A

Project Report

On

Face Detection and Recognition

SUBMITTED TO

BIRLA INSTITUTE OF TECHNOLOGY

FOR THE AWARD OF THE DEGREE OF

Bachelor of Engineering

By

CHETAN AGRAWAL (Roll No. BE/4013/10)

NIKHIL CHANDRA (Roll No. BE/4027/10)

HARSH BALYAN (Roll No. BE/4050/10)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BIRLA INSTITUTE OF TECHNOLOGY, MESRA (RANCHI)

2014

BIRLA INSTITUTE OF TECHNOLOGY, JAIPUR

STUDENT DECLARATION

We hereby declare that the matter embodied in this project is genuine work done by us and has not been submitted either to this university nor any other university/institute for the fulfilment of the requirement of any course of study. All the aspects of the project include deep study of the problem and an effort to provide a solution to this problem.

Chetan Agrawal	Nikhil Chandra	Harsh Balyan
(BE/4013/10)	(BE/4027/10)	(BE/4050/10)

BIRLA INSTITUTE OF TECHNOLOGY, JAIPUR

CERTIFICATE

This is to certify that the project entitled “Face Detection and Recognition” has been carried out by the under mentioned students under my guidance in partial fulfilment of award of degree of Bachelor of Engineering in Computer Science & Engineering from Birla Institute of Technology, Jaipur during the academic year 2013-2014 .

Team Members:
1. Chetan Agrawal (BE/4013/10)
2. Nikhil Chandra (BE/4027/10)
3. Harsh Balyan (BE/4050/10)

Mr. Subhash Chand Gupta
Assistant Professor,
Department of C.S.E
B.I.T-Mesra
(Project Mentor)

Dr. Madhavi Sinha
Head of Department,
Department of C.S.E
B.I.T-Mesra

BIRLA INSTITUTE OF TECHNOLOGY, JAIPUR

ACKNOWLEDGEMENT

We would like to express our profound sense of gratitude to Mr. Subhash Chand Gupta, for his valuable suggestions, constant encouragement and genuine interest in promotion of the project work from the beginning with domain idea that enabled us to pursue the work.

We would like to extend our sincere thanks and deep gratitude to our Professors, Dr. Shripal Vijayvargiya and Dr. Madhavi Sinha who infused in us the spirit to work and have helped us throughout.

Finally, we would like to thank all those who have directly or indirectly helped us in successful completion of this project. We are thankful to all our colleagues for their co-operation and support.

- Chetan Agrawal (BE/4013/10)
- Nikhil Chandra (BE/4027/10)
- Harsh Balyan (BE/4050/10)

ABSTRACT

Objective

The project aims at developing a Face Detection and Recognition system.

Motivation

These days' systems like the attendance system installed at various institutions are prone to inconsistency and corruption due to human involvement. This software facilitates in minimizing human involvement by working in an automated framework. Also the use of biometrics makes proxy negligible. Further this system can be conjoined with an embedded system for authentication services e.g. Access to some facility only after authentication.

Methodology

The software would initially require the people whom it wants to recognize to register with the system. The system would take a series of images of each of them which would later be used to recognize them. At the time of recognition the person would have to stand in front of the camera for a minimum amount of time to allow the system to recognize him. Then the system would process the series of images from the video input. Processing would be done through Digital Image Processing involving feature extraction from the captured images. This data would then be cross checked with the entries in the database. The match would be found using the Eigenface algorithm. The project will be implemented in C++ with the help of OpenCV (Open Source Computer Vision).

Innovativeness and Usefulness

The software can be installed in various offices and institutions as a Centralized Attendance System. It can also be extended as an Authentication System. It would minimize human involvement thereby reducing the chances of meddling with the data.

Contents

I. Software Requirement Specification

1. Introduction

1.1 Purpose: The document provides a brief description of the software, its application area and intended audience .It provides an insight into the functional requirements, non- functional requirements and the hardware support required. It also includes a brief introduction to the algorithms that would be used for face detection and recognition.

1.2 Scope: Human face recognition is a difficult problem in computer vision. Face recognition is challenging because it is a real world problem. The human face is a complex, natural object that tends not to have easily (automatically) identified edges and features. Because of this, it is difficult to develop a mathematical model of the face that can be used as prior knowledge when analyzing a particular image. Face recognition as a security measure has the advantage that it can be done quickly, perhaps even in real time, and does not require extensive equipment to implement. It also does not pose a particular inconvenience to the subject being identified, as is the case in retinal scans.

1.3 Definitions, acronyms, and abbreviations

- CV – Computer Vision
- DIP –Digital Image Processing
- ANN – Artificial Neural Network
- SRS-Software Requirement Specification
- CSV-Comma Separated Value

1.4 References

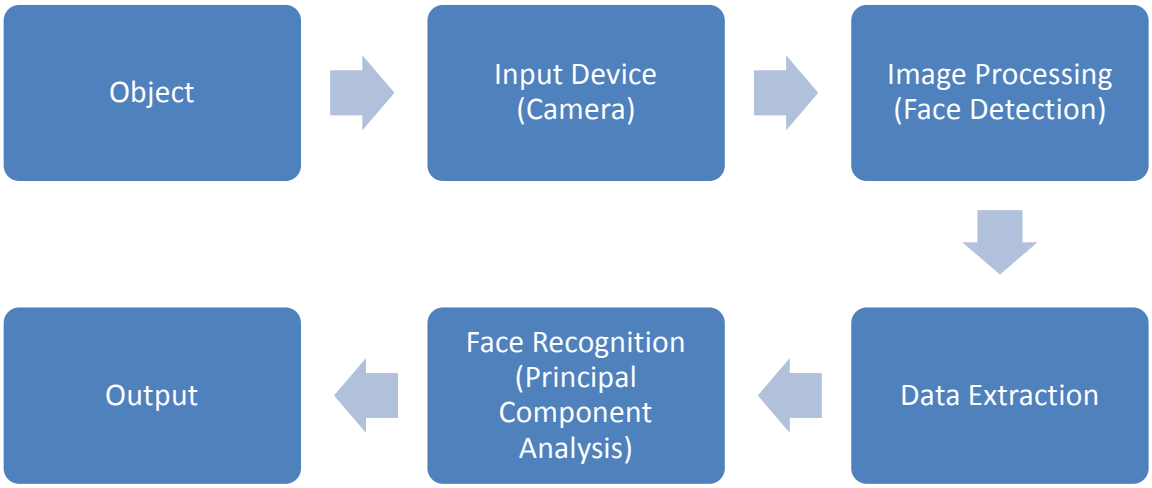
- IEEE-SRS www.ieee.org
- http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html
- http://www.face-rec.org/databases/

1.5 Overview: The rest of the document is divided into four sections .The Overall Description highlights the general factors that affect the product and its requirements. It also includes user perspective, characteristics, dependencies and functions. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in the next section Specific Requirements of the SRS, and makes them easier to understand. This is then followed by Index and Abbreviations which makes the SRS easier to use.

2. Overall Description

2.1 Product Perspective: The product is independent and self-contained but can be extended to be used with embedded systems that require the use of facial recognition.

2.2 Product Functions



The software would initially require the people whom it wants to recognize to register with the system. The system would take a series of images of each of them which would later be used to recognize them. At the time of recognition the person would have to stand in front of the camera for a minimum amount of time to allow the system to recognize him.

Then the system would process the series of images from the video input. Processing would be done through Digital Image Processing involving data extraction from the captured images. This data would then be cross checked with the entries in the database. The match would be found using Eigenfaces.

2.3 User Characteristics

- **Administrator** - Basic computer skills such as use of computer peripherals, installation of program and data retrieval and awareness about the developed application.
- **Maintenance Personnel –** Structural organisation and data flow of the program with detailed knowledge of OpenCV, C++ Programming Language.
- **User–** Knowledge about the interface of the system.

