



Om Sakthi

ADHIPARASAKTHI COLLEGE OF ENGINEERING

G.B.NAGAR, KALAVAI - 632 506. RANIPET DT.,



**FACE RECOGNITION BASED STUDENT
ATTENDANCE SYSTEM**

CS8611-MINI PROJECT

Submitted by

D. JAYAKUMAR

510119104010

G. NITHISH KUMAR

510119104015

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

ACADEMIC YEAR: 2021-2022 (EVEN SEM)

ANNA UNIVERSITY, CHENNAI - 600 025.

BONAFIDE CERTIFICATE

Certified that this project report “**FACE RECOGNITION BASED STUDENT ATTENDANCE SYSTEM**” is the bonafied work of “**D. JAYAKUMAR, G. NITHISH KUMAR**” Who carried the project work under my supervision.

SIGNATURE

Mr. E. SIVARAJAN, M.E.,(Ph.D)

SUPERVISOR

Department of Computer Science and
Engineering.

Adhiparasakthi College of
Engineering.

G.B Nagar, Kalavai-632506.

SIGNATURE

Mr. SUKKRIVAN, M.Tech.,(Ph.D),

HEAD OF THE DEPARTMENT,

Department of Computer Science
and Engineering.

Adhiparasakthi College of
Engineering.

G.B Nagar, Kalavai-632506.

Submitted for the university CS8611-Mini Project Examination & Viva-Voice held
on_____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. We give all honor and praise to GOD ALMIGHTY who gave us wisdom and guided us during the entire course of our project.

We express our heartfelt gratitude towards Dr. K. BALAKANNAN., M.E., Ph.D., MISTE., MIAE., Principal, Adhiparasakthi College of Engineering, G.B. Nagar, Kalavai for granting us permission to work on this project.

We also express our sincere thanks to Mr. B. SUKRIVAN M.Tech., (Ph. D), Head of the department, Computer Science and Engineering, G.B. Nagar, Kalavai for allowing me to undertake this project.

We wish to express my deep sense of gratitude to our guide Mr. E. Sivarajan, M.E., (Ph. D), Assistant professor, Computer Science and Engineering, G.B. Nagar, Kalavai for his valuable guidance throughout the course of this project.

We also wish to express our sincere thanks to the review committee members, all department teaching and non-teaching staff members for their valuable suggestions and providing the required infrastructure.

Finally, we wish to express our indebtedness to our beloved parents, the ultimate force behind our grand success.

Table of Contents

Abstract	
1 Introduction	8
2 Introduction to Facial Recognition	9
Face Detection	10
Feature Extraction	10
Face Recognition	10
2.1 History of Facial Recognition	10
2.2 Importance of Facial Recognition System	11
2.3 Challenges of Facial Recognition System	12
3 Requirement Analysis and Feasibility Study	13
3.1 Literature Review	13
3.2 Requirement Analysis	13
3.2.1 Functional Requirements	13
Admin Module	14
Teacher Module	15
3.2.2 Non-Functional Requirements	15
3.3 Feasibility Analysis	16
3.3.1 Operational Feasibility	16
3.3.2 Economic Feasibility	16

3.3.3	Technical Feasibility	17
4	Method and Materials	17
4.1	Tools and Technologies	17
4.1.1	Python	17
4.1.2	Django	18
4.1.3	OpenCV	19
4.2	Methodology	20
4.3	Implementation	20
4.3.1	System Design	21
	Presentation Layer	21
	Application Layer	21
	Data Layer	21
4.3.2	Database Design	22
4.3.3	Interface Design	24
5	Discussion	30
6	Face Detection	30
7	Face Recognition	37
8	Program Code	49
9	Conclusion	59
	References	60

