



Dissertation on
“Automated Attendance System”

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Karthik R	01FB15ECS142
Krishnakumar Hegde	01FB15ECS153
Nikhil Y Dixit	01FB15ECS190

Under the guidance of

Internal Guide

Prof Dinesh Singh

Associate Professor

Computer Science Department

PES University

January – May 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

‘Automated Attendance System’

is a bonafide work carried out by

**Karthik R
Krishnakumar Hegde
Nikhil Y Dixit**

**01FB15ECS142
01FB15ECS153
01FB15ECS190**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2019 – May. 2019. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature
Prof Dinesh Singh
Associate Professor

Signature
Dr. Shylaja S S
Chairperson

Signature
Dr. B K Keshavan
Dean of Faculty

External Viva

Name of the Examiners

Signature with Date

1. _____

2. _____

DECLARATION

We hereby declare that the project entitled “**Automated Attendance System**” has been carried out by us under the guidance of Prof. Dinesh Singh, Associate Professor and submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2019. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

01FB15ECS142 Karthik R

01FB15ECS153 Krishnakumar Hegde

01FB15ECS190 Nikhil Y Dixit

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof. Dinesh Singh, Associate Professor for providing us an opportunity to work under him. His suggestions have played a vital role in developing our project.

We would like to thank the college management and express our sincere gratitude to Prof. Preet Kanwal, Prof. Sangeeta V I the project coordinators who have been constantly involved in coordinating our project reviews.

We would not forget to remember Dr. Shylaja S S, Chairperson, Dept. of Computer Science and Engineering, PES University for her encouragement and valuable guidance.

We would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help

We owe a big thanks to the Vice-Chancellor Dr. KNB Murthy for encouraging students to build quality projects and also extend help whenever required.

Prof. D Jawahar, Pro-Chancellor, PES University has played a crucial role in motivating us and reminding us of our duties regularly.

We would also like to extend our thanks to Dr. M R Doreswamy, Chancellor, PES Institutions.

Last but not the least; the mini-project would not have been successful without the help and guidance from various faculty members and peers.

ABSTRACT

Face recognition is an important application of Image processing owing to its use in many fields. Identification of individuals in an organization for the purpose of attendance is one such application of face recognition. Maintenance and monitoring of attendance records plays a vital role in the analysis of performance of any organization. The purpose of developing attendance management system is to computerize the traditional way of taking attendance. Automated Attendance Management System performs the daily activities of attendance marking and analysis with reduced human intervention.

The project will show how we can implement algorithms for face detection and recognition in image processing to build a system that will detect and recognise faces of students in a classroom. The automatic attendance management will replace the manual method, which takes a lot of time and difficult to maintain. There are many biometric processes, but among those face recognition is the best method. Uniqueness or individuality of an individual is his face. In this project face of an individual is used for the purpose of attendance making automatically. Attendance of the student is very important for every college, universities and school. Conventional methodology for taking attendance is by calling the name or roll number of the student and the attendance is recorded. Time consumption for this purpose is an important point of concern. To stay away from these losses, an automatic process is used in this project which is based on image processing. In this method the camera is fixed in the classroom and it will capture the image. These images are then used for facial algorithms. In this project face detection and face recognition is used. Face detection is used to locate the position of face region and face recognition is used for marking the understudy's attendance. The database of all the students in the class is stored and when the face of the individual student matches with one of the faces stored in the database then the attendance is recorded.

In recent years, research has been carried out and face recognition and detection systems have been developed. Some of which are used on social media platforms, banking apps, government offices e.g. the Metropolitan Police, Facebook etc.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	01
2.	PROBLEM DEFINITION	03
3.	LITERATURE SURVEY	04
	3.1 Face Detection and Recognition using LBPH	04
	3.1.1 Introduction	04
	3.1.2 Approach	05
	3.1.3 Results and Conclusion	05
	3.2 Machine learning is fun	06
	3.2.1 Introduction	07
	3.2.2 Approach	07
4.	PROJECT REQUIREMENTS SPECIFICATION	08
	4.1 Design Constraints	08
	4.2 Product Perspective	08
	4.3 User Characteristics	09
	4.4 Assumptions, Dependencies, General Constraints	09
	4.5 Risks	09
5.	SYSTEM REQUIREMENTS SPECIFICATION	10
	5.1 Functional Requirement	11
	5.2 Non Functional Requirement	12
	5.2.1 Hardware Requirement	12
	5.2.2 Software Requirement	12
	5.2.3 User Interfaces	13
	5.2.4 Communication Interface	13
	5.2.5 Site adoption Requirement	13
	5.2.6 Safety Requirement	13
6.	SYSTEM DESIGN	14
	6.1 Dataflow Model	16
	6.2 Structural Model	18
7.	DETAILED DESIGN	20
	7.1 Masterclass Diagram	20
	7.1.1 Module 1 Lbph main class	20
	7.1.2 Module 2 Lbp class	21
	7.1.3 Module 3 Histogram class	21
	7.2 Designing algorithm	22
8.	IMPLEMENTATION AND PSEUDOCODE	28

9.	TESTING	39
10.	RESULTS AND DISCUSSION	40
11.	SNAPSHOTS	42
12.	CONCLUSIONS	44
13.	FURTHER ENHANCEMENTS	45
	REFERENCES/BIBLIOGRAPHY	46

LIST OF FIGURES

Figure No.	Title	Page No.
3.1	Literature survey	6
5.1	User Requirements	11
6.1	functional flow design	14
6.2	Flow design	15
6.3	Flow design flowchart	16
6.4	Detailed Flow design	17
6.5	Structural design	18
7.1	master class diagram	19
7.2	Lbph main class diagram	20
7.3	MathLib class diagram	20
7.4	Histogram class diagram	21
7.5	Algorithm Design	22
7.6	Binary to Decimal Image	24
8.1	LBPH processing	31
8.2	Histogram transformation	32

CHAPTER-1

INTRODUCTION

In Face Detection and Recognition systems, the flow process starts by being able to detect and recognise faces from an input device i.e. mobile phone. In today's world, it has been proven that students engage better during lectures only when there is effective classroom control. The need for high level student engagement is very important. Students need to be continuously engaged during lectures and one of the ways is to recognise and address them by their names. Therefore, a system like this will improve classroom control. In our own view based on our experience, during our time, we realised calling any student by his/her name gives me more control of the classroom and this draws the attention of the other students in the classroom to engage during lectures. Face detection and recognition is not new in our society we live in. The capacity of the human mind to recognize particular individuals is remarkable. It is amazing how the human mind can still persist in identification of certain individuals even through the passage of time, despite slight changes in appearance. Attendance is prime important for both the teacher and student of an educational organization. So it is very important to keep record of the attendance. The problem arises when we think about the traditional process of taking attendance in class room. Calling name or roll number of the student for attendance is not only a problem of time consumption but also it needs energy. So an automatic attendance system can solve all above problems.

There are some automatic attendances making system which are currently used by much institution. One of such system is biometric technique. Although it is automatic and a step ahead of traditional method it fails to meet the time constraint. The student has to wait in queue for giving attendance, which is time taking.

This project introduces an involuntary attendance marking system, devoid of any kind of interference with the normal teaching procedure. The system can be also implemented during exam

sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions.

Face detection is defined as finding the position of the face of an individual. In other word it can be defined as locating the face region in an image. After detecting the face of human its facial features is extracted and has wide range of application like facial expression recognition, face recognition, observation systems, human PC interface and so forth...Detecting face in an image of single person is easy but when we consider a group image of an image containing multiple faces, the task becomes difficult.

For the application of face recognition, detection of face is very important and the first step. After detecting face the face recognition algorithm can only be functional. Face detection itself involves some complexities for example surroundings, postures, enlightenment etc.

CHAPTER – 2

PROBLEM DEFINITION

This project is being carried out due to the concerns that have been highlighted on the methods which lectures use to take attendance during lectures. The use of clickers, ID cards swiping and manually writing down names on a sheet of paper as a method to track student attendance has prompted this project to be carried out. This is not in any way to criticize the various methods used for student attendance, but to build a system that will detect the number of faces present in a classroom as well as recognizing them. Also, a teacher will be able to tell if a student was honest as these methods mentioned can be used by anyone for attendance records, but with the face detection and recognition system in place, it will be easy to tell if a student is actually present in the classroom or not. This system will not only improve classroom control during lectures, it will also possibly detect faces for student attendance purposes. we will use LBPH to build and implement this system

CHAPTER-3

LITERATURE SURVEY

3.1. Face Detection and Recognition using LBPH

3.1.1. Introduction

This paper focuses on face detection and recognition using the infamous LBPH algorithm. The authors have tried to explain in detail how the algorithm works and how can it be implemented. The authors have also focused on packaging this onto Raspberry Pi.

The authors discuss how previous research in this field has mainly focussed on the LBPH algorithm, which is arguably of high importance since not many applications make use of LBPH.

3.1.2. Approach

The paper has divided the process into face detection and face recognition. The paper suggests Haar like classifiers for face detection to reduce false positives. The dataset needs to be huge in order to correctly detect false faces.

For face recognition, LBPH algorithm is suggested. It is the best example of pixel wise classification which indirectly extracts feature by generating Histograms.