2.4 Constraints

- **Safety and Security Constraints –** The equipment should not be tampered.
- **Hardware Requirement –** The recognition process is limited to processors speed.

2.5 Assumptions and Dependencies

- Image Processing is not a fool proof method of authentication, since human face appearance is subject to various sporadic changes on a day-to-day basis (shaving, hair style, acne, etc.), as well gradual changes over time (aging). Because of this, face recognition is perhaps best used as an augmentation for other identification techniques.
- The view of the camera should not be obscured.
- OpenCV libraries and C/C++ compiler must be installed on the system where the software will be installed.

3. Requirements

3.1 Functional Requirements

3.1.1 Image Capturing for database

- **Input:** Digital Image
- **Process:** Feature Extraction, Image Pre –processing and adding an entry to the reference table (.csv file)
- **Output:** JPEG file.
- **Error:** Display Error Message

3.1.2 Image Capturing at the time of recognition

- **Input :** Digital Image
- **Process:** Image processing and Face detection. Comparison of detected face with the existing database using Eigenface algorithm.
- **Output:** Display the predicted label of the recognised image.
- **Error:** Cannot recognise the object.

3.2 Non Functional Requirement

- Power backup.
- Security measures to prevent damage.
- Enough secondary storage.
- Detection/Recognition mechanism should be efficient and should not take more than few seconds.

3.3 Hardware Requirements

- Webcam
- Display monitor
- Processing Unit
- Processor : Core 2 Duo with processing speed 2.0 GHz
- Memory: 2GB RAM
- 50 GB Hard Disk

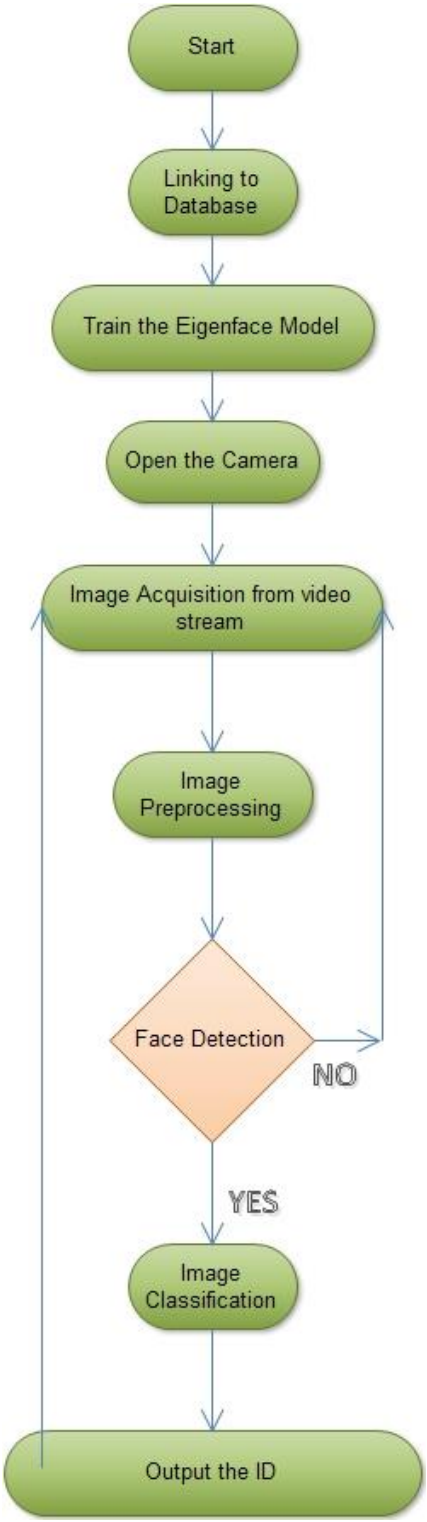
3.4 Software Requirements

- Windows XP or above.
- OpenCV Libraries
- Visual Studio 2008 and above.