List of Abbreviations

LBPH: Local Binary Pattern Histogram

SDLC: Software Development Life Cycle

SQL: Structured Query Language

CSS: Cascading Style Sheets

HTML: Hypertext Markup Language

UI: User Interface

FERET: Face Recognition Technology

DARPA: Defence Advanced Research Projects Agency

NIST: National Institute of Standards and Technology

FRVTs: Face Recognition Vendor Tests

ORM: Object -relational Mapping

ABSTRACT

The main purpose of this project is to build a face recognition-based attendance monitoring system for educational institution to enhance and upgrade the current attendance system into more efficient and effective as compared to before. The current old system has a lot of ambiguity that caused inaccurate and inefficient of attendance taking. Many problems arise when the authority is unable to enforce the regulation that exist in the old system. The technology working behind will be the face recognition system. The human face is one of the natural traits that can uniquely identify an individual. Therefore, it is used to trace identity as the possibilities for a face to deviate or being duplicated is low. In this project, face databases will be created to pump data into the recognizer algorithm. Then, during the attendance taking session, faces will be compared against the database to seek for identity. When an individual is identified, its attendance will be taken down automatically saving necessary information into a excel sheet. At the end of the day, the excel sheet containing attendance information regarding all individuals are mailed to the respective faculty. The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. There are two kinds of methods that are currently popular in developed face recognition pattern namely, Eigenface method and Fisherface method. Facial image recognition Eigenface method is based on the reduction of facedimensional space using Principal Component Analysis (PCA) for facial features. The main purpose of the use of PCA on face recognition using Eigen faces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project face detection system with face recognition is Image processing. The software requirements for this project is matlab software.

Keywords- Smart Attendance System, face detection, OpenCV, NumPy

1 Introduction

The success of an educational institute begins by engaging students and having regular attendance of students. Having a higher attendance score results in higher marks, higher retention rates, and a better educational experience. It is difficult for teachers and students to build a strong relationship if students are frequently absent. This hampers teachers and students to develop their skills and make progression. In many schools, the school budgets are based on the average daily attendance of the school. If the attendance rates are low, then school budgets suffer. Hence, schools have less money to get essential classroom needs for students and eventually end up with less quality education. Therefore, the educational institute needs to have high-quality attendance data. These data provide essential information for the institute to formulate policies, programs, and practices to improve attendance rates. To increase the attendance of students, many teachers give better grades to the students with higher attendance scores.

Even though keeping attendance data is an essential part of educational institutes, there has been little advancement in the attendance system. Still, many institutes use traditional handwritten attendance or use some spreadsheet on the computer. This makes it hard for teachers to track the students' attendance data and their progress. Chances of attendance fraud in this system are relatively higher than it is in automated attendance system. Unless the attendance data is correct, schools cannot formulate proper policies and practices to improve the quality of education.

This project will help eliminate the traditional attendance system, minimize manipulation during attendance and record the arrival time of the students. It is also very easy to use and manage. Like every application, there are some setbacks to this application. The application is not one hundred percent accurate. Different factors such as image quality and lack of data sets can decrease the efficiency of the application. Administrators must add user information manually and with data sets stored associate with the risk of being lost or stolen. The fundamental introduction of our project is given in section 1. It includes a background study of the project, the objectives set to meet, scope and limitations of our final product.

The history of facial recognition and its introduction is provided in Section 2. The advantages and drawbacks of the facial recognition system are discussed in this section. Section 3 summarized the functional and non-functional requirements of the project and the feasibility of the project. Section 4 includes the development method we adopted to build the application and tools and technologies used to build the application are discussed. Section 4 discusses the structural designs of our application and how the application functions in various layers. Through various diagrams and figures, it explains step by step how the entire application works.

2 Introduction to Facial Recognition

A person's face has distinctive physical shape and characteristics that are used to identify or verify an individual. Facial recognition records this biometrics of the face. Different face recognition methods measure the biometric of the face.

Facial recognition has become a very important topic in recent years. Facial recognition is effectively applied in various applications like security systems, authentication, entrance control, surveillance system, unlocking of smartphones and social networking systems, etc. Most of the practices do not use facial recognition as the main form of conceding entry. However, with advancement in technology and algorithm, facial recognition system has the potential to replace the standard passwords and fingerprint scanners.