3.1.2. Results & Conclusion

As a baseline, the face recognition was done using LBPH. To reduce the false-positives drastically and increase the efficiency in this research, the authors have used haar like features and for recognition of face they have used LBPH (local binary pattern histogram). This reference design can be used for authentication in banks , and other public places. Thus for a safety purpose in real time the authors have successfully designed a face recognition system in minimum expenses using open cv and lbp algorithm.

3.2 Machine Learning is fun : Face detection using Deep Learning

3.2.1 Introduction

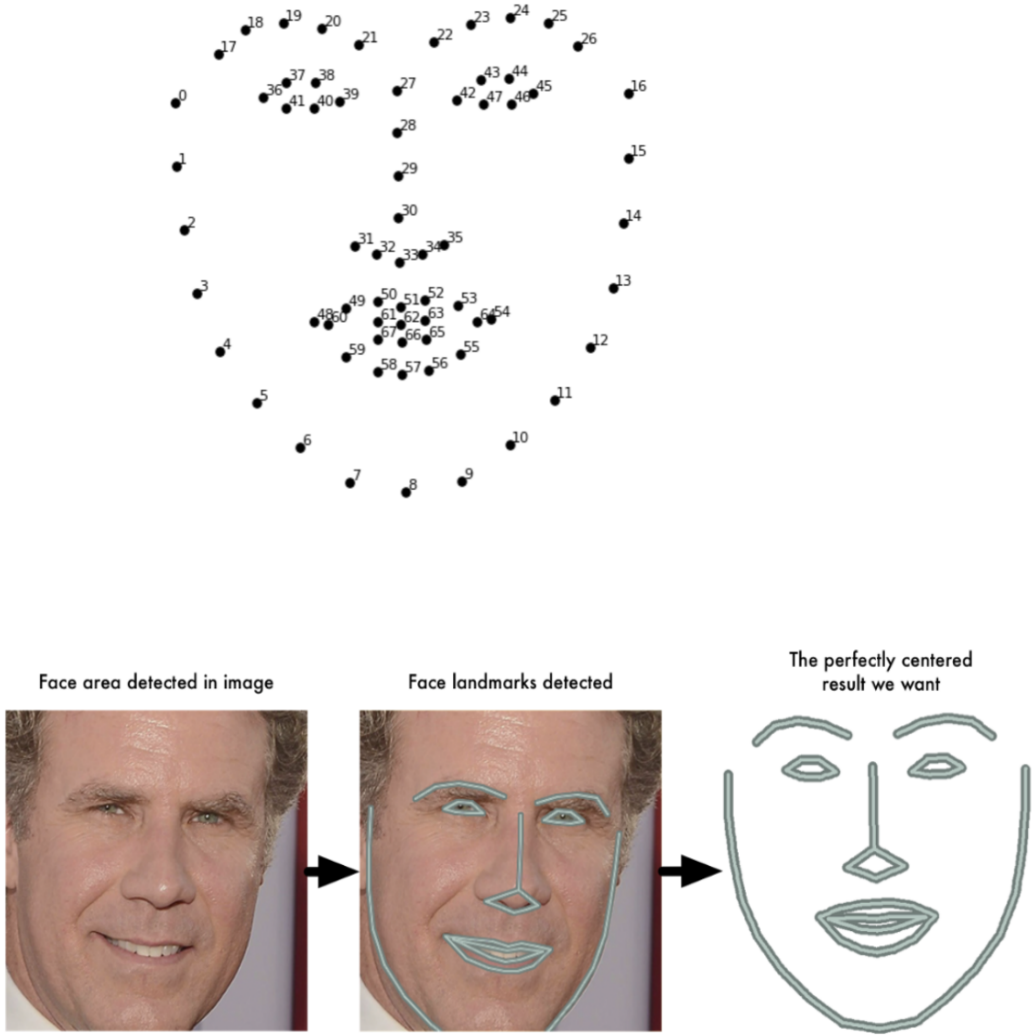
This paper discusses about how deep learning concepts can be used to detect faces in an image. The author solves this problem one step at a time. For each step, the author explains about a different machine learning algorithm. The author doesn't explain every single algorithm but emphasises the main ideas behind each one and explains how one can build an own facial recognition system in Python using OpenFace and dlib.

3.2.2 Approach

Author has made use of an algorithm called face landmark estimation. There are lots of ways to do face recognition, but the author has used the approach which was invented in 2014 by Vahid Kazemi and Josephine Sullivan.

The basic idea is to with 68 specific points (also called landmarks) that exist on every face—the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then training a machine learning algorithm to be able to find these 68 specific points on any face is done.

3.2.3. Results & Conclusion



Face area detected in image

Face landmarks detected

The perfectly centered result we want

The above generated sketch is used to form an actual image of the face by encoding and then comparing it with the original image.

Fig 3.1 Literature survey

3.3 A review paper on Face recognition techniques

3.3.1 Introduction

With data and information accumulating in abundance, there is a crucial need for high security. Face recognition has been a fast growing, challenging and interesting area in real time applications. A large number of recognition algorithms have been developed in last decades. In this paper an attempt is made to review a wide range of methods used for face recognition comprehensively. This include PCA, LDA, ICA and SVM and various hybrid combination of this techniques. This review investigates all these methods with parameters that challenges face recognition like illumination, pose variations, facial expressions.

3.3.2. Approach

This paper has discussed general face detection techniques which include PCA, LDA and SVM. The author has tried to explain in brief how these techniques can be used in different circumstances.

3.3.3 Results & Conclusion

This paper has attempted to review a significant number of papers to cover the recent development in the field of face recognition. Present study reveals that for enhanced face recognition new algorithm has to evolve using hybrid methods of soft computing tools such as ANN, SVM, SOM may yields better performance. We can use any of them as per our requirement and application. We can also work over to improve the efficiency of discussed algorithms and improve the performance.

CHAPTER – 4

PROJECT REQUIREMENTS SPECIFICATION

4.1 Design Constraints

This section of the document shall provide a general description of any other item that will limit the developer's option for designing the system. These can include the following:

- Zero power shutdown tolerance
- System capable of handling 5 requests
- Preferable to Linux os
- Clean visuals for the Camera
- Operations and updates should be atomic
- User access only after credential check
- Privileged access to lecturers to schedule special class/cancel class
- Student restricted only to view and submit request upon wrong marking

4.2 Product Perspective

This product can be made and run independently without any dependency , rather if a college or institution is using this product then would be ease and efficient if this product is given access to the database , hence would be no duplicate maintenance of data. The hardware requirements for development would be an interface with a application gui running. The attendance stats from the captured image would be sent to the main server to get updated after specific interval of time(after each class).

4.3 User Characteristics

The main end-user of this application would be Lecturer.

- end-user may not have the technological expertise that would enable him/her to use the application

- end-user need not know the entire algorithm to use the application.
- Lecturer interact with application through simple user interface that is provided to him/her.
- Lecturer might find this method more convenient and easier than traditional method.
- Lecturer need not carry the attendance record explicitly with him as application will keep that information.

4.4 General Constraints, Assumptions and Dependencies

This section of the CRS shall provide a general description of any other item that will limit the developer's option for designing the system. These can include the following:

- Zero power shut-down tolerance
- System capable of handling 5 requests
- Preferable to Linux os
- Clean visuals for the Camera
- Operations and updations should be atomic
- User access only after credential check
- Privileged access to lecturers to schedule special class/cancel class
- Student restricted only to view and submit request upon wrong marking

4.5 Risks

The major risk is to maintain high true-positive even a single false-positive could lead to improper attendance , hence should majorly focus on the accuracy rate of our project. Resource DB should be given utmost security as it contains details of each and every students with their related data .The access for the attendance portal should be secured with userid and pass-key to restrict anonymous user access.

CHAPTER – 5

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Functional Requirements

Functional requirements are features that the system will need in order to deliver or operate. In the case of this project, it was important to gather some requirements that will be needed to achieve the objectives set out previously. With client (user) story a use case analysis was implemented which resulted in the following functional and non-functional requirements were captured. The functional requirements have been gathered from the user story developed from the minutes collected during meetings with the client and are outlined here

- Capture face images via webcam or external USB camera.
- The faces must be detected in bounding boxes.
- Compute the total attendance based on detected faces.
- Crop the total number of faces detected.
- Resize the cropped faces to match faces the size required for recognition.
- Store the cropped faces to a folder.
- Load faces on database.
- Train faces for recognition. Perform recognition for faces stored on database.