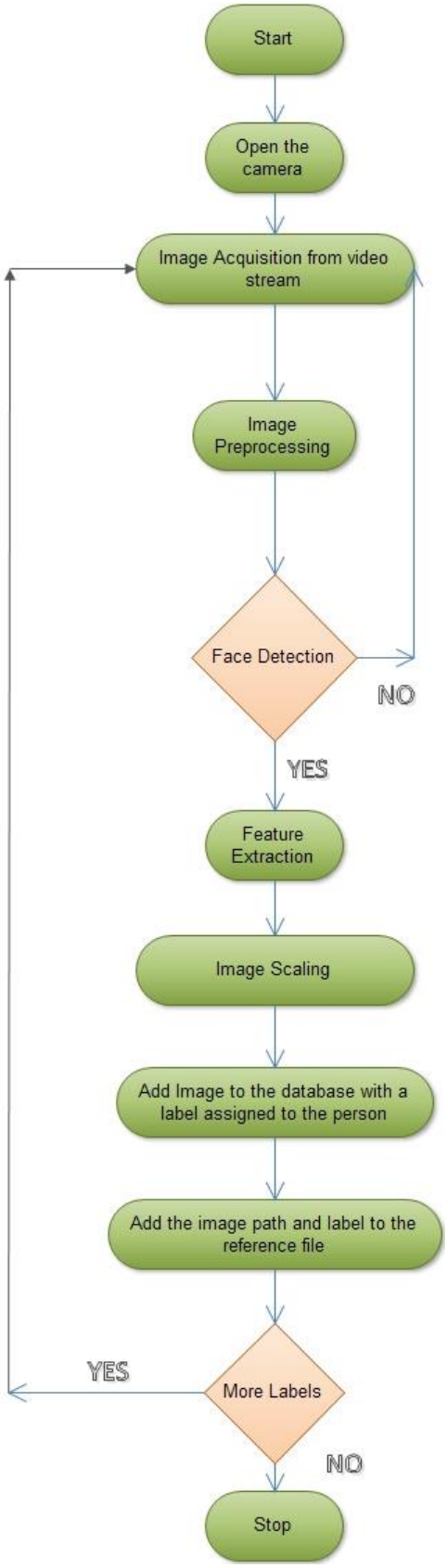
4. UML Diagrams

4.1 Activity Diagram

1. Face Detection and Recognition Control Flow

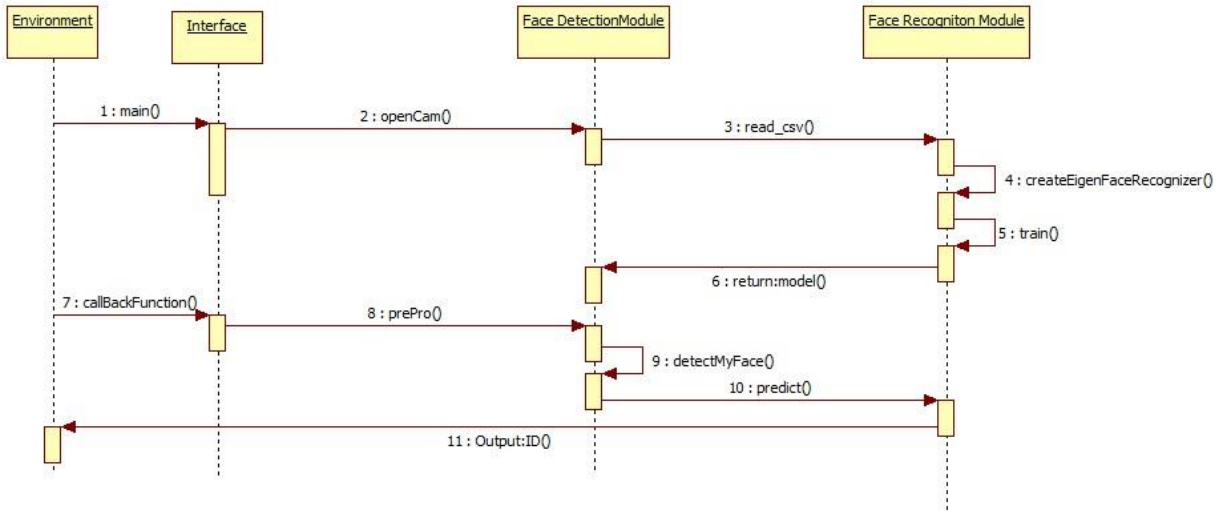


2. Database maintenance activity

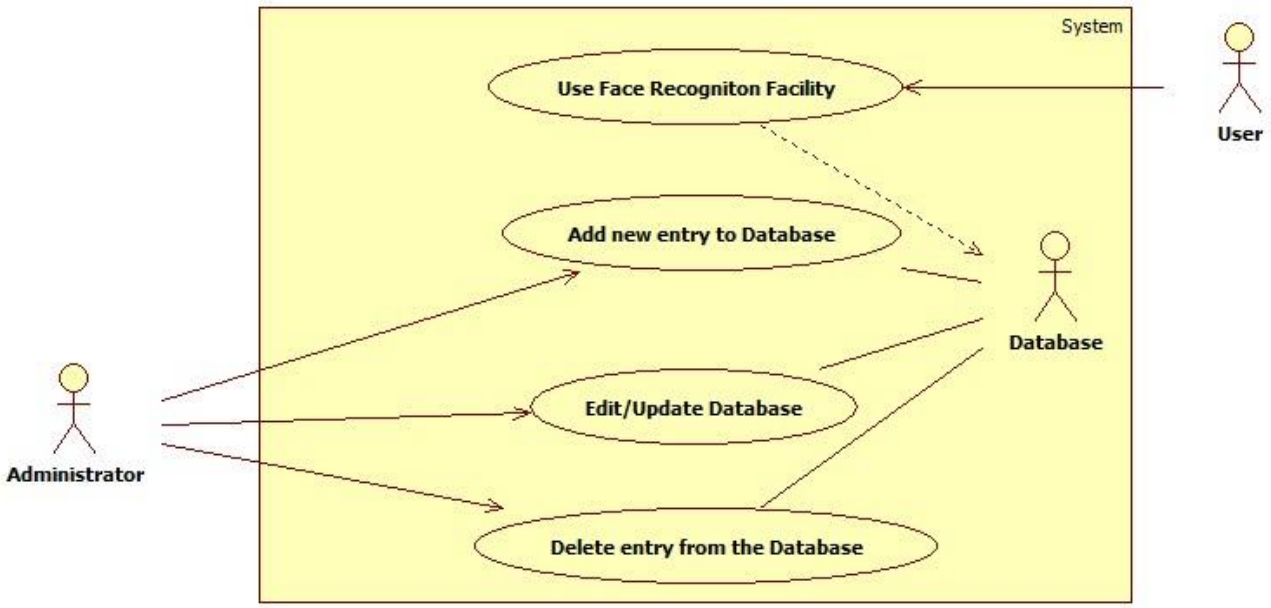


4.2 Sequence Diagram

1. Face Detection and Recognition methods call structure



4.3 Use Case Diagram



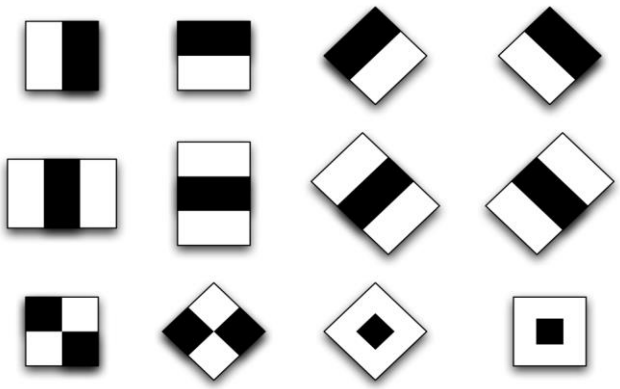
3. Appendix

a. Face Detection Technique

Viola Jones Face Detection Algorithm using Haar like Features

Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector.

In the detection phase of the Viola–Jones object detection framework, a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because such a Haar-like feature is only a weak learner or classifier (its detection quality is slightly better than random guessing) a large number of Haar-like features are necessary to describe an object with sufficient accuracy. In the Viola–Jones object detection framework, the Haar-like features are therefore organized in something called a *classifier cascade* to form a strong learner or classifier. The key advantage of a Haar-like feature over most other features is its calculation speed. Due to the use of *integral images*, a Haar-like feature of any size can be calculated in constant time



b. Face Recognition Techniques

Eigenfaces

Face recognition based on the geometric features of a face is probably the most intuitive approach to face recognition. One of the first automated face recognition systems was described in marker points (position of eyes, ears, nose) were used to build a feature vector (distance between the points, angle between them). The recognition was performed by calculating the euclidean distance between feature vectors of a probe and reference image. Such a method is robust against changes in illumination by its nature, but has a huge drawback: the accurate registration of the marker points is complicated, even with state of the art algorithms. A 22-dimensional feature vector was used and experiments on large datasets have shown, that geometrical features alone my not carry enough information for face recognition.

The Eigenfaces method described in took a holistic approach to face recognition: A facial image is a point from a high-dimensional image space and a lower-dimensional representation is found, where classification becomes easy. The lower-dimensional subspace is found with Principal Component Analysis, which identifies the axes with maximum variance. While this kind of transformation is optimal from a reconstruction standpoint, it doesn't take any class labels into account. Imagine a situation where the variance is generated from external sources, let it be light. The axes with maximum variance do not necessarily contain any discriminative information at all, hence a classification becomes impossible. The basic idea is to minimize the variance within a class, while maximizing the variance between the classes at the same time.

Recently various methods for a local feature extraction emerged. To avoid the high-dimensionality of the input data only local regions of an image are described, the extracted features are (hopefully) more robust against partial occlusion, illumination and small sample size. Algorithms used for a local feature extraction are Gabor Wavelets, Discrete Cosines Transform and Local Binary Patterns. It's still an open research question what's the best way to preserve spatial information when applying a local feature extraction, because spatial information is potentially useful information

II. Software Design Specification

1. Introduction

1.1 Purpose: The document provides a brief description of the software design, its implementation details and intended audience. It also includes a brief introduction to the algorithms, data structures that would be used for face detection and recognition.

1.2 Scope: Human face recognition is a difficult problem in computer vision. Face recognition is challenging because it is a real world problem. The human face is a complex, natural object that tends not to have easily (automatically) identified edges and features. Because of this, it is difficult to develop a mathematical model of the face that can be used as prior knowledge when analyzing a particular image. Face recognition as a security measure has the advantage that it can be done quickly, perhaps even in real time, and does not require extensive equipment to implement. It also does not pose a particular inconvenience to the subject being identified, as is the case in retinal scans.

