FACE RECOGNITION SYSTEM

A REPORT

Submitted by

DESHIREDDY KRISHNAREDDY (20MIS1069)

APPIREDDY GOWTHAM REDDY (20MIS1175)

In partial fulfilment for the award

Of

M.Tech. Integrated Software Engineering

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

NOVEMBER

ABSTRACT

Face recognition is the process of identifying or verifying an individual's identity by looking at their face. Face recognition for attendance systems is a procedure that uses bio-statistics and computer technology to recognize students' faces. As a result, attendance reports will be generated. The system is put through its paces under various scenarios before moving on to the following steps. The designed technology is cost-effective and requires minimal installation. The programme finds the face's distinct traits in the database and encodes them into a pattern image

CHAPTER 1

Introduction

PURPOSE OF THE SYSTEM

- ✓ To save time of taking attendance and also reduces proxy attendance.
- ✓ To lower the burden of the staff.
- ✓ To make the work easy by using technology.

Literature Survey:

Sr.No	Title and year	Author	Techniques or Algorithm used
1.	Face Detection and Recognition using Open CV	J. Manikandan, S. Lakshmi Prathyusha, P. Sai Kumar, Y. Jaya Chandra, M. Umaditya Hanuman	Haar Feature Selection LBP CASCADE CLASSIFIER HAAR CASCADES CLASSIFIER
2.	Real Time Face Detection and Tracking Using OpenCV	Shubham Dhere Shivkumar Hipparagi Prof. P Y Kumbhar	HAAR CASCADES CLASSIFIER
3.	Face Recognition Based Attendance System	Nandhini R, Duraimurugan N, S.P.Chokkalingam	LBP CASCADE CLASSIFIER HAAR CASCADES CLASSIFIER

Algorithm used:

HAAR Cascade Algorithm

- The HAAR cascade is a machine learning approach where a cascade function is trained from a lot of positive and negative images. Positive images are those images that consist of faces, and negative images are without faces. In face detection, image features are treated as numerical information extracted from the pictures that can distinguish one image from another.
- We apply every feature of the algorithm on all the training images. Every image is given equal weight at the starting. It founds the best threshold which will categorize the faces to positive and negative. There may be errors and misclassifications. We select the features with a minimum error rate, which means these are the features that best classifies the face and non-face images.
- All possible sizes and locations of each kernel are used to calculate the plenty of features

- OpenCV provides two applications to train cascade classifier opencv_haartraining and opencv_traincascade. These two applications store the classifier in the different file format.
- For training, we need a set of samples. There are two types of samples:
- Negative sample: It is related to non-object images.
- Positive samples: It is a related image with detect objects.
- A set of negative samples must be prepared manually, whereas the collection of positive samples are created using the opencv_createsamples utility.
- Extracting the Histograms from the image: The image is generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, let's consider the following image:
- There are various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. We can use the Euclidean distance based on the following formula:

CHAPTER 2

2.3 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- **PROCESSOR:** Intel Core i5-8259U, or AMD Ryzen 7 2700X
- **GRAPHICS CARD:** NVIDIA GT 1030 2GB or Quadro P1000

Face recognition based attendance system

A python GUI integrated attendance system using face recognition to take attendance.

In this python project, I have made an attendance system which takes attendance by using face recognition technique. I have also intergrated it with GUI (Graphical user interface) so it can be easy to use by anyone. GUI for this project is also made on python using tkinter.

TECHNOLOGY USED:

1. tkinter for whole GUI
2. OpenCV for taking images and face recognition (cv2.face.LBPHFaceRecognizer_create())
3. CSV, Numpy, Pandas, datetime etc. for other purposes.

FEATURES:

1. Easy to use with interactive GUI support.
2. Password protection for new person registration.
3. Creates/Updates CSV file for deatils of students on registration.
4. Creates a new CSV file everyday for attendance and marks attendance with proper date and time.
5. Displays live attendance updates for the day on the main screen in tabular format with Id, name, date and time.

2.4 INPUT AND OUTPUT

The major inputs and outputs and major functions of the system are follows:

Input

- Faces of all the students should be scanned and uploaded to the database along with the student details (Name, ID, password etc).
- Then upload the necessary details to involve in the storing operation.

Output

- The pictures of the student faces which have been stored in the database will be recognized when they try scanning the face.
- The data will be seen directly in the database

2.5 INPUT DESIGN

- Input design is a part of overall system design. The main objective during the input design as given below.
- **Input States:** User sets the personal data to be stored in the database like name, reg.no etc.
- In the end when the face is recognized he can see his name also.
- **Input Media:** we can also review the details entered.

2.6 LIMITATIONS

- Massive data storage load: The ML technology used for facial recognition requires high-performance data storage that may not be available to all users.
- Detection is vulnerable.
- A potential breach of privacy.
- Difficult in maintaining the large amount of increased overhead.

2.7 DRAWBACKS IN EXISTING SYSTEM:

- Manual systems put pressure on the manual system put pressure on the people to be correct in all details of their work at all times the problem being that people are not perfect however each of us wishes we are
- These attendance systems are manual
- There is always a chance of forgery (one person say signing the presence of the other one) since these are manually so there is a great risk of error.
- More manpower is required
- Calculations related to attendance are done manually (total classes attended in a month) which is prone to error.
- It is difficult to maintain a database or register in manual system.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

Face recognition using OpenCV is about introducing Attendance system using face recognition is a procedure of recognizing students by using face biostatistics based on the high-definition monitoring and other computer technologies. ... The traditional process of making attendance and present biometric systems is vulnerable to proxies.

PURPOSE

The purpose of this Software Requirement Specification (SRS) is to help the project. Instead of using the conventional methods, this proposed system aims to develop an automated system that records the student's attendance by using facial recognition technology. The main objective of this work is to make the attendance marking and management system efficient, time saving, simple and easy.

3.2 FUNCTIONAL REQUIREMENTS

The system functional requirement describes activities and Services that must provide:

- Tracking and marking student attendance by Facial recognition in specific time.

- Allowing the faculty to modify the student Absent or late arrivals.

- Showing the names of students with the exact Time stamp i.e., exact time of entering the class.

3.3 NON-FUNCTIONAL REQUIREMENTS:

3.3.1 Usability

This system can be effectively used in taking attendance for Students, Teachers, Staffs in both schools and colleges, ATM's, identifying duplicate voters, passport and visa verification, driving license verification, in defense, competitive and other exams, in governments and private sectors.

3.3.2 Reliability

Face attendance is practically 100% accurate.

3.3.3 Performance

The system works on face recognition where each student in the class is photographed and their details are stored in a server. The teacher can then record the attendance by just clicking some pictures of the classroom. The system will recognize the faces and verify the presence or absence of each student.

3.3.4 Supportability

There are many benefits facial recognition can offer society, from preventing crimes and increasing safety and security to reducing unnecessary human interaction and labor. In some instances, it can even help support medical efforts.

3.3.5 Packaging

a. The system must be able to run on the Windows OS beginning with Windows 7, and must be able to run on future releases such as the upcoming Windows 10

b. The software must incorporate a license key authentication process.

c. The packaging must come with a manual that details the use of the system, and also the instructions on how to use the program. This manual may be included either in a booklet that comes with the software, or on the disc that the software itself is on.

3.2.6 Implementation -

The face recognition is implemented with the help of Principal Component Analysis (PCA) algorithm. The system will recognize the face of the student and saves the response in database automatically. The system also includes the feature of retrieving the list of students who are absent in a particular day.

3.2.7 Interfacing -

The system must offer an easy and simple way of viewing the attendance.

