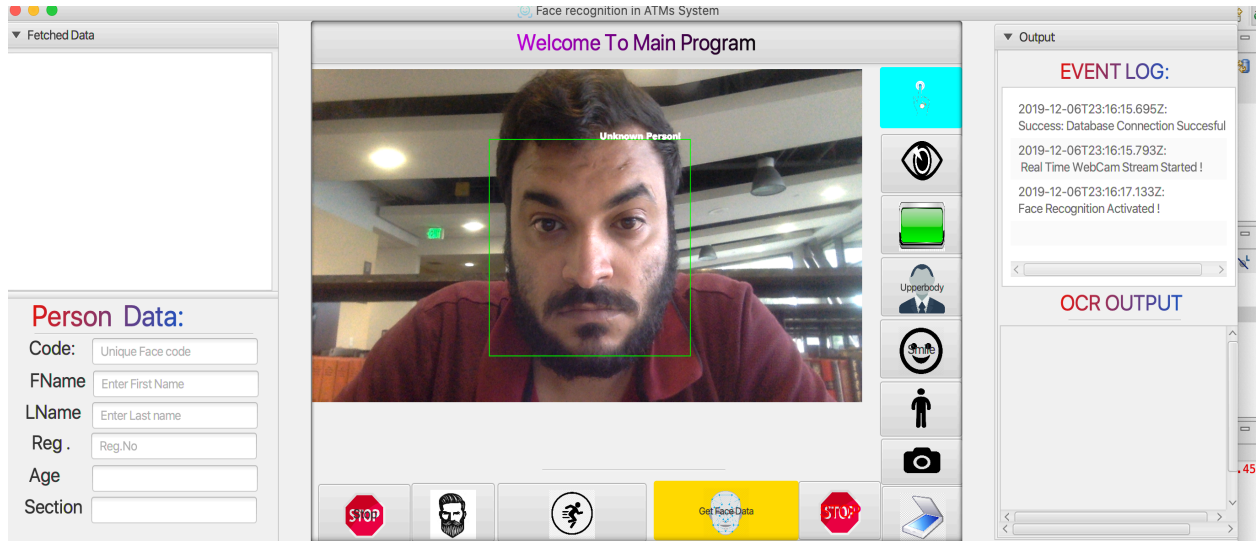


*Figure 18 Application*

· Main Menu is where the camera appears and detect and recognize faces. Application has many controls buttons; each has a role (start, stop, store new person, detect movement, detect eyes, detect rectangles objects, OCR)

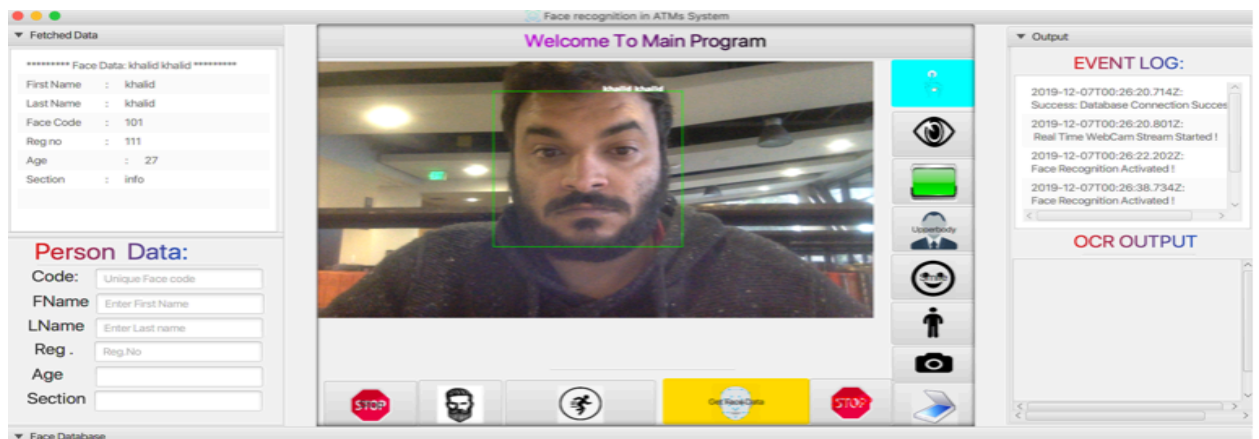
- Database log is where database events logs appear there.
- OCR output log all OCR outputs.
- Faces database is where trained images show up.
- Fetched data is where appears recognized person is information.