1.3 Definitions, acronyms, and abbreviations

- CV – Computer Vision
- DIP –Digital Image Processing
- ANN – Artificial Neural Network
- SRS-Software Requirement Specification
- PCA-Principal Component Analysis

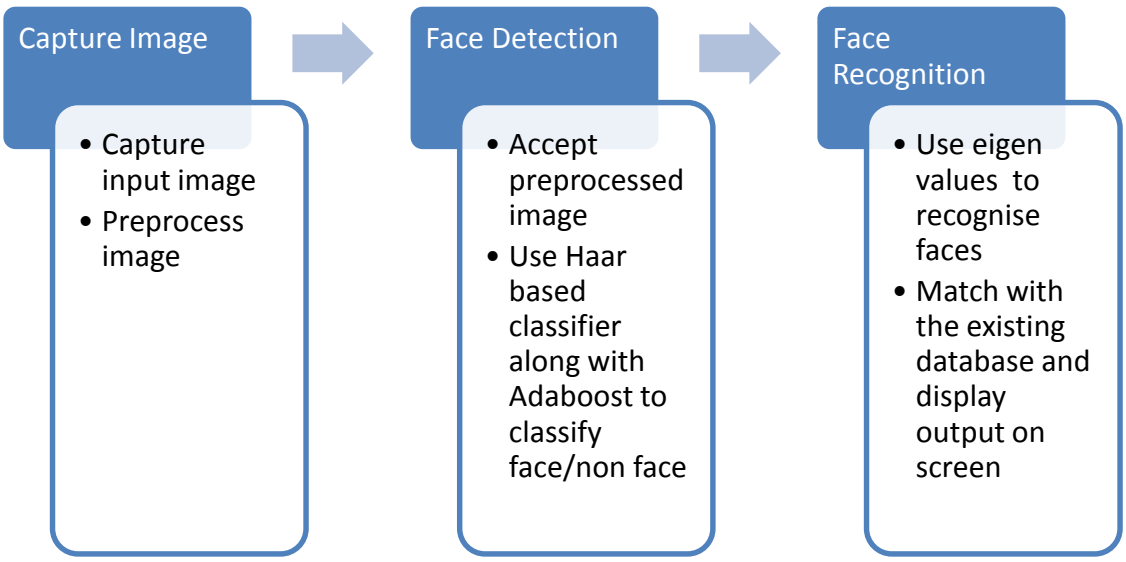
1.4 References

- IEEE-SDS www.ieee.org
- http://www.face-rec.org/
- http://opencv.org/
- http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

5.5 Overview: The rest of the document is divided into four sections. The System Overview highlights the Give a general description of the functionality, context and design of your project. The next section System Architecture describes in detail the architectural design, design rationale behind the architecture. This is then followed by Data Design and Component Design which describes each component such as algorithms, pseudo code.

2. SYSTEM ARCHITECTURE

2.1 Architectural Design



2.2 Decomposition Description

2.2.1 Capture Image

Input – Image from webcam.

Process- Face pre processing aims to reduce problems, such as by making sure the face always appears to have similar brightness and contrast, and perhaps makes sure the features of the face will always be in the same position (such as aligning the eyes and/or nose to certain positions

Output- Pre processed image.

2.2.2 Face Detection (Module 1-Implemented)

Input – Pre processed image, Haar based classifier and training samples.

Process-The image is loaded into the Haar Face Detection classifier that has been trained initially. The classifier detects all faces/ non faces in the image. The basic idea of the Haar-based face detector is that if you look at most frontal faces, the region with the eyes should be darker than the forehead and cheeks, and the region with the mouth should be darker than cheeks, and so on. It typically performs about 20 stages of comparisons like this to decide if it is a face or not, but it must do this at each possible position in the image and for each possible size of the face, so in fact it often does thousands of checks per image.

Output – Image with rectangle drawn around the faces recognised.

2.2.3 Face Recognition (Module 2-Implemented)

Input-Face detected image, Feature description from the database, Eigen Value based recogniser.

Process- The face recognition algorithm will then learn how to distinguish between the faces of the different people. This is referred to as the training phase and the collected faces are referred to as the training set. After the face recognition algorithm has finished training, you could then save the generated knowledge to a file or memory and later use it to recognize which person is seen in front of the camera. This is referred to as the testing phase.

Algorithm Used- Eigenfaces, also referred to as PCA. In simple terms, the basic principle of Eigenfaces is that it will calculate a set of special images (eigenfaces) and blending ratios (eigenvalues), which when combined in different ways can generate each of the images in the training set but can also be used to differentiate the many face images in the training set from each other.

Output- Recognised image along with the error handling mechanism.

3. Component Design

In this section, we take a closer look at what each component does in a more systematic and provide a pseudo code for the various algorithms used in the project.

- **Viola Jones Face Detection Algorithm-**

```
for number of scales in image pyramid do
    Down sample image by one scale
    Compute integral image for current scale
    for each shift step of the sliding detection window do
        for each stage in the cascade classifier do
            for each filter in the stage do
                Filter the detection window
            end
            Accumulate filter outputs within this stage
            if accumulation fails to pass per-stage threshold do
            Break the for loop and reject this window as a face
            end
        end
        if this detection window passes all per-stage threshold do
            Accept this window as a face
        else
            Reject this window as a face
        end
    end
end
```

- **Eigenface Algorithm**

The eigenfaces approach for face recognition involves the following initialization operations:

1. Acquire a set of training images.
2. Calculate the eigenfaces from the training set, keeping only the best *M* images with the highest eigenvalues. These *M* images define the "face space". As new faces are experienced, the eigenfaces can be updated.
3. Calculate the corresponding distribution in *M*-dimensional weight space for each known individual (training image), by projecting their face images onto the face space.

Having initialized the system, the following steps are used to recognize new face images:

1. Given an image to be recognized, calculate a set of weights of the *M* eigenfaces by projecting it onto each of the eigenfaces.
2. Determine if the image is a face at all by checking to see if the image is sufficiently close to the face space.
3. If it is a face, classify the weight pattern as either a known person or as unknown.
4. (Optional) Update the eigenfaces and/or weight patterns.
5. (Optional) Calculate the characteristic weight pattern of the new face image, and incorporate into the known faces.

III. Technologies Used

This section describes the technologies used for coding along with various features of coding language used.

The project will be implemented in C++ with the help of OpenCV (Open Source Computer Vision)**. OpenCV** (Open Source Computer Vision) is a library of programming functions for real time computer vision.

Features of OpenCV

- **Speed:** OpenCV, is basically a library of functions written in C/C++. You are closer to directly provide machine language code to the computer to get executed. As a result of this, programs written in OpenCV run much faster than similar programs written in Matlab. OpenCV is very fast when it comes to speed of execution. For example, we might write a small program to detect people's smiles in a sequence of video frames. In MATLAB, we would typically get 3-4 frames analysed per second. In OpenCV, we would get at least 30 frames per second, resulting in real-time detection.

- **Resources needed:** Typical OpenCV programs only require ~70mb of RAM to run in real-time. The difference as you can easily see is **HUGE!**

- **Cost:** List price for the base (no toolboxes) MATLAB (commercial, single user License) is around USD 2150. OpenCV (BSD license) is ***free!***

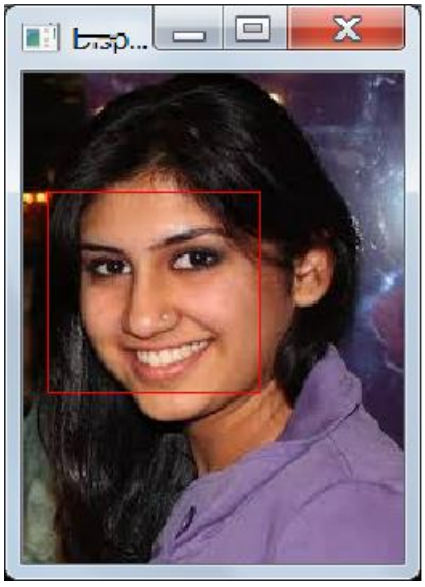
- **Portability:** MATLAB and OpenCV run equally well on Windows, Linux and Mac OS. However, when it comes to OpenCV, any device that can run C, can, in all probability, run OpenCV.