SYSTEM DESIGN

4.3 Diagram

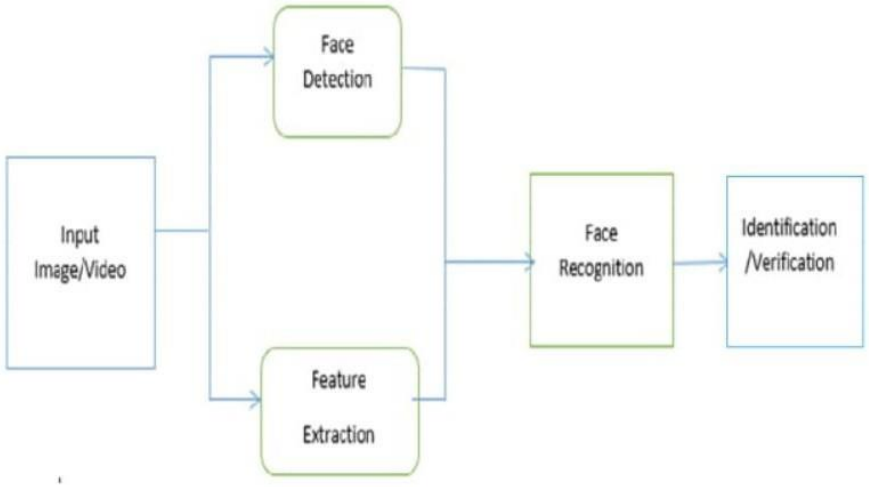


Figure 1: Block diagram of Face Detection

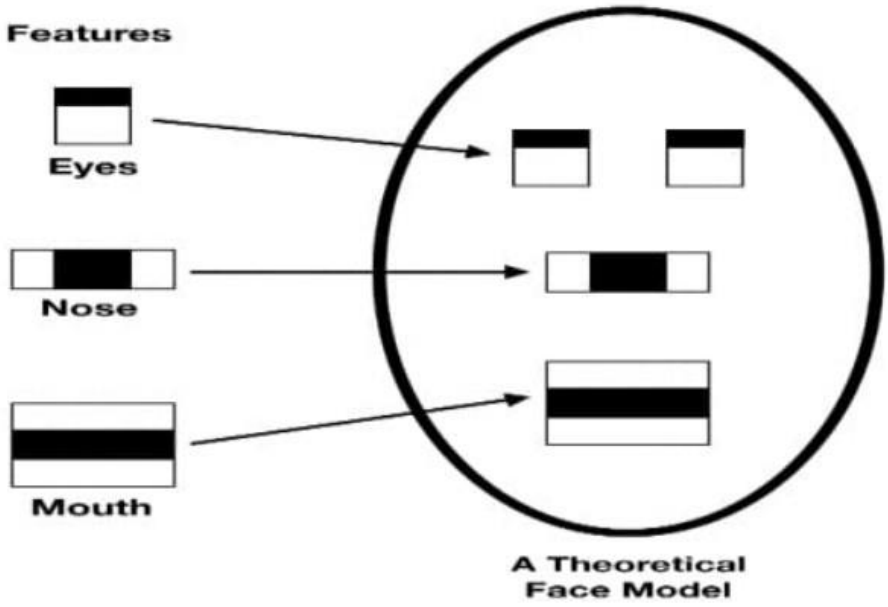
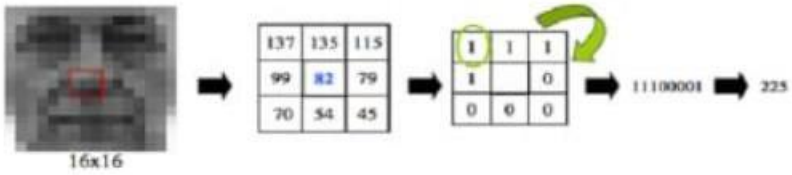