This project was carried out to show how a Local Binary Pattern Histogram (LBPH) face recognizer could be used for taking attendance of students. LBPH facial recognizer is a pre-trained facial recognition classifier. If enough data set are available on the face that is needed to be identified, LBPH can perform facial recognition with high accuracy. Face Recognition Student Attendance System is a desktop application that identifies and verifies student's identities with the help of a digital image. Once the recognized face matches with the stored image, the attendance is completed and marked in the database for the student. This system will provide an alternative and easier way of taking attendance.

The facial recognition system has three main phases, which are described below:

Face Detection

Face detection is the ability to identify the person's faces within the digital images. This system identifies the human face present in an image or video. We need to define a general structure of a face to determine certain picture or video contains a face (or several). Human faces have the same features such as eyes, nose, forehead, mouth, and chin. Therefore, the objective of face detection is to find the location and size of the face in an image. The located face is then used by the facial recognition algorithm.

Feature Extraction

In this phase, we are extracting the features from the detected face. In LBPH, the first local binary pattern images are computed, and a histogram is created for facial recognition. This generates a template. A template is a set of data that represents the unique and distinctive features of the detected face.

Face Recognition

Face Recognition is being able to uniquely identify and verify a person's face by comparing and analyzing a biometrics person's face. A face recognition system is an application that is used for identifying or verifying a person from a digital image.

2.1 History of Facial Recognition

Woody Bledsoe, Helen Chan Wolf, and Charles Bisson were the earliest pioneers of facial recognition. They began working to recognize the human face using a computer in 1964 and 1965. They marked various landmarks on the face such as eye centers, nose, mouth manually. They later used the computer to mathematically rotate to compensate for pose variation. The distances between the facial landmarks were computed automatically and compared with the image to match the identity. This was the dawn of facial recognition.

Sirovich and Kirby applied linear algebra to facial recognition and made it a viable biometric for business. They developed a system called "Eigenface" where less than one hundred values were required to code the facial image

accurately. In 1991, the discovery of face detection within an image by Turk and Pentland led to the beginning of automated facial recognition. This paved the way for the advancement and development of facial recognition technology.

FERET program was rolled out in the early 1990s by the DARPA and NIST for commercial facial recognition. They created a database for facial images, which included 2413 facial images that represented 856 people. In the early 2000s, to provide independent government evaluations of facial recognition system and its prototype technologies, FRVTs was designed. These evaluations provided the necessary information to deploy the facial recognition technology in the best way to the law enforcement agencies and government. Face Recognition Grand Challenge was launched in 2006 to evaluate the face recognition algorithms available. It used high-resolution images, 3D face scans, and iris images for the test. The test concluded that the new algorithm was 10 times more accurate than the algorithms of 2002 and more than 100 times more accurate than the algorithm of 1995. In recent years, Facebook has implemented facial recognition functionality to identify people featured in the user's daily updates.

In 2017, Apple launched the iPhone-X, which was the first iPhone to implement facial recognition to unlock the phone.

2.2 Importance of Facial Recognition System

Applications using facial recognition systems are widespread. They are applied in security systems, authentication systems, verification systems, surveillance systems, etc. We are interacting with face recognition systems without even realizing it. Many Businesses are using facial recognition systems for authentication, verification, and security. There are diverse applications of this system. Countries such as United States, United Kingdom, and Australia are now installing facial recognition technologies in different public spaces such as airports, cafes, shopping areas, factory areas, and government buildings. A large retail company like Alibaba is working on the development of pay-by-face technology.

Workspaces are using this technology to record the clock in and clock out time of the employees. Law enforcement agencies are installing cameras with facial recognition systems to identify criminals and search for missing persons. As facial

recognition technology and algorithms advance, we would see it being implemented more and more in our society.