5.1.1 Use Cases

Use Case Item	Description
Admin	Manages the portal
Lecturer	Manage and view attendance
Student	View attendance status

Table 5.1 Use case items and their description

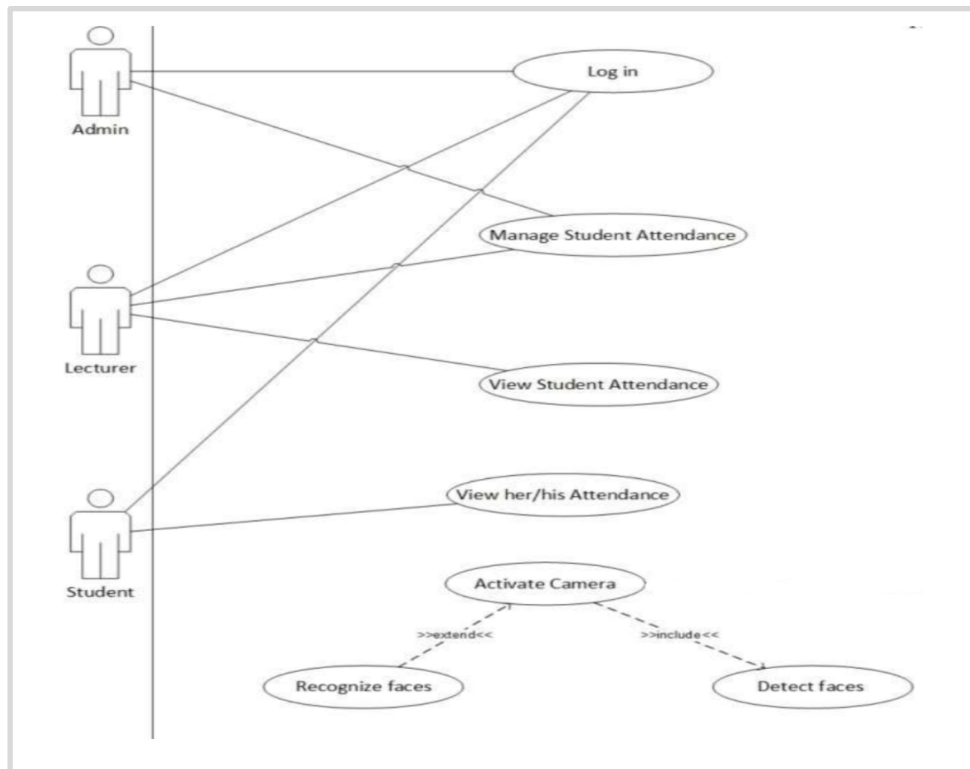


Fig. 5.1 User Requirements

5.2 Non-Functional Requirements

Non-functional requirements are set of requirements with specific criteria to judge the systems operation. These requirements have been collected based on the following after meetings with the client. They cover ease of use to the client, security, support availability, operational speed, and implementation considerations. More specifically: The user will find it very convenient to take photos.

- The user will inform the students when taking a photo with clear instructions on how to position their faces.
- The system is very secure.
- The system will have a response time of 30 seconds.
- The system can be easily installed. The system is 100% efficient.
- The system must be fast and reliable

5.2.1 Hardware Requirements

- To capture images proper camera with high resolution capacity is necessary.
- The device running the application should have a display, High end CPU, proper GUI, adequate RAM and storage.
- It should be capable of running applications that perform operations that involve running machine
- learning algorithm, performing facial recognition, camera activity, media activity, clock and web browsing.

5.2.2 Software Requirements

- Desktop computers and laptops should be capable of running a web browser.
- Java environment is required to run the algorithms.
- Different software packages and python-based modules for facial recognition feature which are helpful to run machine learning algorithms should be installed. Eg: histogram

5.2.3 Performance Requirements

- Storage to store the dataset.
- Recognition of multiple faces in captured images.
- Recognition of face from different angles.
- Storing the features of the datasets.
- Database or file to store attendance details.
- Mapping student name with feature captured.
- Should be compatible across all heterogeneous systems

5.2.4 User Interfaces

Web based user interface is provided where lecturer can check the presence of each student according to the day. According to class section and lecturer students name with value for

presence of the student will be there.

Log in functionality will be provided so that only lecturer can view the site and access the data to further requirements

The design is meant to be very user friendly and intuitive. Minimal button clicks and maximum information display are the current goals of the design. A simple yet immersive user experience is kept in mind while designing the application.

5.2.5 Communication Interfaces

Communication between camera and application happens over the local network and It is preferred to use the application over internet.

5.2.6 Site Adaptation Requirements

- Device should be capable of running machine learning algorithm to work with facial recognition algorithm
- The web browser used to access the application should be able to run html, php and Java script either natively or by support of an external module
- Necessary storage, internet, location and network permissions must be granted on all systems that the application is running on, failing which the application might run incorrectly or fail to run.

5.2.7 Safety Requirements

- One should run the algorithm only after making sure of hardware requirements.
- The login credentials to the application must be private at all costs.
- The user using the app must lecturer or recognized member of the organization.
- Any defect or lope holes must be taken care of.

CHAPTER – 6

SYSTEM DESIGN

This chapter represent design concepts that has led to the current implementation of the prototype of this project. The design of this system is going to be carried out using the requirements analyzed in the previous chapter in order to produce a description of the systems internal structure that will serve as the basis to implement the system. This will result in the systems architecture, showing how the system will be decomposed and organized into components alongside the interface of those components. The model design of this system, will form a blueprint for the implementation that will be put together to achieve the project objectives and best performance for the final product. This system design consists of activities that fit between software requirements analysis and software construction. The algorithms that compute the functionalities of this system have been discussed further in this chapter. The various outputs for this design are functional specification, detailed design, user interface specification, data model, prototype implementation plan

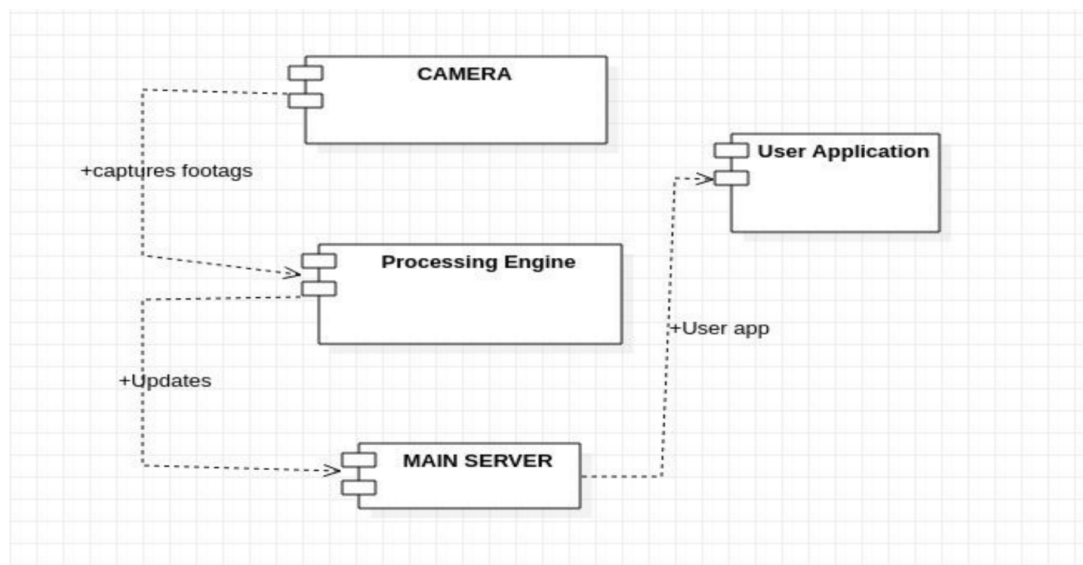


Fig. 6.1 functional flow design

6.1 Data Flow Model

This is a representation of the system of this project, by way of diagrams to show the exchange of information within the system. It is a diagram that gives an overview of the system described here without going into much detail, which can be later elaborated.

Flowchart the of the algorithm with different phases describing each phases

- Capture Image
- Phase image divided into blocks of pixels
- Calculation of histograms for pixels
- Representation of the face in-terms of histograms
- Recognition of the phase

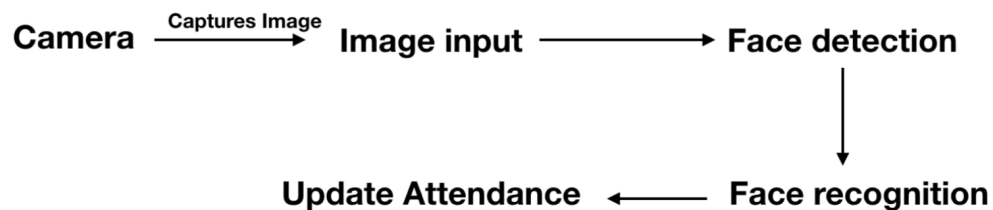


Fig. 6.2 Flow design

The Algorithm works as below:

1. First, we need to start with temp=0
2. Where I, is the training for each image
3. H=0, then Initialize the pattern histogram 145
4. Calculate the model label of LBP
5. Keep adding the corresponding bin by 1.
6. Get the greatest LBP feature during each face image and then merging into the unique vector.
7. It's time to compare the features.
8. Finally, if it resembles with the stored database the image is recognized.