## RESULTS



*Figure 19 New Face*

When we run the system, the system will detect the face and display Unknown Person, if the face is not in the database.



*Figure 20 After recognition*

After we insert new face and person data, we can recognize the face and fetch the data from the database.

```
mysql> delete from person where id=9;
Query OK, 1 row affected (0.00 sec)

mysql> select * from person;
Empty set (0.00 sec)

mysql> select * from person;
+----+-----+-----+-----+-----+-----+-----+
| id | code | first_name | last_name | reg | age | section |
+----+-----+-----+-----+-----+-----+-----+
| 10 | 101 | khalid    | khalid    | 111 | 27 | info    |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> █
```

Figure 21 Database

In the database, it shows before and after we insert new person to the database.

## **Chapter 8**

### **CONCLUSION**

What I have been doing is to build up an application that is able to validate identity at ATMs by doing Detection and face detection recognition. I have chosen the combination of Haar Cascade and Local Binary Pattern (LBP) algorithms. During the development, I have been working on face detection algorithms, preprocessing methods, and Face Recognition. I have done testing on face detection speed and the ability to preprocess. There are also testing on the selection of subspace dimensions. It has been proved that the recognition rate is over 90% using a proper strategy. And that is good results and can be applied as security and validation at ATMs. Future work and modifications of this application might be the focus on the speed and recognition rate.



## REFERENCES

- Agarwala, A., Hertzmann, A., Seitz, S., & Salesin, D. (2004). Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 584–591.
- AL-Allaf, O. (2014). Review Of Face Detection Systems Based Artificial Neural Networks Algorithms. *The International Journal of Multimedia & Its Applications (IJMA)*, 1-16.
- Bakhshi, Y., Kaur, S., & Verma, P. (2015). A Study based on Various Face Recognition Algorithms. *International Journal of Computer Applications*, 16-20.
- Baluja, H., & Kanade, T. (1999). Neural Network-Based Face Detection, Computer Vision and Pattern Recognition. *Pittsburgh, Carnegie Mellon University, Ph.D. thesis*, 1-325.
- Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 509–522.
- Bertalmio, M., Sapiro, G., Caselles, V., & Ballester, C. (2000). Image inpainting. *In ACM SIGGRAPH 2000 Conference Proceedings*, 417–424.
- Bertalmio, M., Vese, L., Sapiro, G., & Osher, S. (2003). Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 882–889.
- Bernabe, S., López, S., Plaza, A., & Sarmiento, R. (2013). GPU implementation of automatic target detection and classification algorithm for hyperspectral image analysis. *IEEE Geoscience and Remote Sensing Letters*, 10(2), 221-225.
- Bonnen, K., Klare, B. F., & Jain, A. K. (2013). Component-based representation in automated face recognition. *IEEE Transactions on Information Forensics and Security*, 8(1), 239-253.