2.3 Challenges of Facial Recognition System

A facial recognition system can revolutionize how businesses and governments interact with people. However, if not used properly, there are potential pitfalls with this technology. Potential misuse of personal and sensitive information is very real. Businesses and Organizations need to make sure that there are proper checks and balances and proper security before implementing this technology. Every time this technology scans someone's face, the distinct biometrics of the person is stored in a database. Depending on who owns the database and security in place to protect the database, the information can be leaked, stolen, or misused without the consent of the person. Facial recognition systems are not perfect. Data collected by humans are used to train the algorithms. If there are a lack of data and a diverse array of data to train the algorithms, the system can misidentify the person. There have been many instances where the system incorrectly identified the gender or identity of people with darker skin tones. This happened because of a lack of data representing a diverse array of people. [4]

With the advancement of new technology comes a new type of crime. Criminals could access the facial recognition data by hacking the database and track people's movement, location, and information without their consent. Criminals can cause significant damage with the aid of a facial recognition system. They can steal sensitive personal information or the identity of a person to commit a crime.

The application of facial recognition technology holds many promise. However, it needs to be handled carefully. Businesses that want to implement this technology need to implement the proper framework and facial data protection measures. If successfully managed to implement this technology, they can reap the benefits of this technology.

3 Requirement Analysis and Feasibility Study

This section of the thesis describes the requirements necessary for the project and its feasibility.

3.1 Literature Review

Viola-Jones algorithm is used to detect the face. A camera is set up in the classroom that scans the facial structure of the students. The detected face is extracted for further processing. 100 images of students are stored in the database as the dataset. These datasets are used to compare the biometrics with the detected face for facial recognition. Facial recognition is done using LBPH. LBPH extracts the histogram of the image and concatenates it to form the face descriptor by segmenting the image into the local region. The distance between the biometrics of the probe image and the trained image is calculated. If the calculated distance is less than the threshold, then the probe image is recognized. Once recognized, the name is updated into an excel sheet.

3.2 Requirement Analysis

In this section, the functionalities need to run the system are described.

3.2.1 Functional Requirements

The system has different functionalities for an admin and teachers. Admin has higher privileges than teachers. Their functionalities are described below.

1. Admin Module
2. Teachers/Students Module

Admin Module

Admin has the highest privileges among all as admin is responsible to design the system. Admin register teacher and provide unique id to the teacher. They are responsible to take images of the students and add them to the database.

Admin can view and update the details of both students and teachers. They can also view the attendance report. Figure 1 shows the use case for admin,