IV. Testing

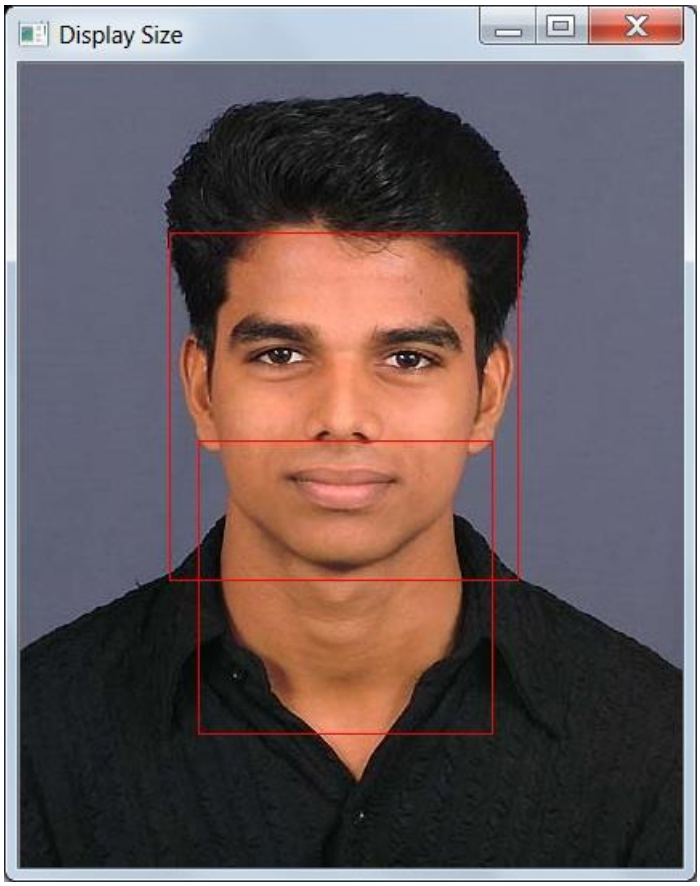
1. Phase I: Face Detection Test Cases

1.

2.



3.



4.

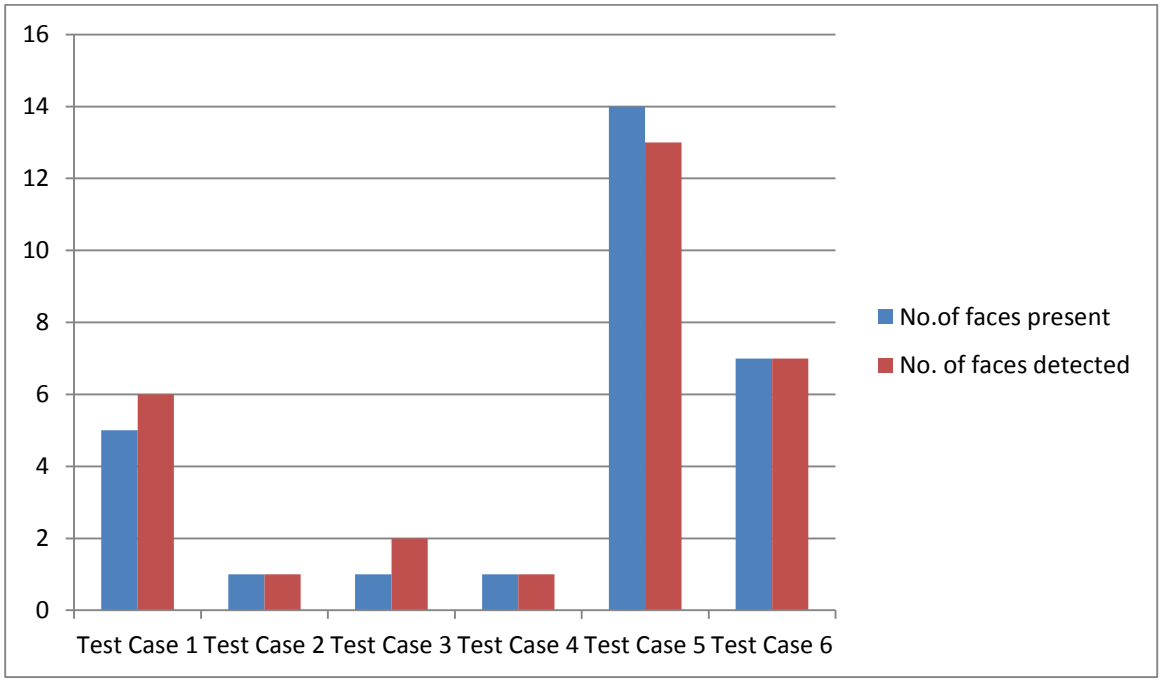
5.

6.

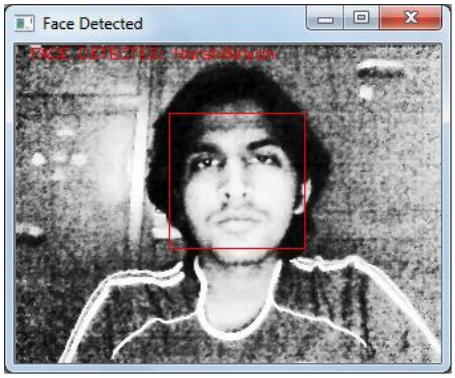
Efficiency of the Face Detection Module

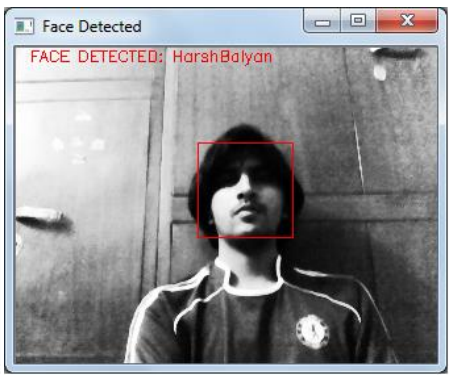
We have implemented the face detection module of the project which involved the use of Viola Jones algorithm for face detection.

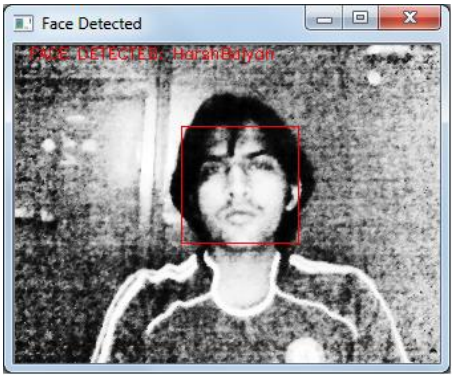
The below mentioned chart depicts the efficiency of the face detection module.