- Burrell, J. (2016). How the machine ‘thinks’: Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1), 2053951715622512
- Boden, M. A. (2006). *Mind As Machine: A History of Cognitive Science*. Oxford, England: Oxford University Press.
- Dudley, B. (2009). New info on Microsoft's natal – how it works, multiplayer and pc versions. *The Seattle Times*, n.d.
- Eisemann, E., & Durand, F. (2004). Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 673–678.
- Egger, J., Kapur, T., Fedorov, A., Pieper, S., Miller, J. V., Veeraraghavan, H., & Kikinis, R. (2013). GBM volumetry using the 3D Slicer medical image computing platform. *Scientific reports*, 3, 1364.
- Eisenberg, J. D., & Bellamy-Royds, A. (2014). *SVG Essentials: Producing Scalable Vector Graphics with XML*. "O'Reilly Media, Inc.."
- Fergus, R., Perona, P., & Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '2005)*, 380–387.
- Fischler, M., & Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 67-92.
- Freeman, W., Perona, P., & Schölkopf, B. (2008). Special issue on machine learning for vision. *International Journal of Computer Vision*, 1-3.
- Golstein, A., Harmon, L., & Lest, A. (1971). Identification of human faces. *Proceedings of the IEEE*, 748–760.

- Hartley, R. I., & Zisserman, A. (2004). *Multiple View Geometry*. Cambridge, UK: Cambridge University Press.
- Hoiem, D., Efros, A. A., & Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, 3-15.
- Kenade, T. (1973). Picture Processing System by Computer Complex and Recognition of Human Faces. Ph.D. *thesis, Kyoto University*, n.d.
- Kirby, M., & Sirovich, L. (1990). Application of the Karhunen–L`oeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 103-108.
- Kodagali, J. A., & Balaji, X. (2012). Computer Vision and Image Analysis based Techniques for Automatic Characterization of Fruits – a Review. *International Journal of Computer Applications*, 0975 – 8887.
- Lalonde, J.-F., Hoiem, D., Efros, A. A., Rother, C., Winn, J., & Criminisi, A. (2007). Photo clip art. *ACM Transactions on Graphics*, n.d.
- Li, B. Y., Mian, A. S., Liu, W., & Krishna, A. (2013, January). Using Kinect for face recognition under varying poses, expressions, illumination, and disguise. *In Applications of Computer Vision (WACV), 2013 IEEE Workshop on* (pp. 186-192). IEEE.
- Lin, D., Kapoor, A., Hua, G., & Baker, S. (2010). Join people, event, and location recognition in personal photo collections using cross-domain context. *In Eleventh European Conference on Computer Vision (ECCV 2010)*, n.d.
- Lu, X. (2018). Image Analysis for Face Recognition. *Michigan State University*, 1-37.

## APPENDICES

## Source Code of Database

```
import java.util.ArrayList;
import java.util.List;

class Database {
    public int code;

    public String fname;
    public String lname;
    public int reg;
    public int age;
    public String sec;

    // database settings: database name username and password
    public final String Database_name = "atmdata";
    public final String Database_user = "root";
    public final String Database_pass = "12345678";

    public Connection con;

    public boolean init() throws SQLException {
        // connect to mysql database if success
        try {
            Class.forName("com.mysql.jdbc.Driver");

            try {
                this.con =
                    DriverManager.getConnection("jdbc:mysql://localhost:3306/" + Database_name,
                        Database_user,
                            Database_pass);
            } catch (SQLException e) {

                e.printStackTrace();
                System.out.println("Error: Database Connection Failed
! Please check the connection Setting");

                return false;
            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

            return false;
        }

        return true;
    }

    // insert function to insert new person data to person table once
    // user has feel in the fields on the application
    public void insert() {
        String sql = "INSERT INTO person (code, first_name, last_name,
reg, age , section) VALUES (?, ?, ?, ?,?,?)";
        // prepare the sql statement for the conecction instance con
```

```

        PreparedStatement statement = null;
        try {
            statement = con.prepareStatement(sql);
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        try {
            //read fields from application and insert them to the
statement
            statement.setInt(1, this.code);
            statement.setString(2, this.fname);

            statement.setString(3, this.Lname);
            statement.setInt(4, this.reg);
            statement.setInt(5, this.age);
            statement.setString(6, this.sec);
            // execute the statement if inserted show success message
            int rowsInserted = statement.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("A new face data was inserted
successfully!");
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    // fetch data for the person for specific person is code
    public ArrayList<String> getUser(int inCode) throws SQLException {

        ArrayList<String> user = new ArrayList<String>();

        try {

            Database app = new Database();

            String sql = "select * from person where code=" + inCode +
" limit 1";

            Statement s = con.createStatement();

            ResultSet rs = s.executeQuery(sql);

            while (rs.next()) {

                // if there is a person with givin code id then fetch
it and
                // insert its data to teh arraylist

                user.add(0, Integer.toString(rs.getInt(2)));
                user.add(1, rs.getString(3));
                user.add(2, rs.getString(4));
                user.add(3, Integer.toString(rs.getInt(5)));
                user.add(4, Integer.toString(rs.getInt(6)));
                user.add(5, rs.getString(7));
            }
        }
    }