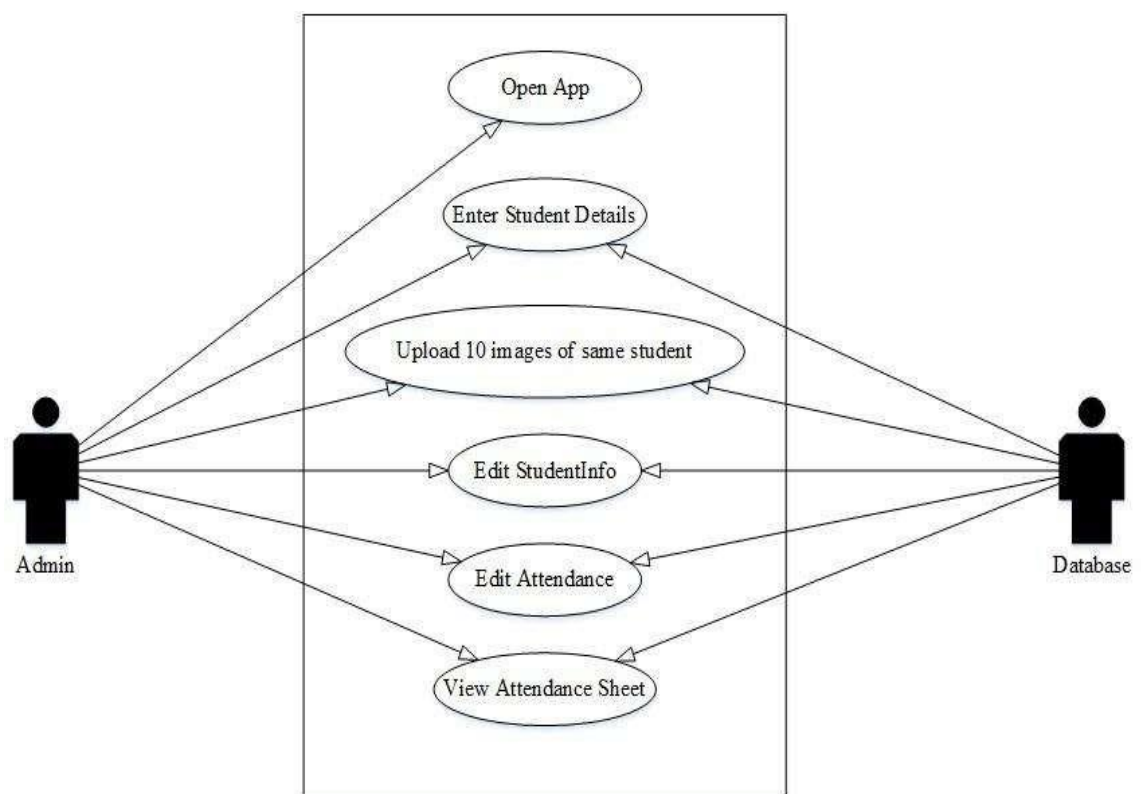


Figure 1 Use Case Admin Module

Teacher Module

Teachers can log in to the system. They can open the application and the images of the students for attendance. They can also view the attendance report. Figure 2 shows the use case for the teacher.

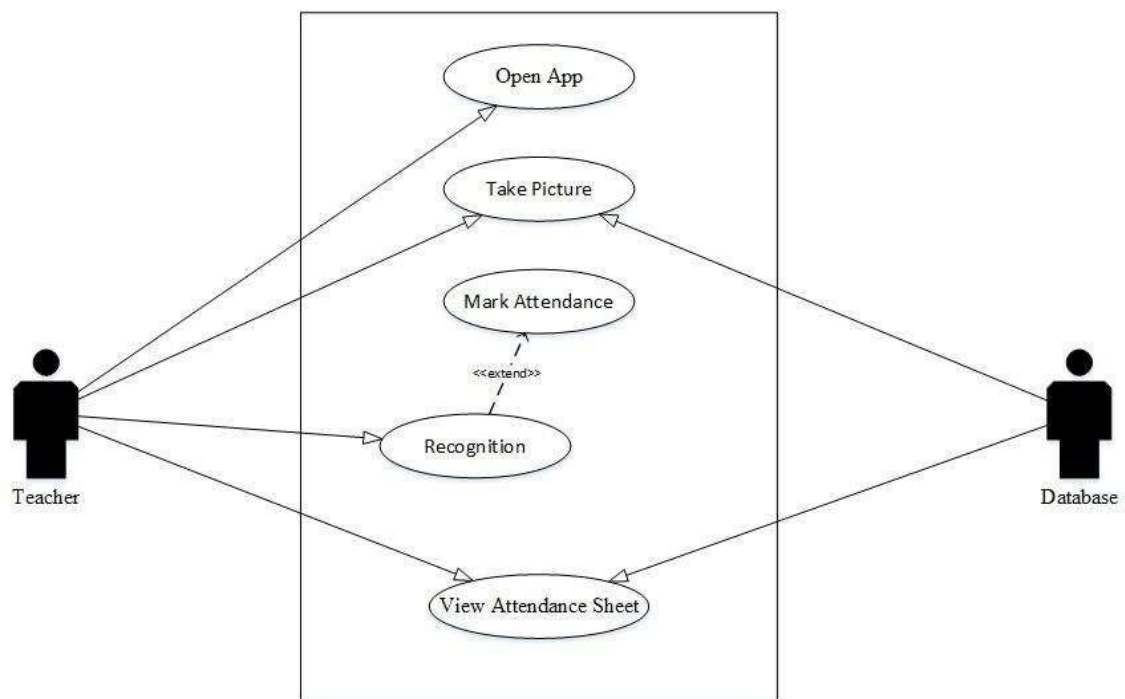


Figure 2 Use Case Teacher/Student Module

3.2.2 Non-Functional Requirements

Non-Functional Requirements are the characteristics or attributes of the system that are necessary for the smooth operation of the system. Those requirements are listed below.