Figure 2: Detection of Theoretical Face model using Haar like features

Eigen Face representation for Face Recognition

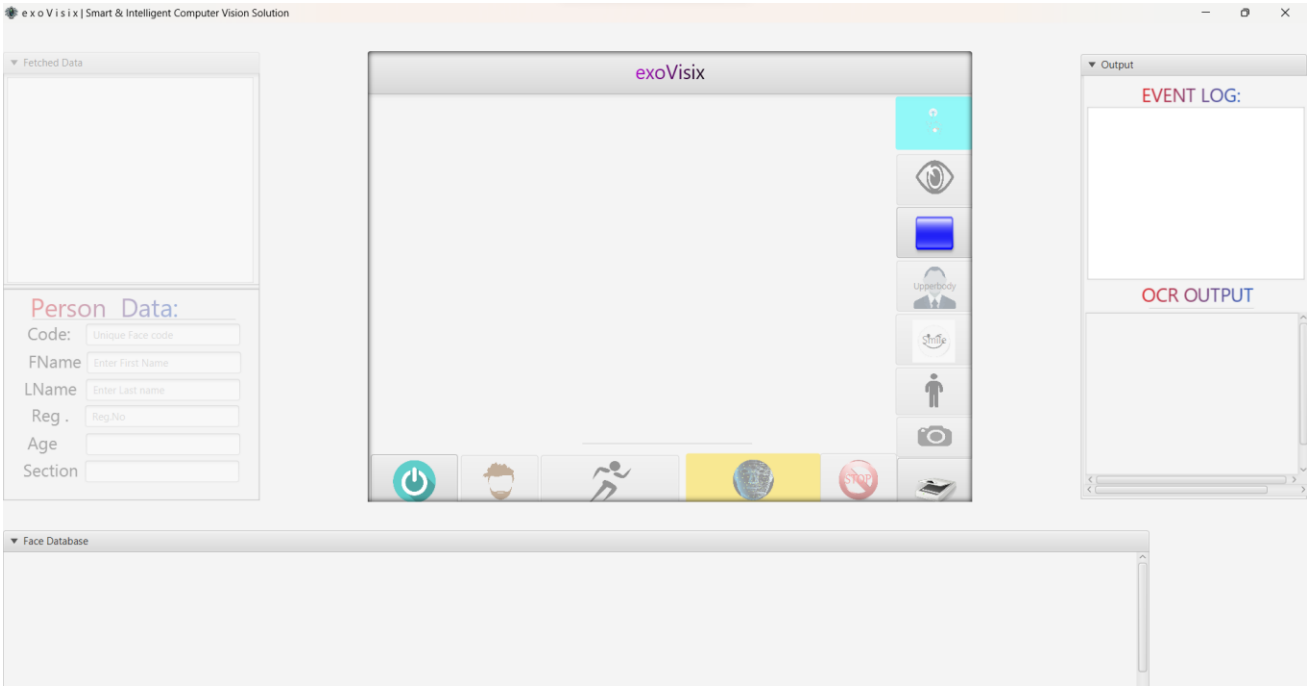


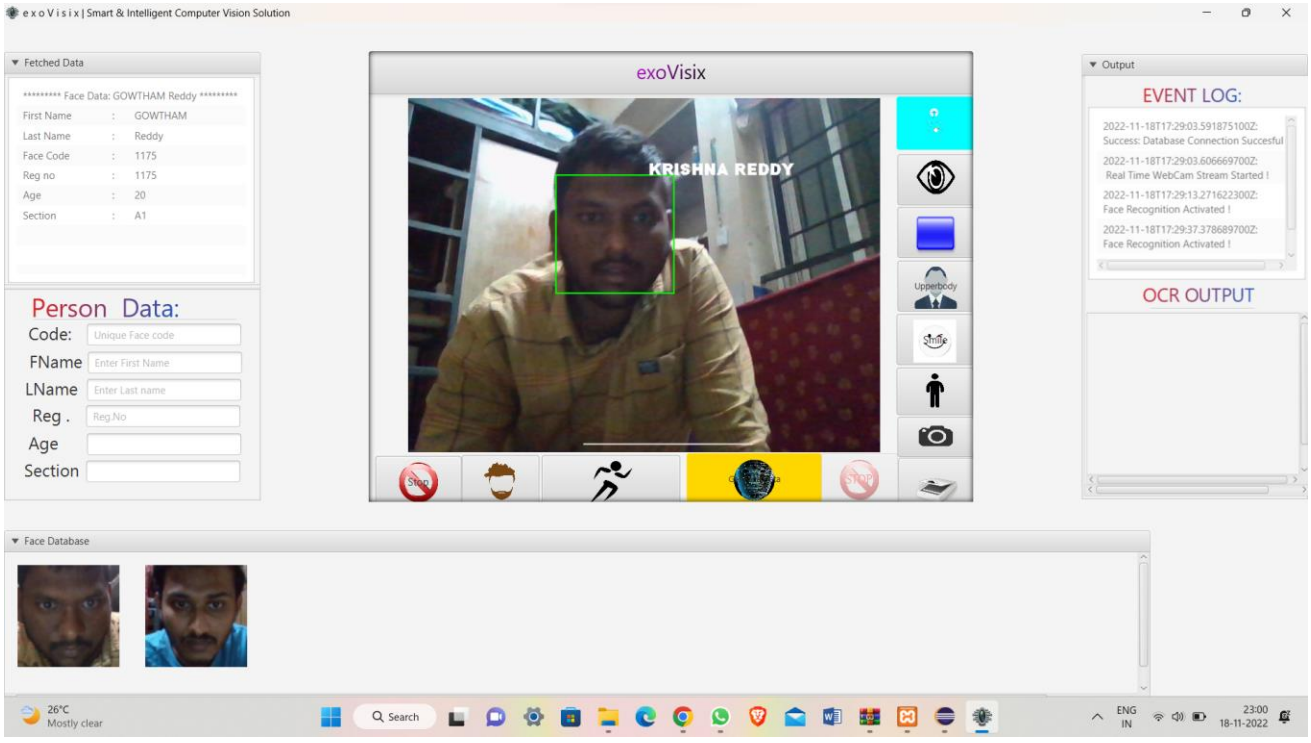
LBP Conversion to Binary

Chapter 5

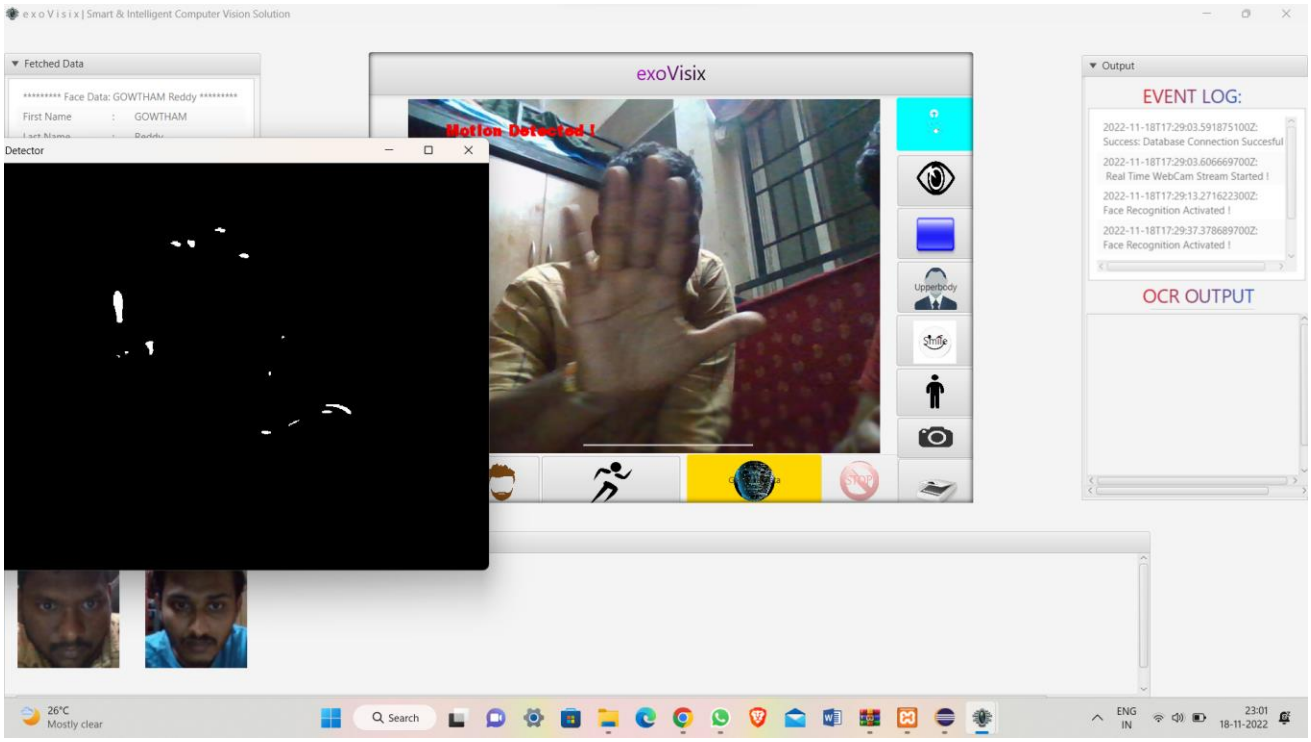
SCREENSHOTS:

User Interface of Attendance System:





Motion Detector:


exo V i s i x | Smart & Intelligent Computer Vision Solution
Fetched Data
********** Face Data: GOWTHAM Reddy **********
First Name : GOWTHAM
exoVisix
Motion Detected !
Detector
Output
EVENT LOG:
2022-11-18T17:29:03.591875100Z:
Success: Database Connection Succesful
2022-11-18T17:29:03.606669700Z:
Real Time WebCam Stream Started !
2022-11-18T17:29:13.271622300Z:
Face Recognition Activated !
2022-11-18T17:29:37.378689700Z:
Face Recognition Activated !
OCR OUTPUT
Upperbody
Stop
26°C
Mostly clear
Search
ENG
IN
23:01
18-11-2022

CODES:

```
package application;


import java.awt.AWTException;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Robot;
import java.awt.image.BufferedImage;
import javax.swing.JPanel;

import org.bytedeco.javacpp.opencv_core.CvScalar;
import org.bytedeco.javacpp.opencv_core.IplImage;
import org.bytedeco.javacpp.opencv_imgproc.CvMoments;
import org.bytedeco.javacv.CanvasFrame;
import org.bytedeco.javacv.FrameGrabber;
```

```
import org.bytedeco.javacv.Java2DFrameConverter;
import org.bytedeco.javacv.OpenCVFrameConverter;

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

public class ColoredObjectTracker implements Runnable {


    final int INTERVAL = 1;// 1sec
    final int CAMERA_NUM = 0; // Default camera for this time
    FrameGrabber grabber;
    OpenCVFrameConverter.ToIplImage converter ;
    IplImage img;

    /**
     * Correct the color range- it depends upon the object,
camera quality,
     * environment.
     */
    static CvScalar rgba_min = cvScalar(0, 0, 130, 0);// RED wide
dabur birko
    static CvScalar rgba_max = cvScalar(80, 80, 255, 0);

    IplImage image;
    CanvasFrame canvas ;
    CanvasFrame path ;
    int ii = 0;
    JPanel jp = new JPanel();
```

```
    public void init() {
        //
canvas.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_
CLOSE);

        canvas = new CanvasFrame("Web Cam Live");
        path = new CanvasFrame("Detection");

    //path.setDefaultCloseOperation(javax.swing.JFrame.EXIT_O
N_CLOSE);
        path.setContentPane(jp);
    }

    public void run() {
        try {
            grabber =
FrameGrabber.createDefault(CAMERA_NUM);
            converter = new
OpenCVFrameConverter.ToIplImage();
            grabber.start();

            int posX = 0;
            int posY = 0;
            while (true) {
                img = converter.convert(grabber.grab());
                if (img != null) {
                    // show image on window
                    cvFlip(img, img, 1);// l-r =
90_degrees_steps_anti_clockwise

    canvas.showImage(converter.convert(img));
```

```
                        IplImage detectThrs =
getThresholdImage(img);

                        CvMoments moments = new
CvMoments();
                        cvMoments(detectThrs, moments, 1);

                        double mom10 =
cvGetSpatialMoment(moments, 1, 0);
                        double mom01 =
cvGetSpatialMoment(moments, 0, 1);
                        double area =
cvGetCentralMoment(moments, 0, 0);
                        posX = (int) (mom10 / area);
                        posY = (int) (mom01 / area);
                        // only if its a valid position
                        if (posX > 0 && posY > 0) {
                              paint(img, posX, posY);
                        }
                  }
                  // Thread.sleep(INTERVAL);
               }
         } catch (Exception e) {
         }
   }

   private void paint(IplImage img, int posX, int posY) {
         Graphics g = jp.getGraphics();
         path.setSize(img.width(), img.height());
         g.clearRect(0, 0, img.width(), img.height());
         g.setColor(Color.RED);
```

```
        Robot mouseControler = null ; // For moving mouse
pointer
        try {
            mouseControler = new Robot();
        } catch (AWTException e) {
            e.printStackTrace();
        }


        mouseControler.mouseMove(posX,posY);

        g.fillOval(posX, posY, 40, 40);
        g.drawString("Detected Here", posX, posY);
        g.drawOval(posX, posY, 40, 40);
        System.out.println("X,Y: "+ posX + " , " + posY);

    }

    private IplImage getThresholdImage(IplImage orgImg) {
        IplImage imgThreshold =
cvCreateImage(cvGetSize(orgImg), 8, 1);
        //
        cvInRangeS(orgImg, rgba_min, rgba_max,
imgThreshold);// red

        cvSmooth(imgThreshold, imgThreshold, CV_MEDIAN,
15,0,0,0);
        //cvSaveImage(++ii + "dsmthreshold.jpg",
imgThreshold);
        return imgThreshold;
```

```
    }


    public IplImage Equalize(BufferedImage bufferedimg) {
        Java2DFrameConverter converter1 = new
Java2DFrameConverter();
        OpenCVFrameConverter.ToIplImage converter2 = new
OpenCVFrameConverter.ToIplImage();
        IplImage iploriginal =
converter2.convert(converter1.convert(bufferedimg));
        IplImage srcimg = IplImage.create(iploriginal.width(),
iploriginal.height(), IPL_DEPTH_8U, 1);
        IplImage destimg = IplImage.create(iploriginal.width(),
iploriginal.height(), IPL_DEPTH_8U, 1);
        cvCvtColor(iploriginal, srcimg, CV_BGR2GRAY);
        cvEqualizeHist(srcimg, destimg);
        return destimg;
    }
    public void stop() {
        img=null;


        try {
            grabber.stop();
        } catch (org.bytedeco.javacv.FrameGrabber.Exception e)
{

            e.printStackTrace();
        }
        try {
            grabber.release();
```

```
            } catch (org.bytedeco.javacv.FrameGrabber.Exception e)
{

                e.printStackTrace();
            }
            grabber = null;
        }
}
```