```

```

        }

        con.close(); // closing connection
    } catch (Exception e) {
        e.printStackTrace();
    }
    return user;
}
// function for closing the connection once done executing any
statement
public void db_close() throws SQLException
{
    try {
        con.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

// getters and setters for all memembers code firstname, lastname,
registration code section..

public int getCode() {
    return code;
}

public void setCode(int code) {
    this.code = code;
}

public String getFname() {
    return fname;
}

public void setFname(String fname) {
    this.fname = fname;
}

public String getLname() {
    return lname;
}

public void setLname(String lname) {
    lname = lname;
}

public int getReg() {
    return reg;
}

public void setReg(int reg) {
    this.reg = reg;
}

public int getAge() {
    return age;
}

```



```
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getSec() {
        return sec;
    }

    public void setSec(String sec) {
        this.sec = sec;
    }
}
```

### Source Code of FaceDetController

```
import java.io.ByteArrayInputStream;

import java.util.concurrent.Executors;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.TimeUnit;


import org.opencv.core.Mat;

import org.opencv.core.MatOfByte;

import org.opencv.core.MatOfRect;

import org.opencv.core.Rect;

import org.opencv.core.Scalar;

import org.opencv.core.Size;

import org.opencv.imgcodecs.Imgcodecs;

import org.opencv.imgproc.Imgproc;

import org.opencv.objdetect.CascadeClassifier;

import org.opencv.objdetect.Objdetect;

import org.opencv.videoio.VideoCapture;


import javafx.event.Event;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.CheckBox;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;
```

```

public class FaceDetController
{
    // FXML buttons
    @FXML
    private Button cameraButton;

    // the FXML area for showing the current frame
    @FXML
    private ImageView originalFrame;

    // checkboxes for enabling/disabling a classifier
    @FXML
    private CheckBox haarClassifier;

    @FXML
    private CheckBox lbpClassifier;

    // a timer for acquiring the video stream
    private ScheduledExecutorService timer;

    // the OpenCV object that performs the video capture
    private VideoCapture capture;

    // a flag to change the button behavior
    private boolean cameraActive;

    // face cascade classifier
    private CascadeClassifier faceCascade;

    private int absoluteFaceSize;

```

```
protected void init()
{
    this.capture = new VideoCapture();

    this.faceCascade = new CascadeClassifier();
    this.absoluteFaceSize = 0;
}
```

```
@FXML
```

```
protected void startCamera()
{
    // set a fixed width for the frame
    originalFrame.setFitWidth(600);

    // preserve image ratio
    originalFrame.setPreserveRatio(true);

    if (!this.cameraActive)
    {
        // disable setting checkboxes
        this.haarClassifier.setDisable(true);
        this.lbpClassifier.setDisable(true);
    }
}
```

```

        // start the video capture

        this.capture.open(0);

        // is the video stream available?
        if (this.capture.isOpened())
        {
            this.cameraActive = true;

            // grab a frame every 33 ms (30 frames/sec)
            Runnable frameGrabber = new Runnable() {

                @Override
                public void run()
                {
                    Image imageToShow = grabFrame();
                    originalFrame.setImage(imageToShow);
                }
            };

            this.timer =
Executors.newSingleThreadScheduledExecutor();

            this.timer.scheduleAtFixedRate(frameGrabber, 0, 33,
TimeUnit.MILLISECONDS);

            // update the button content

            this.cameraButton.setText("Stop Camera");
        }