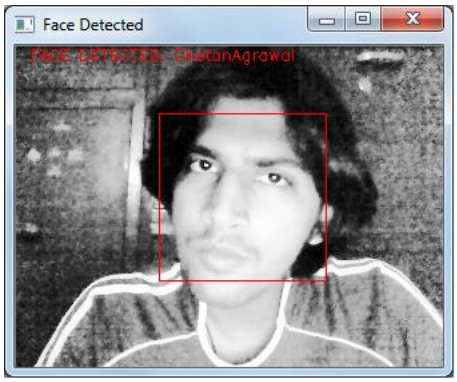


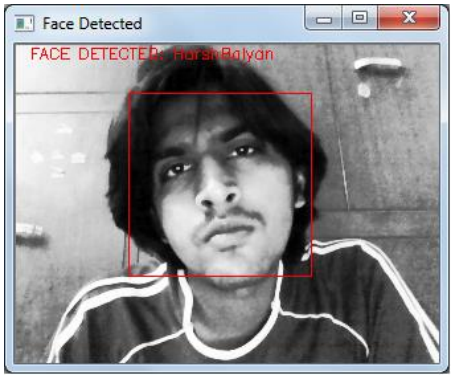
2. Phase II: Face Recognition Test Cases



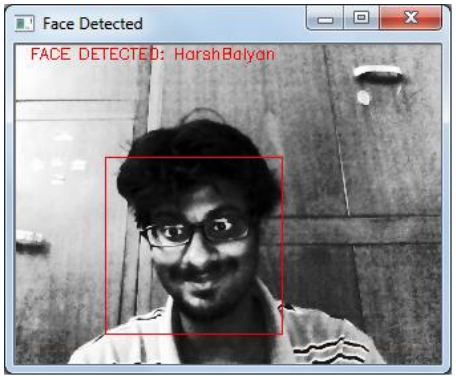


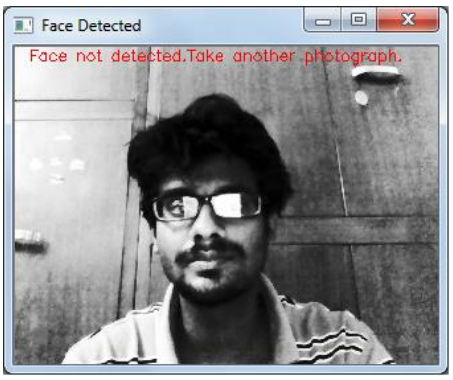


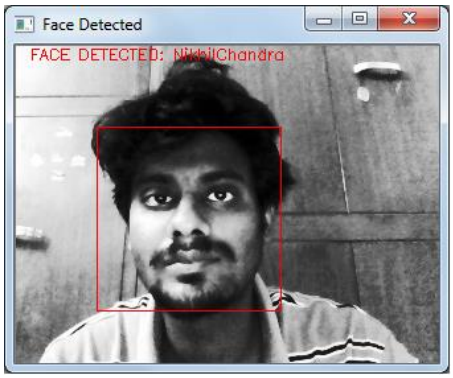


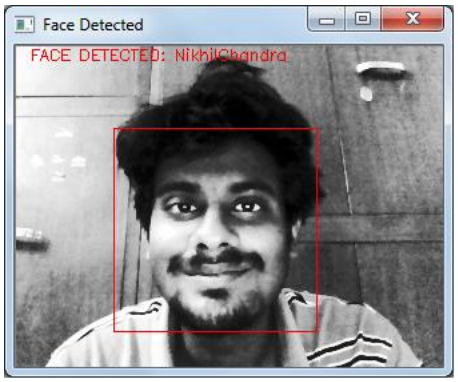


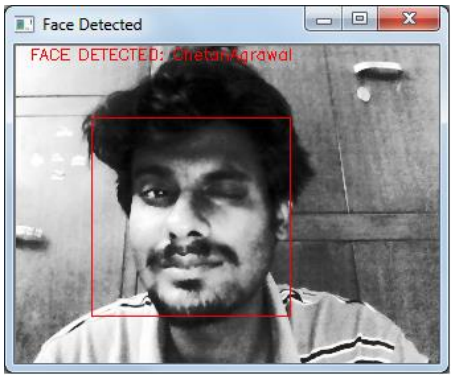

Face Detected
FACE DETECTED: NikhilChandra


Face Detected
FACE DETECTED: HarshBalyan

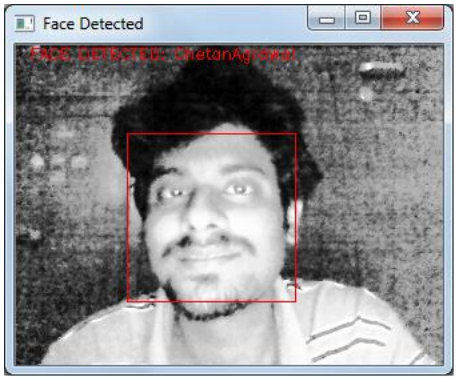

Face Detected
Face not detected.Take another photograph.