DATABSE.JAVA

```
package application;

import java.sql.*;
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;

    public final String Database_name = "ghosteye";
    public final String Database_user = "root";
    public final String Database_pass = "";

    public Connection con;
```

```
    public boolean init() throws SQLException {
        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) {

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

                return false;

            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

            return false;
        }

        return true;
    }

    public void insert() {
        String sql = "INSERT INTO face_bio (code, first_name,
```

```
last_name, reg, age , section) VALUES (?, ?, ?, ?,?,?)";

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

        try {

            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);

            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();
        }

    }
```

```
    public ArrayList<String> getUser(int inCode) throws
SQLException {

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

        try {

            Database app = new Database();

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

            Statement s = con.createStatement();

            ResultSet rs = s.executeQuery(sql);

            while (rs.next()) {

                /*
                 * app.setCode(rs.getInt(2));
app.setFname(rs.getString(3));
                 * app.setLname(rs.getString(4));
app.setReg(rs.getInt(5));
                 * app.setAge(rs.getInt(6));
app.setSec(rs.getString(7));
                 */

                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));

                    /*
                     * System.out.println(app.getCode());
                     * System.out.println(app.getFname());
                     * System.out.println(app.getLname());
                     * System.out.println(app.getReg());
                     * System.out.println(app.getAge());
                     * System.out.println(app.getSec());
                     */

                    // nString="Name:" + rs.getString(3)+"
"+rs.getString(4) +
                    // "\nReg:" + app.getReg()
+"\nAge:"+app.getAge() +"\nSection:"
                    // +app.getSec() ;

                    // System.out.println(nString);
                }

                con.close(); // closing connection
            } catch (Exception e) {
                e.getStackTrace();
            }
            return user;
    }

    public void db_close() throws SQLException
    {
```

```
		try {
			con.close();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}


	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;
    }

}
```

Facedetection.java

```
package application;
```

```
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 FaceDetectionController
{
    // 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/haarcascad
e_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_f
rontalface.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()));
    }

}
```