```

```

        else

        {

            // log the error

            System.err.println("Failed to open the camera
connection...");

        }

    }

    else

    {

        // the camera is not active at this point

        this.cameraActive = false;

        // update again the button content

        this.cameraButton.setText("Start Camera");

        // enable classifiers checkboxes

        this.haarClassifier.setDisable(false);

        this.lbpClassifier.setDisable(false);


        // stop the timer

        try

        {

            this.timer.shutdown();

            this.timer.awaitTermination(33,
TimeUnit.MILLISECONDS);

        }

        catch (InterruptedException e)

        {

            // log the exception

```

```
        System.err.println("Exception in stopping the frame  
capture, trying to release the camera now... " + e);
```

```
    }
```

```
        // release the camera
```

```
        this.capture.release();
```

```
        // clean the frame
```

```
        this.originalFrame.setImage(null);
```

```
    }
```

```
}
```

```
/**
```

```
 * Get a frame from the opened video stream (if any)
```

```
 *
```

```
 * @return the {@link Image} to show
```

```
 */
```

```
private Image grabFrame()
```

```
{
```

```
    // init everything
```

```
    Image imageToShow = null;
```

```
    Mat frame = new Mat();
```

```
    // check if the capture is open
```

```
    if (this.capture.isOpened())
```

```
    {
```

```
        try
```

```
{

    // read the current frame

    this.capture.read(frame);

    // if the frame is not empty, process it
    if (!frame.empty())
    {

        // face detection

        this.detectAndDisplay(frame);

        // convert the Mat object (OpenCV) to Image
        (JavaFX) imageToShow = mat2Image(frame);

    }

}

catch (Exception e)
{

    // log the (full) error

    System.err.println("ERROR: " + e);

}

}

return imageToShow;

}
```



```

/**
 * Method for face detection and tracking
 *
 * @param frame
 *         it looks for faces in this frame
 */
private void detectAndDisplay(Mat frame)
{
    MatOfRect faces = new MatOfRect();

    Mat grayFrame = new Mat();

    // convert the frame in gray scale
    Imgproc.cvtColor(frame, grayFrame, Imgproc.COLOR_BGR2GRAY);

    // equalize the frame histogram to improve the result
    Imgproc.equalizeHist(grayFrame, grayFrame);

    // compute minimum face size (20% of the frame height, in our
case)

    if (this.absoluteFaceSize == 0)
    {
        int height = grayFrame.rows();

        if (Math.round(height * 0.2f) > 0)
        {
            this.absoluteFaceSize = Math.round(height * 0.2f);
        }
    }
}

```

```

        // detect faces

        this.faceCascade.detectMultiScale(grayFrame, faces, 1.1, 2, 0 |
Objdetect.CASCADE_SCALE_IMAGE,

                                new Size(this.absoluteFaceSize,
this.absoluteFaceSize), new Size());

        // each rectangle in faces is a face: draw them!

        Rect[] facesArray = faces.toArray();

        for (int i = 0; i < facesArray.length; i++)
        {
            Imgproc.rectangle(frame, facesArray[i].tl(),
facesArray[i].br(), new Scalar(7, 255, 90), 4);

            System.out.println(facesArray[i].tl());

            System.out.println(facesArray[i].br());
        }

    }

@FXML

protected void haarSelected(Event event)
{
    // check whether the lpb checkbox is selected and deselect it

    if (this.lbpClassifier.isSelected())