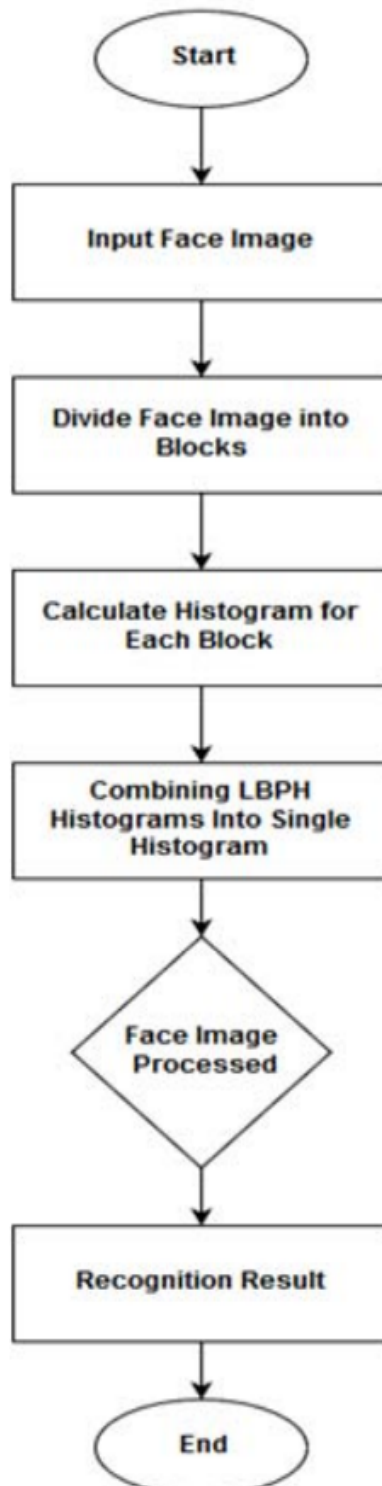


Fig. 6.3 Flow design flowchart

The above diagram represents the activity of the functionalities that will be performed by the system. The activities will be actioned by user manually. The diamond triangle represents decisions either carried out by the user or the system. After cropping the detected faces, the user can either decide to exit or carry

on to the recognition phase. At each phase, the results will be displayed by way of an alert dialogue box (represented by an arrow rectangle on the diagram) or axis shown on the GUI. At exit the system is shut down and the window is destroyed

The above flow design chart depicts how the face is detected using LBPH. The LBPH algorithm can further be extended to recognise faces but with less accuracy. The next design chart gives a pictorial representation of the same

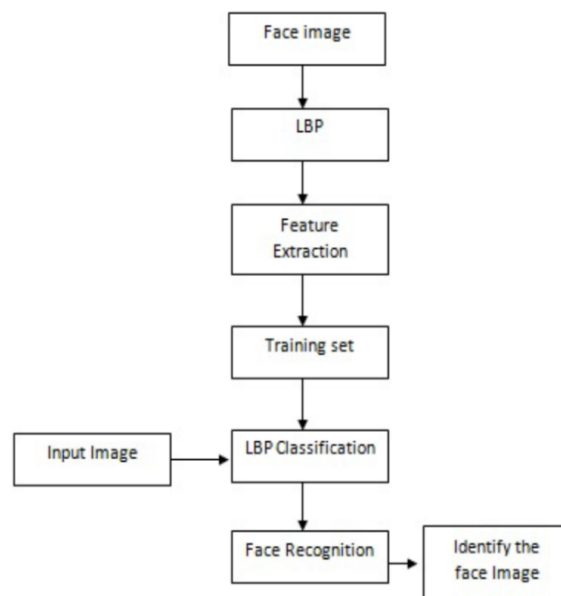


Fig. 6.4 Detailed Flow design

6.2 Structural Model

A structural model will display of the system of this project in terms of components and their relationships. For example, the system of this project is modelled by architectural design to illustrate the systems responds to events based on the system environment and the interaction between other components both externally and internally. The system of this project will also be represented using a behavioural diagram based on the Unified Modelling language (UML). After a clear understanding of the requirements and assigned components for the system in this project, the method chosen to inform the

implementation phase is described below. During this phase, the client provided feedback to match specific objectives. The prototype has been built iteratively by using the requirements gathered in the requirements and analysis section in the previous chapter

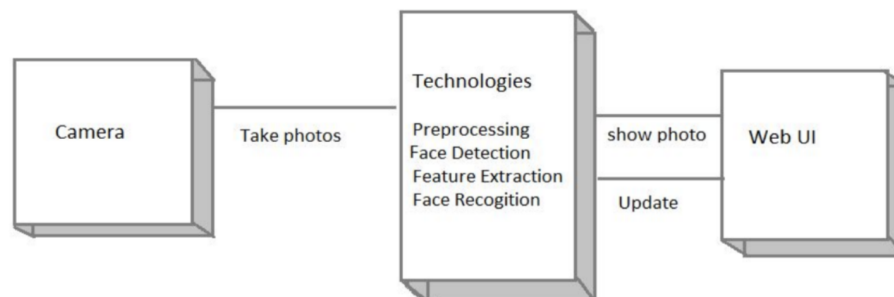


Fig. 6.5 Structural design

This also shows the interaction between software (Internal) components and hardware (External) components with an interface to establish a framework to achieve system objectives. Both external and internal components have been considered. The internal component incorporates all the functionalities with a Graphical User Interface to allow the user to interact with the system. External and Internal components of the System. The Image input and image dataset are all dependencies for the excellent performance of the system. The system will perform best if the images are of good resolution. A good resolution image will enhance face detection

to a wider range. Also, a good resolution of the image will have nearly all the pixels required for training if the images which will boost matching of images on the dataset. The accuracy of the system will very much rely on the resolution and quality of the image and how it is trained for recognition

CHAPTER - 7

DETAILED DESIGN

7.1 Master Class Diagram

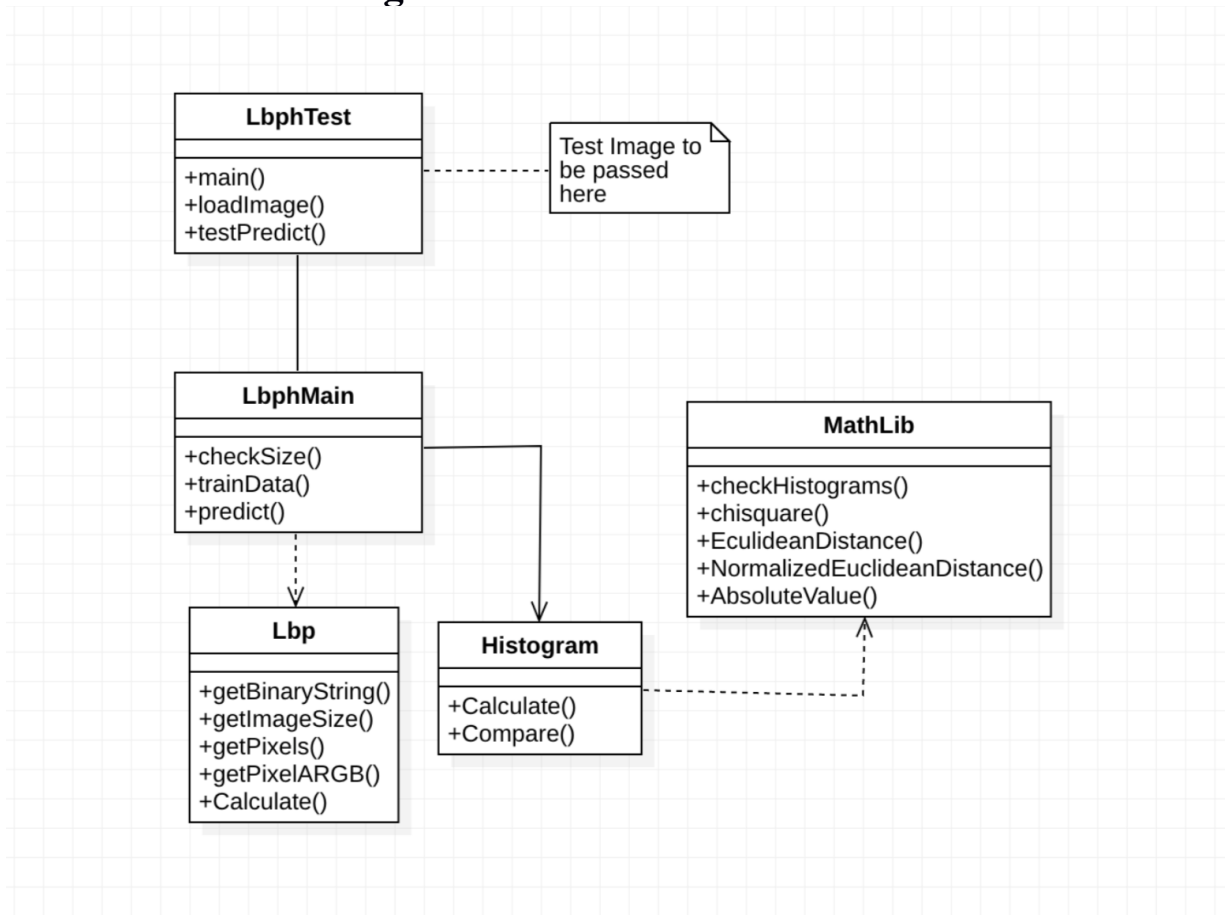


Fig 7.1 master class diagram

Figure 7.1 depicts the entire structure of the modules that have been developed for the project. This is linked extensively with the LBPH hierarchical structure of various classes and methods. Each class performs specific functionalities as described in below sections. These five classes provide main functionalities with description of each class in this class diagram will be given. A diagram of the entire system will be given at a high level and then broken down into sub levels. Classes maybe repeated across class diagrams, to show the interfaces with other classes. The detailed explanation of each class with its methods will be covered in the low-level design document.