FACE DETECTOR:

```
package application;

import application.FaceRecognizer;
import java.awt.BasicStroke;
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.sql.SQLException;
import java.time.Instant;
import java.util.ArrayList;

import javax.imageio.ImageIO;

import org.bytedeco.javacpp.FlyCapture2.ImageMetadata;
import org.bytedeco.javacpp.Loader;
import org.bytedeco.javacpp.opencv_objdetect;
import org.bytedeco.javacpp.helper.opencv_core;
import org.bytedeco.javacpp.opencv_core.Mat;
import org.bytedeco.javacv.CanvasFrame;
import org.bytedeco.javacv.Frame;
import org.bytedeco.javacv.FrameGrabber;
import org.bytedeco.javacv.Java2DFrameConverter;
import org.bytedeco.javacv.OpenCVFrameConverter;
import org.bytedeco.javacv.OpenCVFrameGrabber;
import static org.bytedeco.javacpp.opencv_core.*;
import static org.bytedeco.javacpp.opencv_imgproc.*;
import static org.bytedeco.javacpp.opencv_imgcodecs.*;
import static org.bytedeco.javacpp.opencv_objdetect.*;

import javafx.collections.FXCollections;
```

```
import javafx.collections.ObservableList;
import javafx.embed.swing.SwingFXUtils;
import javafx.fxml.FXML;
import javafx.scene.chart.PieChart.Data;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.image.ImageView;
import javafx.scene.image.WritableImage;
import application.Database;
import application.MotionDetector;
import application.ColoredObjectTracker;
import application.SquareDetector;

public class FaceDetector implements Runnable {

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

    FaceRecognizer faceRecognizer = new FaceRecognizer();
    MotionDetector motionDetector = new MotionDetector();
    OpenCVFrameConverter.ToIplImage grabberConverter = new
OpenCVFrameConverter.ToIplImage();
    Java2DFrameConverter paintConverter = new
Java2DFrameConverter();
    ArrayList<String> output = new ArrayList<String>();

    @FXML
    public Label ll;
    private Exception exception = null;

    private int count = 0;
```

```
    public String classiferName;
    public File classifierFile;


    public boolean saveFace = false;
    public boolean isRecFace = false;
    public boolean isOutput = false;
    public boolean isOcrMode = false;
    public boolean isMotion = false;
    public boolean isEyeDetection = false;
    public boolean isSmile = false;
    public boolean isUpperBody = false;
    public boolean isFullBody = false;
    private boolean stop = false;

    private CvHaarClassifierCascade classifier = null;
    private CvHaarClassifierCascade classifierEye = null;
    private CvHaarClassifierCascade classifierSideFace = null;
    private CvHaarClassifierCascade classifierUpperBody = null;
    private CvHaarClassifierCascade classifierFullBody = null;
    private CvHaarClassifierCascade classifierSmile = null;
    private CvHaarClassifierCascade classifierEyeglass = null;


    public CvMemStorage storage = null;
    private FrameGrabber grabber = null;
    private IplImage grabbedImage = null, temp, temp2,
grayImage = null, smallImage = null;
    public ImageView frames2;
    public ImageView frames;
```

```
    private CvSeq faces = null;
    private CvSeq eyes = null;
    private CvSeq smile = null;
    private CvSeq upperBody = null;
    private CvSeq sideface = null;
    private CvSeq fullBody = null;



    int recogniseCode;
    public int code;
    public int reg;
    public int age;

    public String fname; //first name
    public String Lname; //last name
    public String sec; //section
    public String name;

    public void init() {
          faceRecognizer.init();

          setClassifier("haar/haarcascade_frontalface_alt.xml");
          setClassifierEye("haar/haarcascade_eye.xml");

    setClassifierEyeGlass("haar/haarcascade_eye_tree_eyeglasse
s.xml");

    setClassifierSideFace("haar/haarcascade_profileface.xml");
          setClassifierFullBody("haar/haarcascade_fullbody.xml");
```

```
    setClassifierUpperBody("haar/haarcascade_upperbody.xml");
        setClassifierSmile("haar/haarcascade_smile.xml");

    }

    public void start() {
        try {
            new Thread(this).start();
        } catch (Exception e) {
            if (exception == null) {
                exception = e;

            }
        }
    }

    public void run() {
        try {
            try {
                grabber =
OpenCVFrameGrabber.createDefault(0); //parameter 0 default
camera , 1 for secondary

                grabber.setImageWidth(700);
                grabber.setImageHeight(700);
                grabber.start();

                grabbedImage =
grabberConverter.convert(grabber.grab());
```

```
                storage = CvMemStorage.create();
            } catch (Exception e) {
                if (grabber != null)
                    grabber.release();
                grabber = new OpenCVFrameGrabber(0);
                grabber.setImageWidth(700);
                grabber.setImageHeight(700);
                grabber.start();
                grabbedImage =
grabberConverter.convert(grabber.grab());

            }
            int count = 15;
            grayImage =
cvCreateImage(cvGetSize(grabbedImage), 8, 1); //converting
image to grayscale

            //reducing the size of the image to speed up the
processing
            smallImage =
cvCreateImage(cvSize(grabbedImage.width() / 4,
grabbedImage.height() / 4), 8, 1);

            stop = false;

            while (!stop && (grabbedImage =
grabberConverter.convert(grabber.grab())) != null) {

                Frame frame =
grabberConverter.convert(grabbedImage);
                BufferedImage image =
```

```
paintConverter.getBufferedImage(frame, 2.2 /
grabber.getGamma());
                    Graphics2D g2 = image.createGraphics();

                    if (faces == null) {
                        cvClearMemStorage(storage);

                        //creating a temporary image
                        temp =
cvCreateImage(cvGetSize(grabbedImage),
grabbedImage.depth(), grabbedImage.nChannels());

                        cvCopy(grabbedImage, temp);

                        cvCvtColor(grabbedImage, grayImage,
CV_BGR2GRAY);
                        cvResize(grayImage, smallImage,
CV_INTER_AREA);

                        //cvHaarDetectObjects(image, cascade,
storage, scale_factor, min_neighbors, flags, min_size, max_size)
                        faces = cvHaarDetectObjects(smallImage,
classifier, storage, 1.1, 3, CV_HAAR_DO_CANNY_PRUNING);
                        //face detection

                        CvPoint org = null;
                        if (grabbedImage != null) {

                            if (isEyeDetection) {          //eye
detection logic
                                eyes =
```

```
cvHaarDetectObjects(smallImage, classifierEye, storage, 1.1, 3,

    CV_HAAR_DO_CANNY_PRUNING);

                                        if (eyes.total() == 0) {
                                            eyes =
cvHaarDetectObjects(smallImage, classifierEyeglass, storage,
1.1, 3,

    CV_HAAR_DO_CANNY_PRUNING);

                                        }

                                        printResult(eyes, eyes.total(),
g2);

                                    }

                                    if (isFullBody) { //full body detection
logic
                                        fullBody =
cvHaarDetectObjects(smallImage, classifierFullBody, storage,
1.1, 3,

    CV_HAAR_DO_CANNY_PRUNING);

                                        if (fullBody.total() > 0) {
                                            printResult(fullBody,
fullBody.total(), g2);
                                        }
```

```
                }

                if (isUpperBody) {
                    try {
                        upperBody =
cvHaarDetectObjects(smallImage, classifierUpperBody, storage,
1.1, 3,
    CV_HAAR_DO_CANNY_PRUNING);

                        if (upperBody.total() > 0) {
    printResult(upperBody, upperBody.total(), g2);
                        }

                    } catch (Exception e) {

                        e.printStackTrace();
                    }
                }

                if (isSmile) {
                    try {
                        smile =
cvHaarDetectObjects(smallImage, classifierSmile, storage, 1.1, 3,
    CV_HAAR_DO_CANNY_PRUNING);

                        if (smile != null) {
                            printResult(smile,
smile.total(), g2);
```

```
                    }
                } catch (Exception e) {

                    e.printStackTrace();
                }

            }

            if (isOcrMode) {
                try {

                    OutputStream os = new
FileOutputStream("captures.png");
                    ImageIO.write(image,
"PNG", os);
                } catch (IOException e) {

                    e.printStackTrace();
                }
            }

            isOcrMode = false;

            if (faces.total() == 0) {
                faces =
cvHaarDetectObjects(smallImage, classifierSideFace, storage,
1.1, 3,

    CV_HAAR_DO_CANNY_PRUNING);

            }
```

```
                        if (faces != null) {
                            g2.setColor(Color.green);
                            g2.setStroke(new
BasicStroke(2));
                            int total = faces.total();

                            for (int i = 0; i < total; i++) {

                                //printing rectange box
where face detected frame by frame
                                CvRect r = new
CvRect(cvGetSeqElem(faces, i));
                                g2.drawRect((r.x() * 4),
(r.y() * 4), (r.width() * 4), (r.height() * 4));

                                CvRect re = new
CvRect((r.x() * 4), r.y() * 4, (r.width() * 4), r.height() * 4);

                                cvSetImageROI(temp, re);

                                // File f = new
File("captures.png");

                                org = new CvPoint(r.x(),
r.y());

                                if (isRecFace) {
                                    String
names="Unknown Person!";
                                    this.recogniseCode =
```

```
faceRecognizer.recognize(temp);
                                                    //getting recognised
user from the database
                                                    if(recogniseCode != -
1)
                                                    {
                                                        database.init();
                                                        user = new
ArrayList<String>();
                                                        user =
database.getUser(this.recogniseCode);
                                                        this.output =
user;
                                                        names =
user.get(1) + " " + user.get(2);
                                                    }
                                                    //printing recognised
person name into the frame
    g2.setColor(Color.WHITE);
                                                    g2.setFont(new
Font("Arial Black", Font.BOLD, 20));
                                                    g2.drawString(names,
(int) (r.x() * 6.5), r.y() * 4);
                                                }
```

```
                                        if (saveFace) { //saving
captured face to the disk
                                            //keep it in mind that
face code should be unique to each person
                                            String fName =
"faces/" + code + "-" + fname + "_" + Lname + "_" + count +
".jpg";
                                            cvSaveImage(fName,
temp);
                                            count++;

                                        }

                                    }
                                    this.saveFace = false;
                                    faces = null;
                                }

                                WritableImage showFrame =
SwingFXUtils.toFXImage(image, null);


    javafx.application.Platform.runLater(new Runnable(){

                                    @Override
                                     public void run() {
                                    frames.setImage(showFrame);
                                     }
                                     });
```

```
                        if (isMotion) {
                            new Thread(() -> {

                                try {


    motionDetector.init(grabbedImage, g2);

                                } catch
(InterruptedException ex) {
                                } catch (Exception e) {

                                    e.printStackTrace();
                                }

                            }).start();

                        }
                        isMotion = false;

                    }
                    cvReleaseImage(temp);
                }

            }

        } catch (Exception e) {
            if (exception == null) {
                exception = e;

            }
```

```
        }
    }

    public void stop() {
        stop = true;

        grabbedImage = grayImage = smallImage = null;
        try {
            grabber.stop();
        } catch (org.bytedeco.javacv.FrameGrabber.Exception e)
{

            e.printStackTrace();
        }
        try {
            grabber.release();
        } catch (org.bytedeco.javacv.FrameGrabber.Exception e)
{

            e.printStackTrace();
        }
        grabber = null;
    }

    public void setClassifier(String name) {

        try {

            setClassiferName(name);
            classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");
```

```
                if (classifierFile == null || classifierFile.length() <= 0)
{
                        throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work
around a known bug.
                Loader.load(opencv_objdetect.class);
                classifier = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the
classifier file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }

    public void setClassifierEye(String name) {

        try {
```

```
                classiferName = name;
                classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0)
{
                        throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work
around a known bug.
                Loader.load(opencv_objdetect.class);
                classifierEye = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the
classifier file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }
```

```
    public void setClassifierSmile(String name) {

        try {

            setClassiferName(name);
            classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");

            if (classifierFile == null || classifierFile.length() <= 0)
{
                throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
            }

            // Preload the opencv_objdetect module to work
around a known bug.
            Loader.load(opencv_objdetect.class);
            classifierSmile = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
            classifierFile.delete();
            if (classifier.isNull()) {
                throw new IOException("Could not load the
classifier file.");
            }

        } catch (Exception e) {
            if (exception == null) {
                exception = e;
```

```
                }
            }

        }

    public void printResult(CvSeq data, int total, Graphics2D g2) {
            for (int j = 0; j < total; j++) {
                CvRect eye = new CvRect(cvGetSeqElem(eyes, j));

                g2.drawOval((eye.x() * 4), (eye.y() * 4), (eye.width()
* 4), (eye.height() * 4));

            }
    }

    public void setClassifierSideFace(String name) {

            try {

                classiferName = name;
                classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0)
{
                    throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work
around a known bug.
```

```
                Loader.load(opencv_objdetect.class);
                classifierSideFace = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the
classifier file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }

    public void setClassifierFullBody(String name) {

        try {

                setClassiferName(name);
                classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0)
{
                        throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
```

```
                }

                // Preload the opencv_objdetect module to work
around a known bug.
                Loader.load(opencv_objdetect.class);
                classifierFullBody = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the
classifier file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }

    public void setClassifierEyeGlass(String name) {

        try {

                setClassiferName(name);
                classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");
```

```
                if (classifierFile == null || classifierFile.length() <= 0)
{
                        throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work
around a known bug.
                Loader.load(opencv_objdetect.class);
                classifierEyeglass = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the
classifier file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }

    public void setClassifierUpperBody(String name) {

        try {
```

```
                classiferName = name;
                classifierFile =
Loader.extractResource(classiferName, null, "classifier", ".xml");

                if (classifierFile == null || classifierFile.length() <= 0)
{
                        throw new IOException("Could not extract \""
+ classiferName + "\" from Java resources.");
                }

                // Preload the opencv_objdetect module to work
around a known bug.
                Loader.load(opencv_objdetect.class);
                classifierUpperBody = new
CvHaarClassifierCascade(cvLoad(classifierFile.getAbsolutePath())
);
                classifierFile.delete();
                if (classifier.isNull()) {
                        throw new IOException("Could not load the
classifier file.");
                }

        } catch (Exception e) {
                if (exception == null) {
                        exception = e;

                }
        }

    }
```

```
public String getClassiferName() {
	return classiferName;
}

public void setClassiferName(String classiferName) {
	this.classiferName = classiferName;
}

public void setFrames2(ImageView frames2) {
	this.frames2 = frames2;
}

public void setSmile(boolean isSmile) {
	this.isSmile = isSmile;
}

public void setUpperBody(boolean isUpperBody) {
	this.isUpperBody = isUpperBody;
}

public void setFullBody(boolean isFullBody) {
	this.isFullBody = isFullBody;
}

public boolean isEyeDetection() {

	return isEyeDetection;
}

public void setEyeDetection(boolean isEyeDetection) {
	this.isEyeDetection = isEyeDetection;
```

```
}

public boolean getOcrMode() {
	return isOcrMode;
}

public void setOcrMode(boolean isOcrMode) {
	this.isOcrMode = isOcrMode;
}

public void destroy() {
}

public boolean isMotion() {
	return isMotion;
}

public void setMotion(boolean isMotion) {
	this.isMotion = isMotion;
}

public ArrayList<String> getOutput() {
	return output;
}

public void clearOutput() {
	this.output.clear();
}

public void setOutput(ArrayList<String> output) {
	this.output = output;
```

```
}

public int getRecogniseCode() {
	return recogniseCode;
}

public void setRecogniseCode(int recogniseCode) {
	this.recogniseCode = recogniseCode;
}

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;
    }

    public void setFrame(ImageView frame) {
        this.frames = frame;
    }
```

```
    public void setSaveFace(Boolean f) {
        this.saveFace = f;
    }

    public Boolean getIsRecFace() {
        return isRecFace;
    }

    public void setIsRecFace(Boolean isRecFace) {
        this.isRecFace = isRecFace;
    }


}
```