```

```

        this.lbpClassifier.setSelected(false);

        this.checkboxSelection("resources/haarcascades/haarcascade_frontalcatface.xml");
    }

    @FXML
    protected void lbpSelected(Event event)
    {
        // check whether the haar checkbox is selected and deselect it
        if (this.haarClassifier.isSelected())
            this.haarClassifier.setSelected(false);

        this.checkboxSelection("resources/lbpcascades/lbpcascade_frontalface.xml");
    }

    /**
     * Method for loading a classifier trained set from disk
     *
     * @param classifierPath
     *
     * the path on disk where a classifier trained set is
    located
     */

    private void checkboxSelection(String classifierPath)

```

```

{

    // load the classifier(s)

    this.faceCascade.load(classifierPath);

    // now the video capture can start

    this.cameraButton.setDisable(false);

}

/**
 * Convert a Mat object (OpenCV) in the corresponding Image for JavaFX
 *
 * @param frame
 *         the {@link Mat} representing the current frame
 * @return the {@link Image} to show
 */
private Image mat2Image(Mat frame)
{
    // create a temporary buffer

    MatOfByte buffer = new MatOfByte();

    // encode the frame in the buffer, according to the PNG format

    Imgcodecs.imencode(".png", frame, buffer);

    // build and return an Image created from the image encoded in
the

    // buffer

    return new Image(new ByteArrayInputStream(buffer.toArray()));

}

```

```
}
```

#### Source Code of FaceIdentifier

```
package application;
```

```
import java.io.File;
```

```
import java.io.FilenameFilter;
```

```
import java.nio.IntBuffer;
```

```
import static org.bytedeco.javacpp.opencv_core.*;
```

```
import static org.bytedeco.javacpp.opencv_face.createLBPHFaceRecognizer;
```

```
import org.bytedeco.javacpp.opencv_core.Mat;
```

```
import org.bytedeco.javacpp.opencv_core.MatVector;
```

```
import org.bytedeco.javacpp.opencv_face.*;
```

```
import static org.bytedeco.javacpp.opencv_imgcodecs.*;
```

```
import static org.bytedeco.javacpp.opencv_imgproc.*;
```

```
import static org.bytedeco.javacpp.opencv_imgcodecs.imread;
```

```
import static org.bytedeco.javacpp.opencv_imgcodecs.CV_LOAD_IMAGE_GRAYSCALE;
```

```

import org.bytedeco.javacpp.IntPointer;

import org.bytedeco.javacpp.BytePointer;

import org.bytedeco.javacpp.DoublePointer;


public class FaceIdentifier {

    LBPHFaceRecognizer faceRecognizer;


    public File root;

    MatVector images;

    Mat labels;


    public void init() {

        // mention the directory the faces has been saved

        String trainingDir = "./faces";


        root = new File(trainingDir);


        FilenameFilter imgFilter = new FilenameFilter() {

            public boolean accept(File dir, String name) {

                name = name.toLowerCase();

                return name.endsWith(".jpg") || name.endsWith(".pgm")
|| name.endsWith(".png");

            }

        };

    };

```

```

File[] imageFiles = root.listFiles(imgFilter);

this.images = new MatVector(imageFiles.length);

this.labels = new Mat(imageFiles.length, 1, CV_32SC1);
IntBuffer labelsBuf = labels.createBuffer();

int counter = 0;

// reading face images from the folder

for (File image : imageFiles) {

    Mat img = imread(image.getAbsolutePath(),
CV_LOAD_IMAGE_GRAYSCALE);

    // extracting unique face code from the face image names

    /*
information from    this unique face will be used to fetch all other

    I dont put face data on database.

    I just store face indexes on database.

    For example:

    When you train a new face to the system suppose person
named ABC.

    Now this person named ABC has 10(can be more or less) face
image which

    will be saved in the project folder named "/Faces" using a
naming convention such as

    1_ABC1.jpg

```

```
1_ABC2.jpg  
1_ABC3.jpg  
.....  
1_ABC10.jpg
```

The initial value of the file name is the index key in the database table of that person.