7.1.1 Module 1: LBPH main and test classes

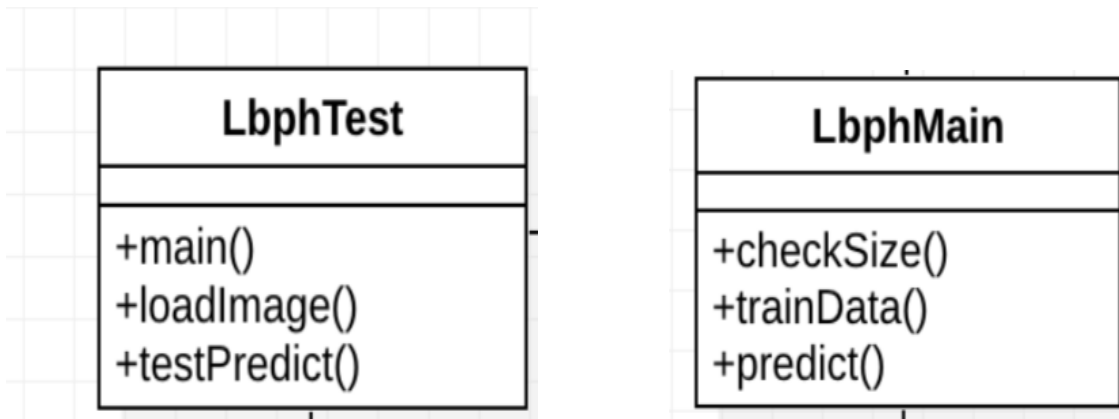


Fig. 7.2 LBPH main and test class Diagram

The Lbph has the main function which is responsible for all the actions. It will load the image and performs test prediction using two functionalities. Lbph main provides all the main functionalities such as checking and correcting the size of all the images to proper values, training the data for the module to work and predicting the face by comparing with actual trained data.

7.1.2 Module 2: MathLib class

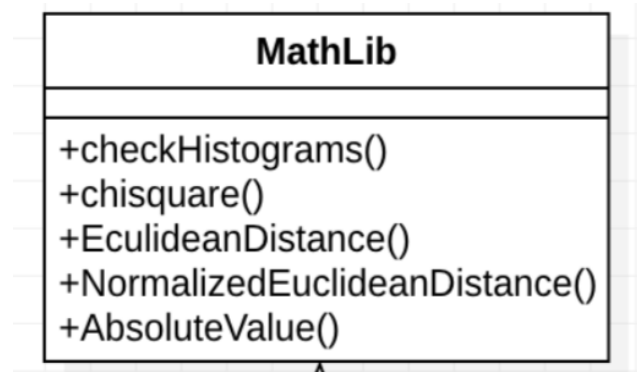


Fig. 7.3 Mathlib class Diagram

MathLib class provides all the mathematical related functionalities such as comparing two histograms, calculating the chi-square distance between two histograms, calculating the

Euclidean distance between two histograms, calculating the Normalized-Euclidean distance between two histograms and calculating the absolute difference between two histograms.

7.1.3 Module 3: Histogram and Lbp class

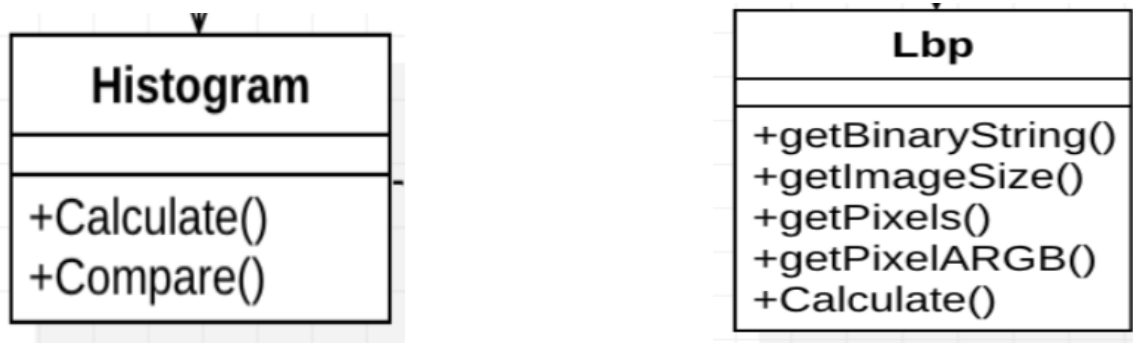


Fig. 7.4 Histogram class Diagram

Histogram will calculate the histograms of the images passed as pixel value to it and It will take X-axis as value between 0 to 255 and Y-axis as decimal corresponding value of binary pixel calculated by the algorithm. Lbp will work on pixels of the images, getting binary string which is the binary value according LBPH algorithm , getting the size of the image in pixels with X * X format, getting RGB values of the pixels.

7.2 Designing the Algorithm

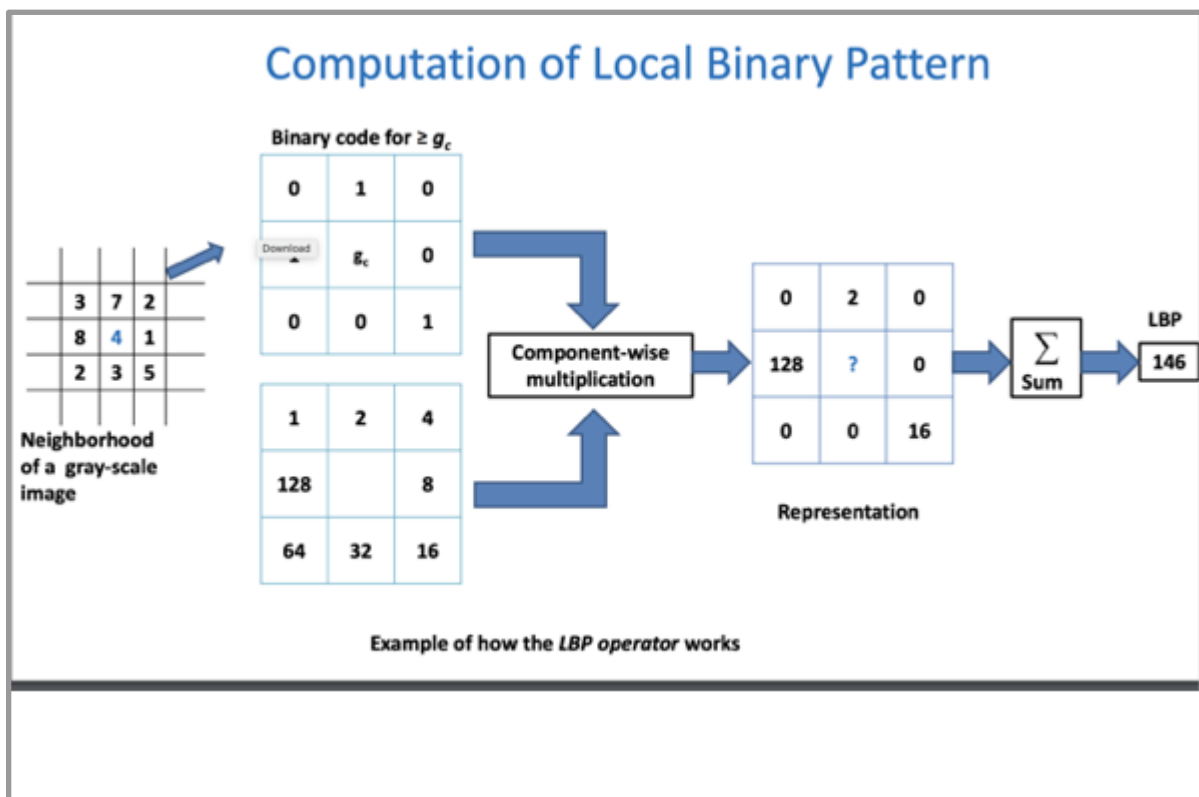


Fig 7.2 LBPH algorithm

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of

oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

Using the LBP combined with histograms we can represent the face images with a simple data vector.

As LBP is a visual descriptor it can also be used for face recognition tasks, as can be seen in the following step-by-step explanation.