FACE RECOGNIZER:

```
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_GR
AYSCALE;

import org.bytedeco.javacpp.IntPointer;
import org.bytedeco.javacpp.BytePointer;
import org.bytedeco.javacpp.DoublePointer;

public class FaceRecognizer {

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

        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;

    }
}
```

MAIN.JAVA

```
package application;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;

// One man's constants is another mans variable.
// Think twice but code once.
// Happy Coding  :)

/*************************************************************
****
*      e x o V i s i x | The GhostEye              *
*            Anup Kumar Sarkar                    *
*         me.anup.sarkar@gmail.com                      *
```

```
*         fb.com/i.am.anup.aronno                    *
*                                                    *
******************************************************
*****/

//Feel Free to communicate

public class Main extends Application {


    @Override
    public void start(Stage primaryStage) {
        try {
            BorderPane root =
(BorderPane)FXMLLoader.load(getClass().getResource("Sample.f
xml"));
            Scene scene = new Scene(root,1350,720);


    scene.getStylesheets().add(getClass().getResource("applicatio
n.css").toExternalForm());
            primaryStage.getIcons().add(new
Image("logo.png"));
        primaryStage.setTitle("e x o V i s i x | Smart & Intelligent
Computer Vision Solution ");

            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
```

```
    }

    public static void main(String[] args) {
        launch(args);


    }
}
```

MOTION DETECTOR.JAVA

```
package application;


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();
  }


}
```

OCR.JAVA

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

import static org.bytedeco.javacpp.lept.*;
import static org.bytedeco.javacpp.tesseract.*;
import java.awt.Font;
```

```
import java.awt.FontFormatException;
import java.awt.GraphicsEnvironment;
import java.io.File;
import java.io.IOException;




public class OCR
{
   public  String init() throws FontFormatException
   {
    try {
        GraphicsEnvironment ge =
           GraphicsEnvironment.getLocalGraphicsEnvironment();
        ge.registerFont(Font.createFont(Font.TRUETYPE_FONT,
new File("f2.ttf")));
    } catch (IOException e) {
        //Handle exception
    }

     BytePointer outText;

     TessBaseAPI api=new TessBaseAPI();




      //to use tesseract api,at first you have to install tesseract
with desired language training data on your system.After That
you have to mention
```

```
        //the installation folder.
        if(api.Init("C:/tessdata", "eng") != 0)
        {
         System.out.println("could not initialize tesseract");
         System.exit(1);

        }

        //

        //For Bengali Language
        /*    if(api.Init("C:/tesseract-ocr/tessdata", "ben") != 0)
        {
         System.out.println("could not initialize tesseract");
         System.exit(1);

        }*/

        //read an image from default location for ocr output
        PIX image=pixRead("ocr_test.png");
        if(image==null)
        {
         System.err.println("Could not opened the image or Image
not found ");

        }

        api.SetImage(image);

        outText=api.GetUTF8Text();
```

```
        String output= outText.getString();

        api.End();
        outText.deallocate();
        pixDestroy(image);

        return output;
    }
}
```

SAMPLE CONTROLLER:

```
package application;

import javafx.scene.control.Button;
import javafx.scene.control.ContentDisplay;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;

import javafx.scene.control.ProgressIndicator;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.geometry.Insets;
import javafx.scene.control.TextField;
import javafx.scene.control.TitledPane;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.TilePane;
```

```
import javafx.scene.paint.Stop;
import javafx.scene.text.Text;
import javafx.scene.text.TextFlow;

import java.awt.FontFormatException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.sql.SQLException;
import java.time.Instant;
import java.util.ArrayList;

import application.FaceDetector;
import application.Database;
import application.OCR;
import application.Database;

public class SampleController {

    //*********************************************
*****************************************
    //Mention The file location path where the face will be saved
& retrieved

    public String filePath="./faces";


    //*********************************************
*****************************************
    @FXML
    private Button startCam;
```

```
@FXML
private Button stopBtn;
@FXML
private Button motionBtn;
@FXML
private Button eyeBtn;
@FXML
private Button shapeBtn;
@FXML
private Button upperBodyBtn;
@FXML
private Button fullBodyBtn;
@FXML
private Button smileBtn;
@FXML
private Button gesture;
@FXML
private Button gestureStop;
@FXML
private Button saveBtn;
@FXML
private Button ocrBtn;
@FXML
private Button capBtn;
@FXML
private Button recogniseBtn;
@FXML
private Button stopRecBtn;
@FXML
private ImageView frame;
@FXML
```

```
private ImageView motionView;
@FXML
private AnchorPane pdPane;
@FXML
private TitledPane dataPane;
@FXML
private TextField fname;
@FXML
private TextField lname;
@FXML
private TextField code;
@FXML
private TextField reg;
@FXML
private TextField sec;
@FXML
private TextField age;
@FXML
public ListView<String> logList;
@FXML
public ListView<String> output;
@FXML
public ProgressIndicator pb;
@FXML
public Label savedLabel;
@FXML
public Label warning;
@FXML
public Label title;
@FXML
public TilePane tile;
```

```
    @FXML
    public TextFlow ocr;
//**********************************************************
**************************************
    FaceDetector faceDetect = new FaceDetector();   //Creating
Face detector object
    ColoredObjectTracker cot = new ColoredObjectTracker();
//Creating Color Object Tracker object
    Database database = new Database();          //Creating
Database object

    OCR ocrObj = new OCR();
    ArrayList<String> user = new ArrayList<String>();
    ImageView imageView1;

    public static ObservableList<String> event =
FXCollections.observableArrayList();
    public static ObservableList<String> outEvent =
FXCollections.observableArrayList();

    public boolean enabled = false;
    public boolean isDBready = false;


    //**********************************************************
**************************************
    public void putOnLog(String data) {

            Instant now = Instant.now();

            String logs = now.toString() + ":\n" + data;
```

```
        event.add(logs);

        logList.setItems(event);

    }

    @FXML
    protected void startCamera() throws SQLException {


    //*****************************************************
*****************************************
        //initializing objects from start camera button event
        faceDetect.init();

        faceDetect.setFrame(frame);

        faceDetect.start();

        if (!database.init()) {

            putOnLog("Error: Database Connection Failed ! ");

        } else {
            isDBready = true;
            putOnLog("Success: Database Connection Succesful
! ");
        }
```

```
    //*****************************************************
*****************************************
        //Activating other buttons
        startCam.setVisible(false);
        eyeBtn.setDisable(false);
        stopBtn.setVisible(true);
        //ocrBtn.setDisable(false);
        capBtn.setDisable(false);
        motionBtn.setDisable(false);
        gesture.setDisable(false);
        saveBtn.setDisable(false);

        if (isDBready) {
            recogniseBtn.setDisable(false);
        }

        dataPane.setDisable(false);
        // shapeBtn.setDisable(false);
        smileBtn.setDisable(false);
        fullBodyBtn.setDisable(false);
        upperBodyBtn.setDisable(false);

        if (stopRecBtn.isDisable()) {
            stopRecBtn.setDisable(false);
        }

    //*****************************************************
*****************************************


        tile.setPadding(new Insets(15, 15, 55, 15));
```

```
        tile.setHgap(30);


    //*********************************************************
**********************************************
        //Picture Gallary

        String path = filePath;

        File folder = new File(path);
        File[] listOfFiles = folder.listFiles();

        //Image reader from the mentioned folder
        for (final File file : listOfFiles) {

            imageView1 = createImageView(file);
            tile.getChildren().addAll(imageView1);
        }
        putOnLog(" Real Time WebCam Stream Started !");


    //*********************************************************
**********************************************
    }
    int count = 0;

    @FXML
    protected void faceRecognise() {


        faceDetect.setIsRecFace(true);
```

```
            // printOutput(faceDetect.getOutput());

            recogniseBtn.setText("Get Face Data");