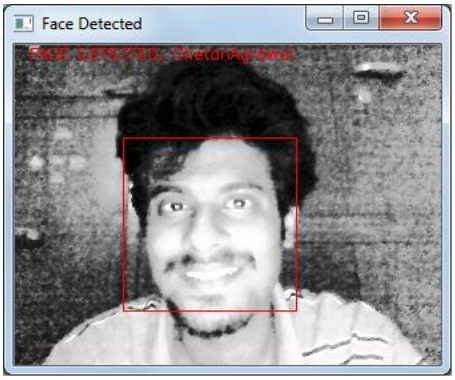

Face Detected
FACE DETECTED: NikhilChandra


Face Detected
FACE DETECTED: NikhilChandra

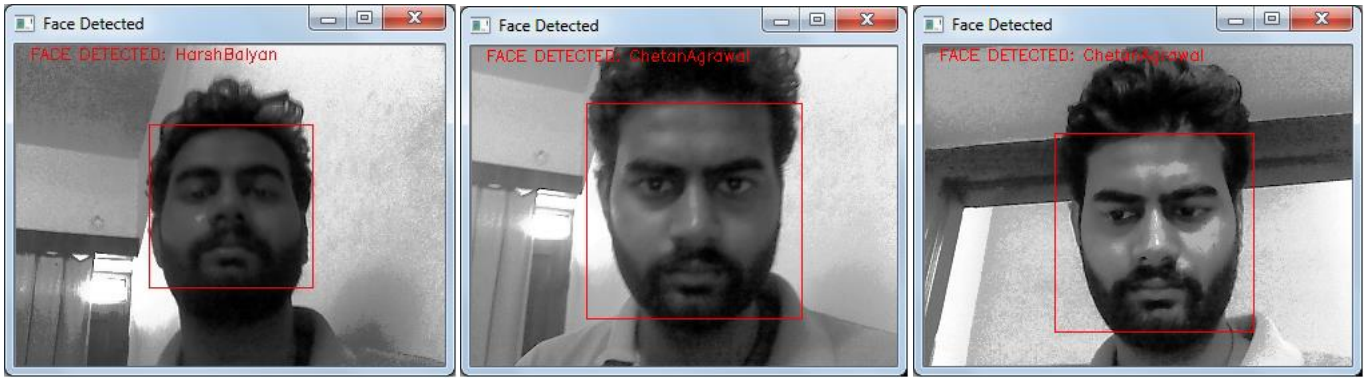

Face Detected
FACE DETECTED: ChetanAgrawal

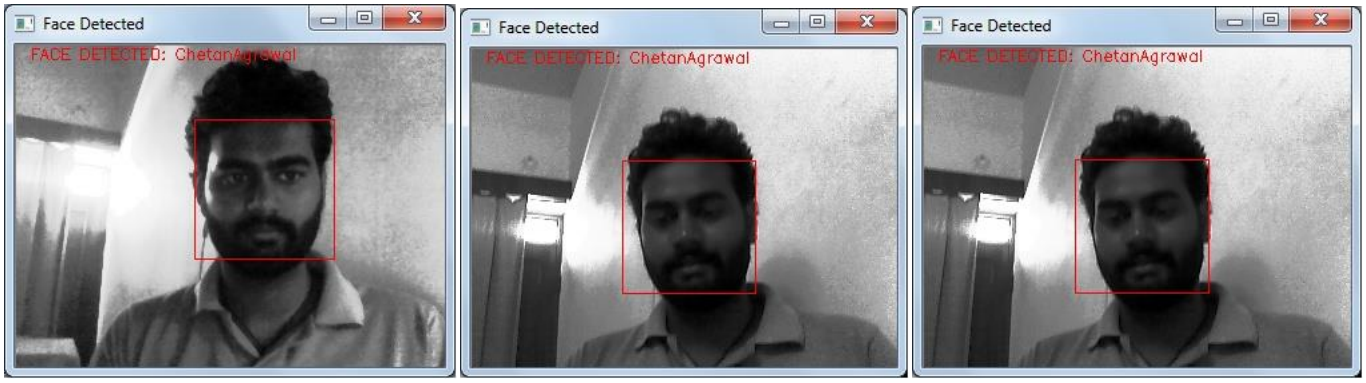

Face Detected
FACE DETECTED: NikhilChandra

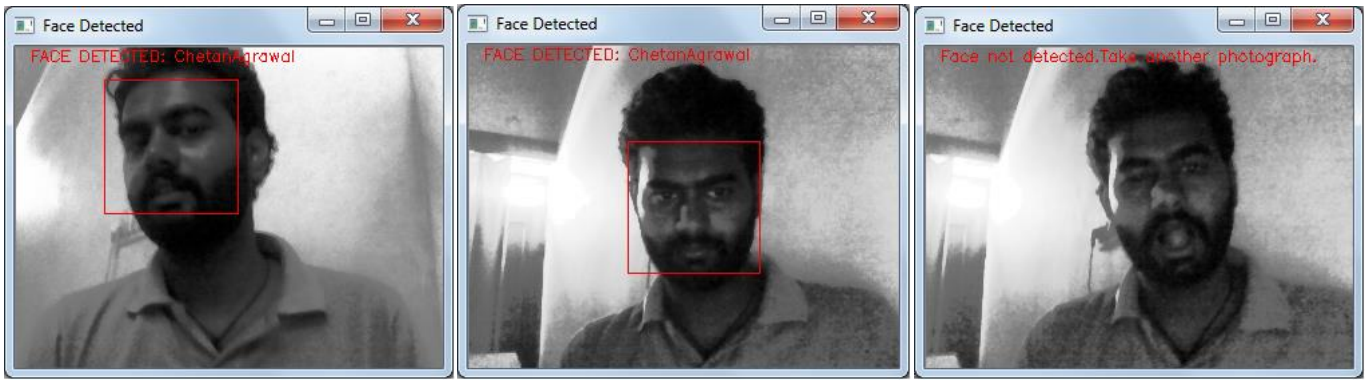

Face Detected


Face Detected


Face Detected
FACE DETECTED: NikhilChandra


Face Detected
FACE DETECTED: HarshBalyan
Face Detected
FACE DETECTED: ChetanAgrawal
Face Detected
FACE DETECTED: ChetanAgrawal


Face Detected
FACE DETECTED: ChetanAgrawal
Face Detected
FACE DETECTED: ChetanAgrawal
Face Detected
FACE DETECTED: ChetanAgrawal

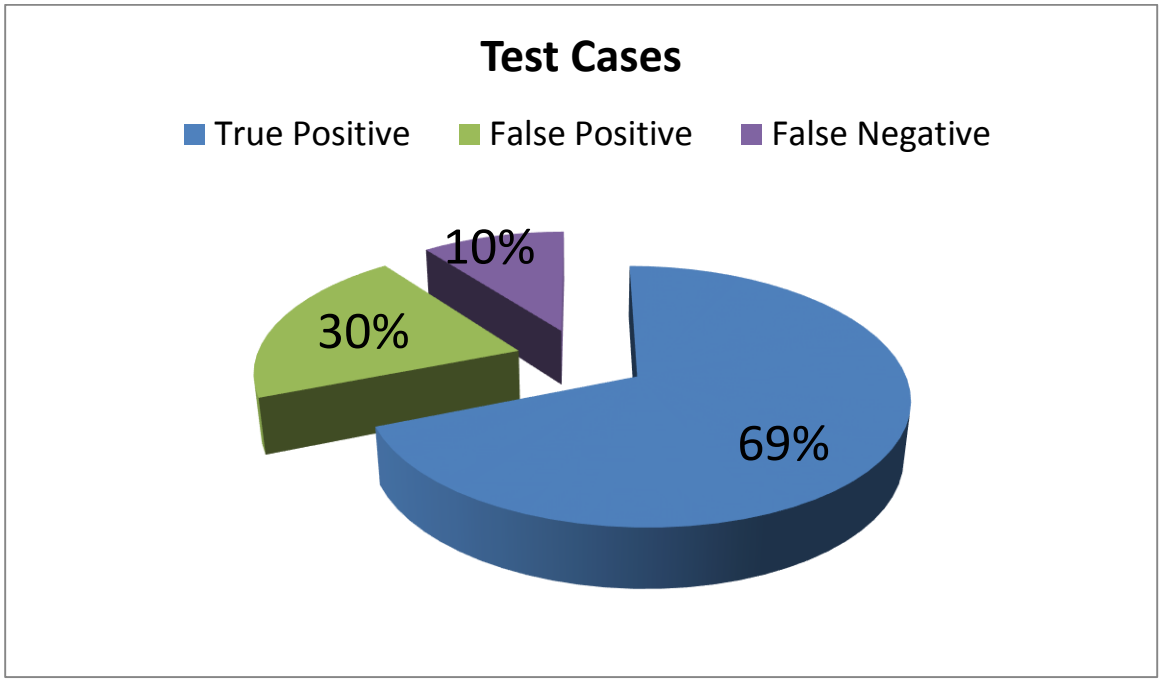

Face Detected
FACE DETECTED: ChetanAgrawal
Face Detected
FACE DETECTED: ChetanAgrawal
Face Detected
Face not detected.Take another photograph.


Face Detected
FACE DETECTED: ChetanAgrawal

Efficiency of the Face Recognition Module

We have implemented the face detection module of the project which involved the use of Viola Jones algorithm for face detection.

The below mentioned chart depicts the efficiency of the face detection module.



V. Results and Discussion

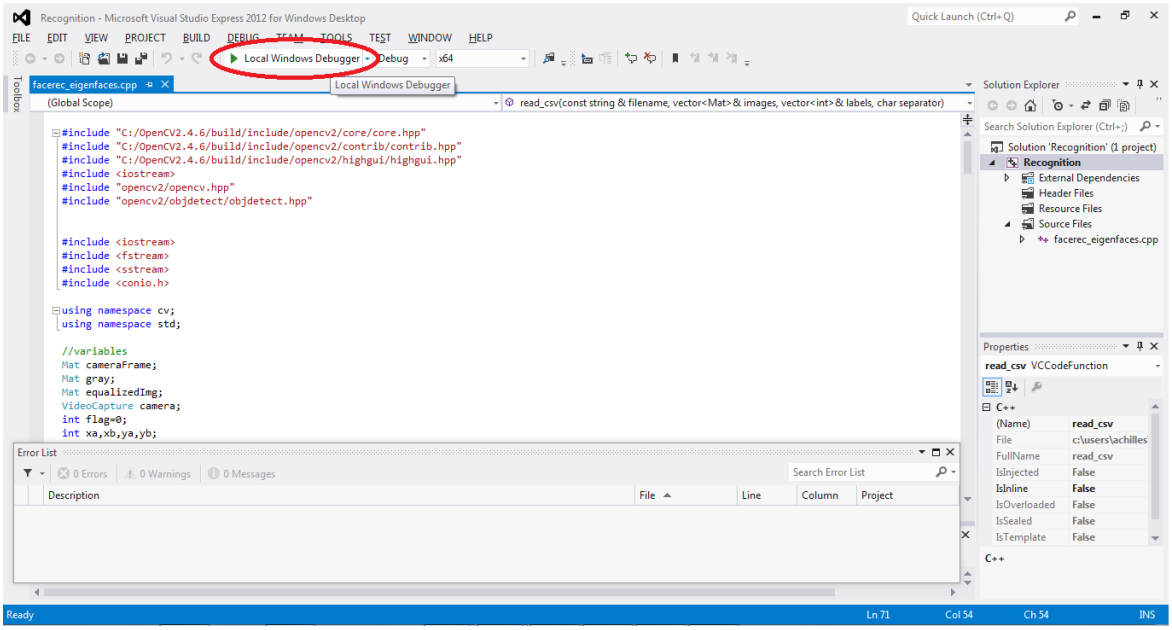
Scope and Limitations

- This project can be extended to be a part of facial recognition based attendance system, album summarization.
- Learning time is proportional to the database size. For large databases back-end must be implemented by an efficient DBMS that supports indexing.
- Accuracy directly depends on number of training samples per person.
- Haar classifier is used for the Frontal Face Detection; hence any other view of the face may not be detected.
- In case of change in the facial appearance of a person, suitable training sets must be added to the database.