7.2.1 Step-by-Step Approach

Now that we know a little more about face recognition and the LBPH, let's go further and see the steps of the algorithm:

Parameters: the LBPH uses 4 parameters:

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

Training the Algorithm

First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

Applying the LBP operation

The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors. Procedures involved in the above steps are explained

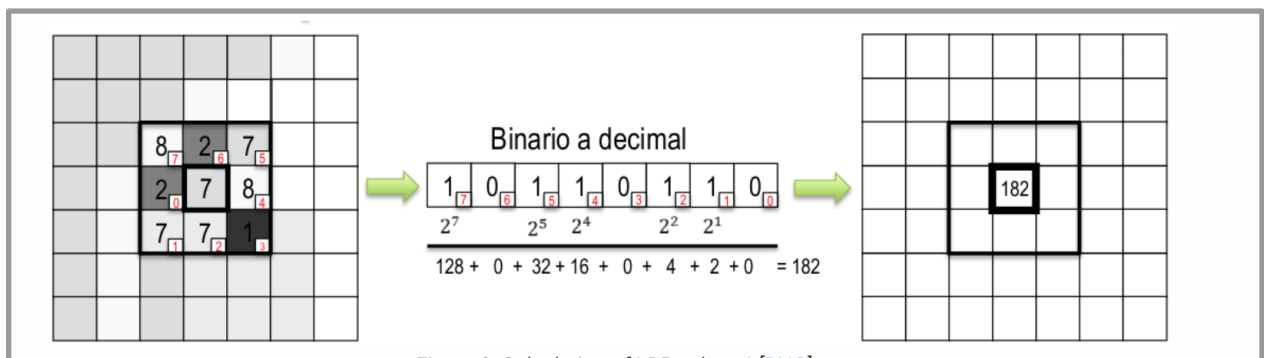


Fig 7.3 binary to decimal image

Suppose we have a facial image in grayscale.

- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.

- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

Extracting the Histograms:

Using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids

we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have $8 \times 8 \times 256 = 16384$ positions in the final histogram. The final histogram represents the characteristics of the image original image.

Performing the face recognition

In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.

- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^n (hist1_i - hist2_i)^2}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a 'confidence' measurement. Note: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

LBPH is one of the easiest face recognition algorithms, advantages of using this algorithm are as follows.

- It can represent local features in the images.
- It is possible to get great results (mainly in a controlled environment).
- It is robust against monotonic gray scale transformations.
- It is provided by the OpenCV library (Open Source Computer Vision Library).

7.3 Some of the distance methods used in the algorithm are

- Euclidean distance
- Normalised-euclidean distance
- Chi-square
- Absolute value

7.3.1 Euclidean distance

In the context of Euclidean geometry, a metric is established in one dimension by fixing two points on a line, and choosing one to be the origin. The length of the line segment between these points defines the unit of distance and the direction from the origin to the second point is defined as the positive direction. This line segment may be translated along the line to build longer segments whose lengths correspond to multiples of the unit distance. In this manner real numbers can be associated to points on the line (as the distance from the origin to the point) and these are the Cartesian coordinates of the points on what may now be called the real line. As an alternate way to establish the metric, instead of choosing two points on the line, choose one point to be the origin, a unit of length and a direction along the line to call positive. The second point is then uniquely determined as the point on the line that is at a distance of one positive unit from the origin.

The distance between any two points on the real line is the absolute value of the numerical difference of their coordinates. It is common to identify the name of a point with its Cartesian coordinate. In one dimension, there is a single homogeneous, translation-invariant metric (in other words, a distance that is induced by a norm), up to a scale factor of length, which is the Euclidean distance.

7.3.2 Normalised-euclidean distance

The normalized squared euclidean distance gives the squared distance between two vectors where their lengths have been scaled to have unit norm. This is helpful when the direction of the vector is meaningful but the magnitude is not

7.3.2 Chi-square distance

The chi squared distance $d(x,y)$ is, as you already know, a distance between two histograms $x=[x_1,...,x_n]$ and $y=[y_1,...,y_n]$ having n bins both. Moreover, both histograms are normalized, i.e. their entries sum up to one.

The distance measure d is usually defined (although alternative definitions exist) as $d(x,y) = \sum (x_i - y_i)^2 / (x_i + y_i) / 2$. It is often used in computer vision to compute distances between some bag-of-visual-word representations of images.

The name of the distance is derived from Pearson's chi squared test statistic $X^2(x,y) = \sum (x_i - y_i)^2 / x_i$ for comparing discrete probability distributions (i.e histograms). However, unlike the test statistic, $d(x,y)$ is symmetric wrt. x and y , which is often useful in practice, e.g., when you want to construct a kernel out of the histogram distances.

7.3.4 Absolute value

Absolute value distance is simply the difference between two quantities with their actual value without any squaring or other modifications. It is the simplest among all the distance measurements.

CHAPTER - 8

IMPLEMENTATION AND PSEUDOCODE

8.1 Implementation Language – JAVA

JAVA is a open-source ,general-purpose programming language that is class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible .It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to "bytecode" that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture making it faster than any interpreter languages like python.

There are total of 5 packages that is built in this project , as to provide modularisation

1.LBPH

This package is responsible for providing interface for testing, predicting, training and loading test data of images from paths .Each functionality is differentiated into several methods :

Firstly, we convert the input color image to grayscale, since LBP works on grayscale images. For each pixel in the grayscale image, a neighbourhood is selected around the current pixel and then we calculate the LBP value for the pixel using the neighbourhood. After calculating the LBP value of the current pixel, we update the corresponding pixel location in the LBP mask (It is of same height and width as the input image.) with the LBP value calculated as shown below. In the image, we have 8 neighboring pixels.

Training data class :

```
TrainingData {  
    public ArrayList<String> labels;  
    public ArrayList<BufferedImage> imageArray;  
    public ArrayList<ArrayList<Integer>> histogram;
```

```
public TrainingData(BufferedImage img,String str,ArrayList<Integer> arr){
    this.labels.add(str);
    this.imageArray.add(img);
    this.histogram.add(arr);

}

public TrainingData(ArrayList<BufferedImage> imageArrayList, ArrayList<String> labels,
ArrayList<ArrayList<Integer>> hist){
    this.histogram=hist;
    this.labels=labels;
    this.imageArray=imageArrayList;
}
}
```

Parameter Class:

```
Params {
    public int radius;
    public int neighbours;
    public int gridX;
    public int gridY;

    public Params(int rad,int nei,int gx,int gy){
        this.radius=rad;
        this.neighbours=nei;
        this.gridX=gx;
        this.gridY=gy;
    }
}
```

2.LBP

To calculate the LBP value for a pixel in the grayscale image, we compare the central pixel value with the neighbouring pixel values. We can start from any neighbouring pixel and then we can

transverse either in clockwise or anti-clockwise direction but we must use the same order for all the pixels. Since there are 8 neighbouring pixels – for each pixel, we will perform 8 comparisons. The results of the comparisons are stored in a 8-bit binary array.

If the current pixel value is greater or equal to the neighbouring pixel value, the corresponding bit in the binary array is set to 1 else if the current pixel value is less than the neighbouring pixel value, the corresponding bit in the binary array is set to 0.

The whole process is shown in the image below (Figure 2). The current (central) pixel has value 7. We start comparing from the neighbouring pixel where the label 0. The value of the neighbouring pixel with label 0 is 2. Since it is less than the current pixel value which is 7, we reset the 0th bit location in the 8 bit binary array to 0. We then iterate in the counter-clockwise direction. The next label location 1 have value 7 which is equal to the current pixel value, so we set the 1st bit location in the 8 bit binary to 1. We then continue to move to the next neighbouring pixel until we reach the 8th neighbouring pixel. Then the 8-bit binary pattern is converted to a decimal number and the decimal number is then stored in the corresponding pixel location in the LBP mask.

Once we have calculated the LBP Mask, we calculate the LBP histogram. The LBP mask values range from 0 to 255, so our LBP Descriptor will be of size 1x256. We then normalize the LBP histogram. The image below shows the scheme of the algorithm -

1. Load the color image.
2. Convert to grayscale image.
3. Calculate the LBP mask.
4. Calculate the LBP Histogram and normalize it.

This package is mainly intended to convert the image into pixel values 0-255

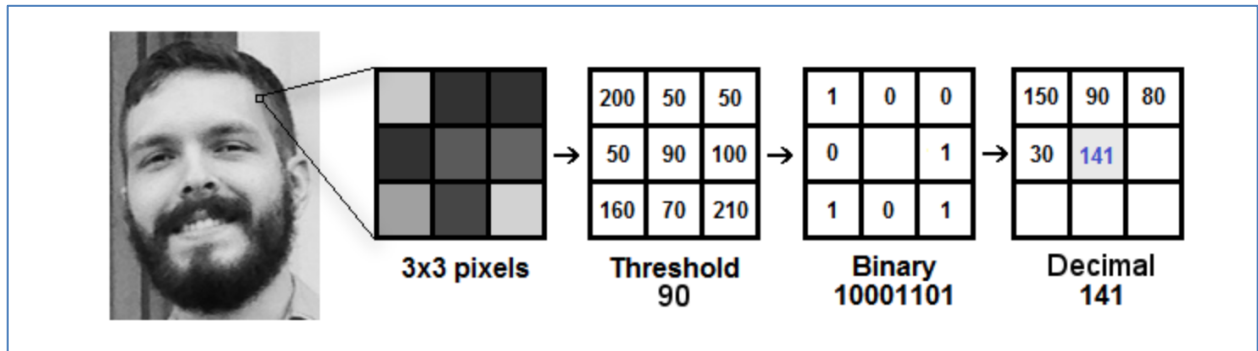


Fig 8.1: LBPH processing

As the above diagram pointer traverses each pixel and keeping the same as the threshold it converts into binary number and appends into the list.

```
Calculate(BufferedImage img,int rad,int neigh){
    //pixel 2d arraylist
    ArrayList<ArrayList<Integer>> lbpPixels=new ArrayList<ArrayList<Integer>>();
    if(img==null){
        return null;
    }
    if(rad<=0||neigh<=0){
        return null;
    }

    ArrayList<ArrayList<Integer>> pixels= GetPixels(img);;
    int[] out=getImageSize(img);

    for(int x=1;x<out[1]-1;x++){
        ArrayList<Integer> curRow=new ArrayList<Integer>();
```

```
for(int y=1;y<out[0]-1;y++){
    int threshold=pixels.get(x).get(y);
    String binResult="";

    //traverse to create decimal values
    for(int tempX=x-1 ; tempX<=x+1; tempX++){
        for (int tempY=y-1;tempY<=y+1;tempY++){
            if(tempX!=x||tempY!=y){

                //set value to 1 or 0 based on threshold
                binResult+=getBinaryString(threshold,pixels.get(tempX).get(tempY));
            }
        }
    }

    //convert into decimal from binary value
    int decimalValue = Integer.parseInt(binResult, 2);
    curRow.add(decimalValue);
}
lbpPixels.add(curRow);
}

return lbpPixels;
}
```