            //Getting detected faces
            user = faceDetect.getOutput();

            if (count > 0) {

                //Retrieved data will be shown in Fetched Data
pane
                String t = "********* Face Data: " + user.get(1) + "
" + user.get(2) + " *********";

                outEvent.add(t);

                String n1 = "First Name\t\t:\t" + user.get(1);

                outEvent.add(n1);

                output.setItems(outEvent);

                String n2 = "Last Name\t\t:\t" + user.get(2);

                outEvent.add(n2);

                output.setItems(outEvent);

                String fc = "Face Code\t\t:\t" + user.get(0);

                outEvent.add(fc);
```

```
            output.setItems(outEvent);

            String r = "Reg no\t\t\t:\t" + user.get(3);

            outEvent.add(r);

            output.setItems(outEvent);

            String a = "Age \t\t\t\t:\t" + user.get(4);

            outEvent.add(a);

            output.setItems(outEvent);
            String s = "Section\t\t\t:\t" + user.get(5);

            outEvent.add(s);

            output.setItems(outEvent);

        }

        count++;

        putOnLog("Face Recognition Activated !");

        stopRecBtn.setDisable(false);

    }

    @FXML
```

```
    protected void stopRecognise() {

        faceDetect.setIsRecFace(false);
        faceDetect.clearOutput();

        this.user.clear();

        recogniseBtn.setText("Recognise Face");

        stopRecBtn.setDisable(true);

        putOnLog("Face Recognition Deactivated !");

    }

    @FXML
    protected void startMotion() {

        faceDetect.setMotion(true);
        putOnLog("motion Detector Activated !");

    }

    @FXML
    protected void saveFace() throws SQLException {

        //Input Validation
        if (fname.getText().trim().isEmpty() ||
reg.getText().trim().isEmpty() || code.getText().trim().isEmpty())
{
```

```
            new Thread(() -> {

                try {
                    warning.setVisible(true);

                    Thread.sleep(2000);

                    warning.setVisible(false);

                } catch (InterruptedException ex) {
                }

            }).start();

        } else {
            //Progressbar
            pb.setVisible(true);

            savedLabel.setVisible(true);

            new Thread(() -> {

                try {

                    faceDetect.setFname(fname.getText());

                    faceDetect.setFname(fname.getText());
                    faceDetect.setLname(lname.getText());

faceDetect.setAge(Integer.parseInt(age.getText()));
```

```
faceDetect.setCode(Integer.parseInt(code.getText()));
                        faceDetect.setSec(sec.getText());

faceDetect.setReg(Integer.parseInt(reg.getText()));

                        database.setFname(fname.getText());
                        database.setLname(lname.getText());

database.setAge(Integer.parseInt(age.getText()));

database.setCode(Integer.parseInt(code.getText()));
                        database.setSec(sec.getText());

database.setReg(Integer.parseInt(reg.getText()));

                        database.insert();

                        javafx.application.Platform.runLater(new
Runnable(){

                                @Override
                                public void run() {
                                        pb.setProgress(100);
                                }
                                });




                        savedLabel.setVisible(true);
                        Thread.sleep(2000);
```

```
                    javafx.application.Platform.runLater(new
Runnable(){

                        @Override
                        public void run() {
                            pb.setVisible(false);
                        }
                        });




                    javafx.application.Platform.runLater(new
Runnable(){

                        @Override
                        public void run() {
                        savedLabel.setVisible(false);
                        }
                        });

                } catch (InterruptedException ex) {
                }

            }).start();

            faceDetect.setSaveFace(true);
```

```
        }

    }

    @FXML
    protected void stopCam() throws SQLException {

        faceDetect.stop();

        startCam.setVisible(true);
        stopBtn.setVisible(false);

        /* this.saveFace=true; */

        putOnLog("Cam Stream Stopped!");

        recogniseBtn.setDisable(true);
        saveBtn.setDisable(true);
        dataPane.setDisable(true);
        stopRecBtn.setDisable(true);
        eyeBtn.setDisable(true);
        smileBtn.setDisable(true);
        fullBodyBtn.setDisable(true);
        upperBodyBtn.setDisable(true);

        database.db_close();
        putOnLog("Database Connection Closed");
        isDBready=false;
    }
```

```
@FXML
protected void ocrStart() {

	try {

		Text text1 = new Text(ocrObj.init());

		text1.setStyle("-fx-font-size: 14; -fx-fill: blue;");

		ocr.getChildren().add(text1);

	} catch (FontFormatException e) {

		e.printStackTrace();
	}

}

@FXML
protected void capture() {

	faceDetect.setOcrMode(true);

}

@FXML
protected void startGesture() {

	faceDetect.stop();
	cot.init();
```

```
        Thread th = new Thread(cot);
        th.start();

        gesture.setVisible(false);
        gestureStop.setVisible(true);

    }

    @FXML
    protected void startEyeDetect() {

        faceDetect.setEyeDetection(true);
        eyeBtn.setDisable(true);

    }

    @FXML
    protected void upperBodyStart() {

        faceDetect.setUpperBody(true);
        ;
        upperBodyBtn.setDisable(true);

    }

    @FXML
    protected void fullBodyStart() {

        faceDetect.setFullBody(true);
        fullBodyBtn.setDisable(true);
```

```
}

@FXML
protected void smileStart() {

	faceDetect.setSmile(true);
	smileBtn.setDisable(true);

}

@FXML
protected void stopGesture() {

	cot.stop();
	faceDetect.start();

	gesture.setVisible(true);
	gestureStop.setVisible(false);

}

@FXML
protected void shapeStart() {

	// faceDetect.stop();

	SquareDetector shapeFrame = new SquareDetector();
	shapeFrame.loop();

}
```

```
    private ImageView createImageView(final File imageFile) {

        try {
            final Image img = new Image(new
FileInputStream(imageFile), 120, 0, true, true);
            imageView1 = new ImageView(img);

            imageView1.setStyle("-fx-background-color:
BLACK");
            imageView1.setFitHeight(120);

            imageView1.setPreserveRatio(true);
            imageView1.setSmooth(true);
            imageView1.setCache(true);

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        return imageView1;
    }

}
```

SQUARE DETECTOR.JAVA

```
package application;

import javax.swing.JFrame;