the key 1 will be used to fetch data from database.

```
*/  
  
int label = Integer.parseInt(image.getName().split("\\-"  
")[0]);  
  
images.put(counter, img);  
  
labelsBuf.put(counter, label);  
  
counter++;  
}  
  
// face training  
  
//this.faceRecognizer = createLBPHFaceRecognizer();  
this.faceRecognizer = createLBPHFaceRecognizer();  
  
  
this.faceRecognizer.train(images, labels);
```



```
}
```

```
public int recognize(IplImage faceData) {
```

```
    Mat faces = cvarrToMat(faceData);
```

```
    cvtColor(faces, faces, CV_BGR2GRAY);
```

```
    IntPtr label = new IntPtr(1);
```

```
    DoublePointer confidence = new DoublePointer(0);
```

```
    this.faceRecognizer.predict(faces, label, confidence);
```

```
    int predictedLabel = label.get(0);
```

```
    //System.out.println(confidence.get(0));
```

```
    //Confidence value less than 60 means face is known
```

```
    //Confidence value greater than 60 means face is unknown
```

```
    if(confidence.get(0) > 60)
```

```
{  
    //System.out.println("-1");  
    return -1;  
}  
  
return predictedLabel;
```

## Source Code of MotionDetector

```
import org.bytedeco.javacpp.*;

import org.bytedeco.javacpp.opencv_core.IplImage;

import org.bytedeco.javacv.*;

import static org.bytedeco.javacpp.opencv_core.*;

import static org.bytedeco.javacpp.opencv_imgproc.*;


import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics2D;


public class MotionDetector {


    public void init( IplImage frame,Graphics2D g2 ) throws Exception {


        OpenCVFrameConverter.ToIplImage converter = new
        OpenCVFrameConverter.ToIplImage();


        IplImage image = null;
```

```

IplImage prevImage = null;

IplImage diff = null;


CanvasFrame canvasFrame = new CanvasFrame("Motion Detector");

canvasFrame.setCanvasSize(frame.width(), frame.height());


CvMemStorage storage = CvMemStorage.create();


while (canvasFrame.isVisible() && (frame != null)) {

    cvClearMemStorage(storage);


    cvSmooth(frame, frame, CV_GAUSSIAN, 9, 9, 2, 2);

    if (image == null) {

        image = IplImage.create(frame.width(), frame.height(),
IPL_DEPTH_8U, 1);

        cvCvtColor(frame, image, CV_RGB2GRAY);

    } else {

        prevImage = IplImage.create(frame.width(), frame.height(),
IPL_DEPTH_8U, 1);

        prevImage = image;

        image = IplImage.create(frame.width(), frame.height(),
IPL_DEPTH_8U, 1);

        cvCvtColor(frame, image, CV_RGB2GRAY);

    }


    if (diff == null) {

        diff = IplImage.create(frame.width(), frame.height(),
IPL_DEPTH_8U, 1);

```

```

    }

    if (prevImage != null) {

        // perform ABS difference
        cvAbsDiff(image, prevImage, diff);

        // do some threshold for wipe away useless details
        cvThreshold(diff, diff, 64, 255,CV_THRESH_BINARY);

        canvasFrame.showImage(converter.convert(diff));

        // recognize contours

        CvSeq contour = new CvSeq(null);

        cvFindContours(diff, storage, contour,
Loader.sizeof(CvContour.class), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE);

        while (contour != null && !contour.isNull()) {

            if (contour.elem_size() > 0) {

                CvBox2D box = cvMinAreaRect2(contour, storage);

                g2.setColor(Color.RED);

                g2.setFont(new Font("Arial Black", Font.BOLD,
20));

```

```

        String name = "Motion Detected !";

        g2.drawString(name, (int) (50), (50));

        // test intersection
        if (box != null) {
            CvPoint2D32f center = box.center();
            CvSize2D32f size = box.size();

        }
    }
    contour = contour.h_next();
}

}

}

canvasFrame.dispose();
}

}

```