To summarize this method it gets the value of each pixel from the image and compare it with the surrounding pixels by keeping self pixel value as the threshold and converts into binary number by giving 1/0 to its eight neighbours . These values 0-255 will be used to form histogram of size 255*255 where each field represents the relative value of the pixel.

One advantage of LBP is that it is illumination and translation invariant. We have selected a 8 point neighbourhood, but most implementations use a circular neighbourhood as shown below. In the code, we will use a circular neighbourhood.

3. Histogram package

Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image

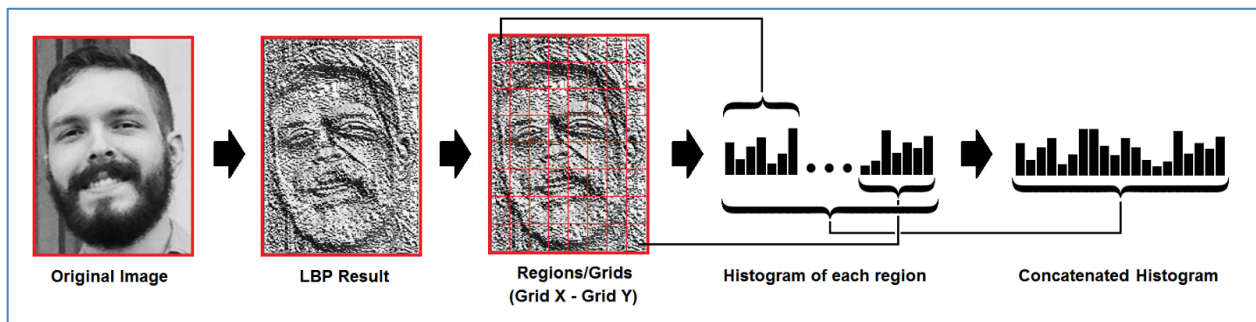


Fig 8.2 : Histogram transformation

Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have $8 \times 8 \times 256 = 16,384$ positions in the final histogram. The final histogram represents the characteristics of the image original image.

```
Calculate(ArrayList<ArrayList<Integer>> pixels, int gridX, int gridY)
```

```
{
    ArrayList<Integer> hist = new ArrayList<Integer>();
    int rows,cols ;
    int gridWidth, gridHeight ;
    int gX,gY ;
```

```
int startPosX, startPosY , endPosX, endPosY ;

int[] regionHistogram;
int x,y ;
// Check the pixels 'matrix'
if(pixels.size()== 0)
{
    System.out.println("The pixels slice passed to the GetHistogram function is empty") ;

    return hist ;
}

// Get the 'matrix' dimensions
rows = pixels.size();
cols = pixels.get(0).size() ;

// Check the grid (X and Y)
if(gridX <= 0 || gridX >= cols)
{
    System.out.println("Invalid grid X passed to the GetHistogram function") ;
    return hist;
}
if(gridY <= 0 || gridY >= rows)
{
    System.out.println("Invalid grid Y passed to the GetHistogram function");
    return hist;
}

// Get the size (width and height) of each region
gridWidth = cols / gridX ;
gridHeight = rows / gridY ;

// Calculates the histogram of each grid
```

```
for(gX = 0; gX < gridX; gX++)
{
    for(gY = 0; gY < gridY; gY++)
    {
        // Create a slice with empty 256 positions
        regionHistogram = new int[256] ;

        // Define the start and end positions for the following loop
        startPosX = gX * gridWidth;
        startPosY = gY * gridHeight;

        endPosX = (gX + 1) * gridWidth;
        endPosY = (gY + 1) * gridHeight;

        // Make sure that no pixel has been leave at the end
        if(gX == gridX-1)
        {
            endPosX = cols ;
        }
        if(gY == gridY-1)
        {
            endPosY = rows ;
        }

        // Creates the histogram for the current region
        for(x = startPosX; x < endPosX; x++)
        {
            for(y = startPosY; y < endPosY; y++)
            {
                // Make sure we are trying to access a valid position
                if(x < pixels.size())
                {
                    if(y < pixels.get(x).size())
```

```
        {
            if((pixels.get(x).get(y)) < regionHistogram.length)
            {
                regionHistogram[pixels.get(x).get(y)] += 1 ;
            }
        }
    }
}

// Concatenate two slices
for(int i=0;i<regionHistogram.length;i++){
    hist.add(regionHistogram[i]);
}

return hist ; //final output of histogram of each image
}
```

4.MathLib Package

In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **euclidean distance**, **chi-square**, **absolute value**, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^n (hist1_i - hist2_i)^2}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a ‘**confidence**’ measurement. **Note:** don’t be fooled about the ‘confidence’ name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

4.1 Chisquare distance

```
public static int ChiSquare(ArrayList<Integer> hist1, ArrayList<Integer>hist2) {
```

```
    // Check the histogram sizes
```

```
    int err = checkHistograms(hist1, hist2);
```

```
    if ( err == 0){
```

```
        return 0;
```

```
    }
```

```
    int index;
```

```
    int sum = 0;
```

```
    int numerator=1;
```

```
    int denominator=1;
```

```
    for (index = 0; index < hist1.size(); index++ ){
```

```
        numerator = (int)Math.pow(hist1.get(index) - hist2.get(index), 2);
```

```
        denominator=(hist1.get(index)>0)?hist1.get(index):1;
```

```
        sum += numerator / denominator;
    }
    return sum;
}
```

4.2 Euclidean Distance

```
public static int EuclideanDistance(ArrayList<Integer> hist1, ArrayList<Integer> hist2) {

    // Check the histogram sizes
    int err = checkHistograms(hist1, hist2);
    if ( err == 0){
        return 0;
    }

    int index;
    int sum=0;
    for (index = 0; index < hist1.size(); index++ ){
        sum += Math.pow(hist1.get(index) - hist2.get(index), 2);

    }
    return (int) Math.sqrt(sum);
}
```

4.3 Normalized-Euclidean Distance

```
public static int NormalizedEuclideanDistance(ArrayList<Integer> hist1, ArrayList<Integer> hist2)
{

    // Check the histogram sizes
    int err = checkHistograms(hist1, hist2);
    if ( err == 0){
        return 0;
    }
}
```



```
int index;
int sum=0;
int n=hist1.size();
for (index = 0; index < hist1.size(); index++) {
    sum += Math.pow(hist1.get(index) - hist2.get(index), 2) / n;
}
return (int) Math.sqrt(sum);
}
```

4.4 Absolute Distance

```
public static int AbsoluteValue(ArrayList<Integer> hist1, ArrayList<Integer>hist2) {

    // Check the histogram sizes
    int err = checkHistograms(hist1, hist2);
    if ( err == 0){
        return 0;
    }

    int index;
    int sum=0;

    for (index = 0; index < hist1.size(); index++) {
        sum += Math.abs(hist1.get(index) - hist2.get(index));
    }
    return sum;
}
```

CHAPTER 9

TESTING

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

Testing for this project has been carried out in various stages during development of the code , to ensure no further errors may introduce . During the transformation of the images into ArrayList , we ensured right pixel values are converting and appending into list by changing the input image color , hence giving same pixel value for entire image .

Distance metrics was tested externally by changing the images passed on to the training data .