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

```
import org.bytedeco.javacpp.opencv_core.CvContour;
import org.bytedeco.javacpp.opencv_core.CvMemStorage;
import org.bytedeco.javacpp.opencv_core.CvPoint;
import org.bytedeco.javacpp.opencv_core.CvSeq;
import org.bytedeco.javacpp.opencv_core.CvSize;
import org.bytedeco.javacpp.opencv_core.CvSlice;
import org.bytedeco.javacpp.opencv_core.IplImage;
import org.bytedeco.javacv.*;

import javafx.scene.image.ImageView;
import static
org.bytedeco.javacpp.helper.opencv_core.CV_RGB;
import static org.bytedeco.javacpp.opencv_core.*;
import static org.bytedeco.javacpp.opencv_imgproc.*;
import org.bytedeco.javacv.OpenCVFrameGrabber;

public class SquareDetector {

    Java2DFrameConverter paintConverter = new
Java2DFrameConverter();
    IplImage frame;
    ImageView frameShow;
    public Exception exception = null;
    public IplImage grabbedImage = null, grayImage = null,
smallImage = null;
    public CanvasFrame canvass = new CanvasFrame("Webcam");

    // use default camera
    public OpenCVFrameGrabber grabber = new
OpenCVFrameGrabber(0);;
```

```
    public void setFrameShow(ImageView frameShow) {
        this.frameShow = frameShow;
    }

    public void setFrame(IplImage frame) {
        this.frame = frame;
    }

    int thresh = 50;
    IplImage img = null;
    IplImage img0 = null;
    CvMemStorage storage = null;
    CvMemStorage storage2 = null;
    String wndname = "Square Detection Demo";

    // Java spesific

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

    // helper function:
    // finds a cosine of angle between vectors
    // from pt0->pt1 and from pt0->pt2
    double angle(CvPoint pt1, CvPoint pt2, CvPoint pt0) {
        double dx1 = pt1.x() - pt0.x();
        double dy1 = pt1.y() - pt0.y();
        double dx2 = pt2.x() - pt0.x();
        double dy2 = pt2.y() - pt0.y();

        return (dx1 * dx2 + dy1 * dy2) / Math.sqrt((dx1 * dx1 +
dy1 * dy1) * (dx2 * dx2 + dy2 * dy2) + 1e-10);
```

```
    }

    // returns sequence of squares detected on the image.
    // the sequence is stored in the specified memory storage
    CvSeq findSquares4(IplImage img, CvMemStorage storage) {
        // Java translation: moved into loop
        // CvSeq contours = new CvSeq();
        int i, c, l, N = 11;
        CvSize sz = cvSize(img.width() & -2, img.height() & -2);
        IplImage timg = cvCloneImage(img); // make a copy of
input image
        IplImage gray = cvCreateImage(sz, 8, 1);
        IplImage pyr = cvCreateImage(cvSize(sz.width() / 2,
sz.height() / 2), 8, 3);
        IplImage tgray = null;
        // Java translation: moved into loop
        // CvSeq result = null;
        // double s = 0.0, t = 0.0;

        // create empty sequence that will contain points -
        // 4 points per square (the square's vertices)
        CvSeq squares = cvCreateSeq(0,
Loader.sizeof(CvSeq.class), Loader.sizeof(CvPoint.class),
storage);

        // select the maximum ROI in the image
        // with the width and height divisible by 2
        cvSetImageROI(timg, cvRect(0, 0, sz.width(),
sz.height()));

        // down-scale and upscale the image to filter out the
```

```
noise
        cvPyrDown(timg, pyr, 7);
        cvPyrUp(pyr, timg, 7);
        tgray = cvCreateImage(sz, 8, 1);

        // find squares in every color plane of the image
        for (c = 0; c < 3; c++) {
            // extract the c-th color plane
            cvSetImageCOI(timg, c + 1);
            cvCopy(timg, tgray);

            // try several threshold levels
            for (l = 0; l < N; l++) {
                // hack: use Canny instead of zero threshold
level.
                // Canny helps to catch squares with gradient
shading
                if (l == 0) {
                    // apply Canny. Take the upper threshold
from slider
                    // and set the lower to 0 (which forces
edges merging)
                    cvCanny(tgray, gray, 0, thresh, 5);
                    // dilate canny output to remove
potential
                    // holes between edge segments
                    cvDilate(gray, gray, null, 1);
                } else {
                    // apply threshold if l!=0:
                    // tgray(x,y) = gray(x,y) < (l+1)*255/N ?
255 : 0
```

```
                    cvThreshold(tgray, gray, (l + 1) * 255 / N,
255, CV_THRESH_BINARY);
                }

                // find contours and store them all as a list
                // Java translation: moved into the loop
                CvSeq contours = new CvSeq();
                cvFindContours(gray, storage, contours,
Loader.sizeof(CvContour.class), CV_RETR_LIST,
                        CV_CHAIN_APPROX_SIMPLE,
cvPoint(0, 0));

                // test each contour
                while (contours != null && !contours.isNull()) {
                    // approximate contour with accuracy
proportional
                    // to the contour perimeter
                    // Java translation: moved into the loop
                    CvSeq result = cvApproxPoly(contours,
Loader.sizeof(CvContour.class), storage, CV_POLY_APPROX_DP,
                            cvContourPerimeter(contours)
* 0.02, 0);
                    // square contours should have 4 vertices
after
                    // approximation
                    // relatively large area (to filter out noisy
contours)
                    // and be convex.
                    // Note: absolute value of an area is used
because
                    // area may be positive or negative - in
```

```
accordance with the
                        // contour orientation
                        if (result.total() == 4 &&
Math.abs(cvContourArea(result, CV_WHOLE_SEQ, 0)) > 1000
                                &&
cvCheckContourConvexity(result) != 0) {

                            // Java translation: moved into loop
                            double s = 0.0, t = 0.0;

                            for (i = 0; i < 5; i++) {
                                // find minimum angle between
joint
                                // edges (maximum of cosine)
                                if (i >= 2) {
                                    // Java translation:
                                    // Comment from the
HoughLines.java sample code:
                                    // " Based on JavaCPP, the
equivalent of the C
                                    // code:
                                    // CvPoint* line =
                                    //
(CvPoint*)cvGetSeqElem(lines,i);
                                    // CvPoint first=line[0];
                                    // CvPoint second=line[1];
                                    // is:
                                    // Pointer line =
cvGetSeqElem(lines, i);
                                    // CvPoint first = new
                                    // CvPoint(line).position(0);
```

```
                                        // CvPoint second = new
                                        // CvPoint(line).position(1);
                                        // "
                                        // ... so after some trial and
error this seem
                                        // to work
                                        // t = fabs(angle(
                                        //
(CvPoint*)cvGetSeqElem( result, i ),
                                        //
(CvPoint*)cvGetSeqElem( result, i-2 ),
                                        //
(CvPoint*)cvGetSeqElem( result, i-1 )));
                                        t = Math.abs(angle(new
CvPoint(cvGetSeqElem(result, i)),
                                                new
CvPoint(cvGetSeqElem(result, i - 2)),
                                                new
CvPoint(cvGetSeqElem(result, i - 1))));
                                        s = s > t ? s : t;
                                    }
                                }

                                // if cosines of all angles are small
                                // (all angles are ~90 degree) then
write quandrange
                                // vertices to resultant sequence
                                if (s < 0.3)
                                    for (i = 0; i < 4; i++) {
                                        cvSeqPush(squares,
cvGetSeqElem(result, i));
```

```
                                }
                            }

                            // take the next contour
                            contours = contours.h_next();
                        }
                    }
                }

                // release all the temporary images
                cvReleaseImage(gray);
                cvReleaseImage(pyr);
                cvReleaseImage(tgray);
                cvReleaseImage(timg);

                return squares;
            }

            // the function draws all the squares in the image
            void drawSquares(IplImage img, CvSeq squares) {

                // Java translation: Here the code is somewhat different
from the C
                // version.
                // I was unable to get straight forward CvPoint[] arrays
                // working with "reader" and the
"CV_READ_SEQ_ELEM".

                // CvSeqReader reader = new CvSeqReader();

                IplImage cpy = cvCloneImage(img);
```

```
        int i = 0;

        // Used by attempt 3
        // Create a "super"-slice, consisting of the entire
sequence of squares
        CvSlice slice = new CvSlice(squares);

        // initialize reader of the sequence
        // cvStartReadSeq(squares, reader, 0);

        // read 4 sequence elements at a time (all vertices of a
square)
        for (i = 0; i < squares.total(); i += 4) {
            CvPoint rect = new CvPoint(4);
            IntPointer count = new IntPointer(1).put(4);
            // get the 4 corner slice from the "super"-slice
            cvCvtSeqToArray(squares, rect,
slice.start_index(i).end_index(i + 4));

            cvPolyLine(cpy, rect.position(0), count, 1, 1,
CV_RGB(0, 255, 0), 3, CV_AA, 0);

            // Frame frame =converter.convert(cpy);
            // canvass.showImage(converter.convert(cpy));

            /*
             * BufferedImage image =
paintConverter.getBufferedImage(frame, 2.2
             * );
             *
             *
```

```
                * WritableImage display =
SwingFXUtils.toFXImage(image, null);
                *
                *
                * frameShow.setImage(display);
                */

               // return frame;
               // canvas.showImage(converter.convert(cpy));

          }
          canvass.showImage(converter.convert(cpy));

          cvReleaseImage(cpy);
     }

     public void loop() {
          //
canvass.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

          storage2 = CvMemStorage.create();

          try {

               storage = cvCreateMemStorage(0);

               try {
                    grabber.start();
               } catch
(org.bytedeco.javacv.FrameGrabber.Exception e1) {
                    // TODO Auto-generated catch block
```

```
                e1.printStackTrace();
            }

            // get framerate
            double frameRate = grabber.getFrameRate();
            long wait = (long) (1000 / (frameRate == 0 ? 10 :
frameRate));

            // keep capturing
            while (true) {
                Thread.sleep(wait);
                grabbedImage =
converter.convert(grabber.grab());

                // drawSquares(grabbedImage,
findSquares4(grabbedImage,
                // storage2));

                //
canvas.showImage(converter.convert(grabbedImage));
                // show grabbed image

                cvClearMemStorage(storage2);
                cvClearMemStorage(storage);

                if (grabbedImage != null) {

                    //
canvas.showImage(converter.convert(grabbedImage));

                    drawSquares(grabbedImage,
```

```
findSquares4(grabbedImage, storage));
                        }

                    }

                    // show stack trace
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

}
```

CHAPTER -6
CONCLUSION AND FUTURE ENHANCEMENT

So the project completes here. There are many changes and improvements to be done which will be worked on.

THANK YOU

DESHIREDDY KRISHNAREDDY

20MIS1069

APPIREDDY GOWTHAM REDDY

20MIS1175

References

https://www.ijrte.org/wp-content/uploads/papers/v8i5/E5753018520.pdf

https://ijrest.net/downloads/volume-4/issue-4/pid-ijrest-44201715.pdf

https://www.researchgate.net/publication/341876647_Face_Recognition_based_Attendance_Management_System

https://link.springer.com/chapter/10.1007/978-3-030-69921-5_47

OpenCV - Face Detection in a Picture (tutorialspoint.com)