RESULTS

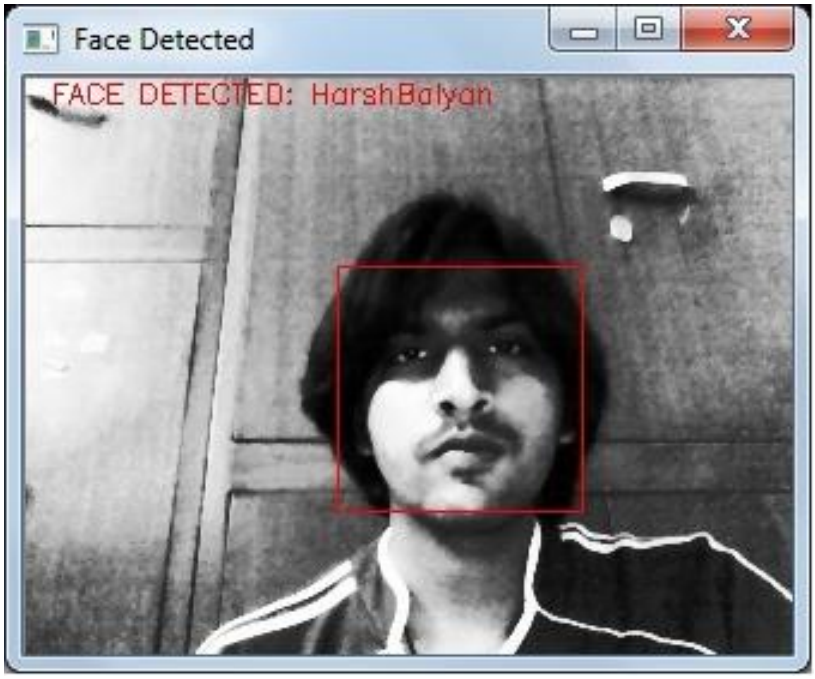
- **Based on the set of test cases provided to the system, the system has an accuracy of 69 %.**
  (Training Set includes minimum 12 images per person)

VI. Project Snapshots



1. Click on the 'Local Windows Debugger'


Face Detected
Face not detected.Take another photograph.


Face Detected
FACE DETECTED: HarshBalyan

Name	Nikhil Chandra																			
Roll No.	BE/4027/10																			
Role	Coder	**JULY – 13**			**AUGUST – 13**			**SEPTEMBER – 13**			**OCTOBER – 13**			**NOVEMBER – 13**			**DECEMBER – 13**			
Phase	**Item**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**Total Effort**
Requirement Analysis	Problem Overview	3	5																	8
	Others	1	1																	2
Design	Flow Chart/Structure Chart				4															4
	Pseudo Code				2	2		2												6
	Report Layout										2									2
	Use Case Diagram											2								2
	Sequence Diagram												2							2
Coding and Unit Testing	Coding												2		2		4	1		9
	Unit Test Plan													2			2			4
System Testing	System Test Plan															2			2	3
		JANUARY – 14(Effort in Hours)																		
Role	Analyst	**WEEK-1**						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding	Image Pre-processing		3	2	3		1		2		1		4		1	4		1	1	22
	Face Detection									2		1		2	3		2			10
		FEBRUARY- 14(Effort in Hours)																		
	Testing and Debugging																			
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding and Unit Testing	Model designing	2	1	3	2	2	1	3	1	2	2	1			2					23
	Face Recognition												2	2	1	1	2	2	1	11
		MARCH – 14(Effort in Hours)																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding and Unit Testing	Testing and Debugging		1		2			2			1			1			1			8
		APRIL – 14(Effort in Hours)																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
System Testing	Report generation	1			1	2	1		1		2		1	2		2				13

Name	Harsh Balyan																			
Roll No.	BE/4050/10																			
Role	Coder	**JULY – 13**			**AUGUST – 13**			**SEPTEMBER – 13**			**OCTOBER – 13**			**NOVEMBER – 13**			**DECEMBER – 13**			
Phase	**Item**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**Total Effort**
Requirement Analysis	Problem Overview	3	5																	8
	Others	1	1																	2
Design	Flow Chart/Structure Chart				4															4
	Pseudo Code				2	2		2												6
	Report Layout										2									2
	Use Case Diagram											2								2
	Sequence Diagram												2							2
Coding and Unit Testing	Coding												2		2		4	1		9
	Unit Test Plan													2			2			4
System Testing	System Test Plan															2			2	3
		JANUARY – 14(Effort in Hours)																		
Role	Analyst	**WEEK-1**						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding	Image Pre-processing		3	2	3		1		2		1		4		1	4		1	1	22
	Face Detection									2		1		2	3		2			10
	Testing and Debugging	**FEBRUARY- 14(Effort in Hours)**																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding and Unit Testing	Model designing	2	1	3	2	2	1	3	1	2	2	1			2					23
	Face Recognition												2	2	1	1	2	2	1	11
		MARCH – 14(Effort in Hours)																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding and Unit Testing	Testing and Debugging		1		2			2			1			1			1			8
		APRIL – 14(Effort in Hours)																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
System Testing	Report generation	1			1	2	1		1		2		1	2		2				13

Name	Chetan Agrawal																				
Roll No.	BE/4013/10																				
Role	Coder	**JULY – 13**			**AUGUST – 13**			**SEPTEMBER – 13**			**OCTOBER – 13**			**NOVEMBER – 13**			**DECEMBER – 13**			
Phase	**Item**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**WEEK-1**	**WEEK-2**	**WEEK-3**	**Total Effort**
Requirement Analysis	Problem Overview	3	5																	8
	Others	1	1																	2
Design	Flow Chart/Structure Chart				4															4
	Pseudo Code				2	2		2												6
	Report Layout										2									2
	Use Case Diagram											2								2
	Sequence Diagram												2							2
Coding and Unit Testing	Coding												2		2		4	1		9
	Unit Test Plan													2			2			4
System Testing	System Test Plan															2			2	3
		JANUARY – 14(Effort in Hours)																		
Role	Analyst	**WEEK-1**						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding	Image Pre-processing		3	2	3		1		2		1		4		1	4		1	1	22
	Face detection									2		1		2	3		2			10
	Testing and Debugging	**FEBRUARY- 14(Effort in Hours)**																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding and Unit Testing	Model Designing	2	1	3	2	2	1	3	1	2	2	1			2					23
	Face Recognition												2	2	1	1	2	2	1	11
		MARCH – 14(Effort in Hours)																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
Coding and Unit Testing	Testing and Debugging		1		2			2			1			1			1			8
		APRIL – 14(Effort in Hours)																		
		WEEK-1						**WEEK-2**						**WEEK-3**						
Phase	**Item**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**	**Total Effort**
System Testing	Report generation	1			1	2	1		1		2		1	2		2				13