Initial final testing was done with sample set of data with pre-defined labels and validated the output accordingly

CHAPTER 10

RESULTS AND DISCUSSION

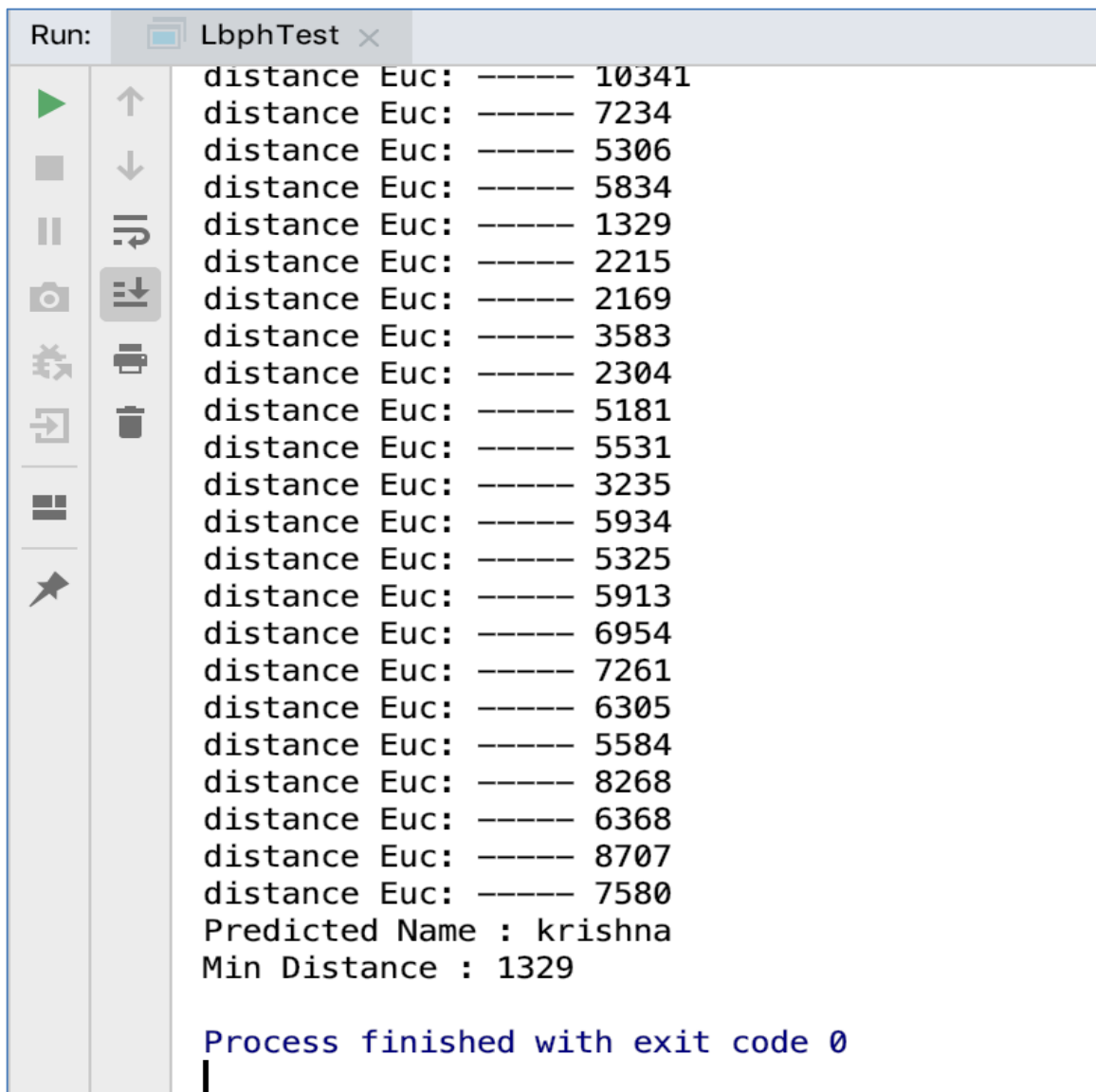
Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

- This technique is best suited for small set of image data and can give best results with less number of training images
- This has been implemented from scratch without using any pre-defined image processing libraries for LBPH algorithm
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **Euclidean distance**, **chi-square**, **absolute value**, etc.
- Once the set of trained metrics is generated we can save it and use it as reference model during image recognition of an individual, if there is any addition/modification or deletion of images we can just do that by appending to the trained histogram list.
- The test images can be identified by the closest histogram , algorithm also returns the nearest distance which can be used to measure confidence .
- In this experiment, each image in the face database has the distinct ID number. First, prepare the face database, and then extract the LBP texture features of each test image. Finally, classify and recognize the face information. For this test we have collected 250 face images, those face images are taken with a webcam. We compare the input face images with database face images and work as if the given appearance images, after

- extracting features compared with the dataset so finally we can figure-out the face image is favorably recognized otherwise the face image would not be recognized.
- Based on the algorithm, this information of face image of known and an unknown identity is compared with the face image of known individuals from the available database. In the research, we have performed major three tasks, capture, train, and recognize the face images by using the camera. A. Face Detection In face detection step, the system detects the face in an input image via camera and captures the gray scale image. 146 B. Training Face Images After image acquisition and pre-processing task, we have to perform dataset training. For training phase, the training recognizer is applied to store the histogram values of face images.

CHAPTER 11

SNAPSHOTS



```
Run: LbphTest x
distance Euc: ----- 10341
distance Euc: ----- 7234
distance Euc: ----- 5306
distance Euc: ----- 5834
distance Euc: ----- 1329
distance Euc: ----- 2215
distance Euc: ----- 2169
distance Euc: ----- 3583
distance Euc: ----- 2304
distance Euc: ----- 5181
distance Euc: ----- 5531
distance Euc: ----- 3235
distance Euc: ----- 5934
distance Euc: ----- 5325
distance Euc: ----- 5913
distance Euc: ----- 6954
distance Euc: ----- 7261
distance Euc: ----- 6305
distance Euc: ----- 5584
distance Euc: ----- 8268
distance Euc: ----- 6368
distance Euc: ----- 8707
distance Euc: ----- 7580
Predicted Name : krishna
Min Distance : 1329

Process finished with exit code 0
```

```
Run: LbphTest x
distance chisquare: -----184141
distance chisquare: -----142623
distance chisquare: -----161587
distance chisquare: -----145256
distance chisquare: -----173373
distance chisquare: -----154941
distance chisquare: -----180879
distance chisquare: -----298526
distance chisquare: -----296753
distance chisquare: -----250752
distance chisquare: -----287068
distance chisquare: -----527035
distance chisquare: -----343320
distance chisquare: -----245395
distance chisquare: -----219378
distance chisquare: -----21720
distance chisquare: -----34378
distance chisquare: -----34240
distance chisquare: -----77051
distance chisquare: -----39675
distance chisquare: -----193273
distance chisquare: -----209872
distance chisquare: -----66033
distance chisquare: -----209804
distance chisquare: -----129487
distance chisquare: -----123188
distance chisquare: -----321314
distance chisquare: -----286393
distance chisquare: -----234148
distance chisquare: -----158726
distance chisquare: -----166109
distance chisquare: -----185475
distance chisquare: -----368949
distance chisquare: -----226193
Predicted Name : krishna
Min Distance : 21720
Process finished with exit code 0
```

CHAPTER 12

CONCLUSION

We have successfully implemented face detection and recognition using LBPH algorithm in pure Java. Libraries like OpenCV and lib were not used during the course of the project. As a matter of fact, none of the built in libraries related to image processing were used. Though LBPH has less accuracy than Hair cascade and other popular image processing algorithms, it works well in poor lighted images and hence takes care of occlusions. We have made use of Euclidean metric to calculate distance between the histograms of test image and trained image. Other metrics were also tried but Euclidean distance gave best results.

The accuracy of the system was based on the face recognition rate and the efficiency of the system was determined by the computational time. We however have encountered problems, which are prone to most facial recognition systems. The system was affected by the illumination problem thus variation in lighting conditions. Whenever there was insufficient lighting the room, the recognition rate declined, as there was a significant figure of false positives. Another challenge was that of hardware, facial recognition requires high performance computing hardware and most particularly a high definition camera with a high resolution. The development of the proposed system was narrowed towards finding faces in a class of students.

CHAPTER 13

FUTURE ENHANCEMENTS

However, the same system can be improved by implementing it on DSP processors and using other Hardware devices like Raspberry Pi.

The developed system could identify human faces in real time so it can be integrated with google maps to track any subject of interest.

Furthermore, facial recognition systems can also be used in the development of attendance management system for faculty as well

It can also be extended for investigating criminal activities.

REFERENCES/ BIBLIOGRAPHY

- <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>
- https://en.wikipedia.org/wiki/Euclidean_distance
- https://www.researchgate.net/publication/326986115_Face_Detection_and_Recognition_Student_Attendance_System
- <https://www.geeksforgeeks.org/program-make-histogram-array/>
- <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffe121d78>
- https://www.researchgate.net/publication/270583005_A_Review_Paper_on_Face_Recognition_Techniques
- https://globaljournals.org/GJCST_Volume13/1-Face-Recognition-using-Local.pdf
- https://www.researchgate.net/publication/308836179_A_comparative_study_between_LBP_and_Haar-like_features_for_Face_Detection_using_OpenCV
- https://link.springer.com/chapter/10.1007/978-3-319-12484-1_12
- http://www.academia.edu/Documents/in/Local_Binary_Pattern_LBP
- <https://medium.com/search?q=face%20recognition>
- <https://ieeexplore.ieee.org/document/7947889>
- https://www.researchgate.net/.../What_is_chi-squared_distance_I_need_help
- <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>
- http://www.scholarpedia.org/article/Local_Binary_Patterns
- <http://troindia.in/journal/ijacet/vol5iss1/1-4.pdf>