- The system should perform the process accurately and precisely to avoid problems.
- The system should be easy to modify for any updates. Any errors or bugs that are identified should be easy to mend.

- The system should be secure and maintain the privacy of the students.
- The system should be easy to understand and use.
- Execution of the operation should be fast.

3.3 Feasibility Analysis

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions. . It must therefore, be conducted with an objective, unbiased approach to provide information upon which decisions can be based. Here, we discuss 3 major feasibility studies required for our project.

3.3.1 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems with the users. Operational feasibility is dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented. The project is operationally feasible for the users as nowadays almost all the teachers/staffs are familiar with digital technology.

3.3.2 Economic Feasibility

Economic feasibility defines whether the expected benefit equals or exceeds the expected costs. It is also commonly referred to as cost/benefit analysis. The procedure is to determine the benefits and the savings expected from the system and compare them with the costs. A proposed system is expected to outweigh the costs. This is a small project with no cost for development. The system is easy to understand and use. Therefore, there is no need to spend on training to use the system. This system has the potential to grow by adding functionalities for students as well as teachers. This can Hence, the project could have economic benefits in the future.

3.3.3 Technical Feasibility

Technical feasibility is carried out to determine whether the project is feasible in terms of software, hardware, personnel, and expertise, to handle the completion of the project. It considers determining resources for the proposed system.

As the system is developed using python, it is platform independent. Therefore, the users of the system can have average processing capabilities, running on any platform. The technology is one of the latest hence the system is also technically feasible.

4 Method and Materials

This is the most important section of the thesis. This section describes the detailed workflow of the project and the necessary theoretical background. Tools and Technologies

4.1 Tools and Technologies

Tools and techniques used in the project are described in this section of the thesis. This project focused was mainly focused on Python Programming and its libraries.

4.1.1 Python

Python is a high-level object-oriented programming language. It was created by Guido van Rossum in 1991 as Python 0.9.0. It was created as the successor of the ABC programming language. Python 2.0 was released on 16

October 2000 and added many features like list comprehension and garbage collecting system. On 3 December 2008, Python 3.0 was released. Python is a very popular programming language and can be used for various purposes. It is widely used for web development, software development, mathematics and data analysis, system scripting, etc. Python is a multi-purpose programming language that works on different platforms like Windows, Linux, Mac, RaspberryPie, etc. Python is popular than other programming languages because it has a simple syntax than other programming languages. Its syntax allows the programs to write code that is easier to understand and in fewer lines. It runs in an interpreter system. Hence, the code can be executed as soon as it is written. In this thesis, we use Python for web development. This project demonstrated how Python is used for an effective and reliable web application. Various Python frameworks, libraries are used in this project.

4.1.2 Django

Django is a high-level web framework based on python. Django was developed between 2003 and 2005 by a team responsible for creating and maintaining newspaper sites. It has continued to grow by releasing Django 1.0 in 2008 through the latest Django 3.1 in 2020. It enables users with rapid and secure development of the websites. It is open source, free with a thriving community, and has up-to-date documentation. Any kind of website can be built using Django. It works on any framework and delivers content in any format (JSON, HTML, XML, etc). Django provides a security framework that helps developers protect their websites. Django uses component-based architecture. It means each component is independent of the other, hence can be easily changed or replaces if needed. It provides a clear separation of different parts that enable to scale for increased traffic at any level by adding hardware. Django uses the Do not Repeat Yourself (DRY) principle. Hence, there is no unnecessary repetition of code. Django is written in Python and hence, can be run on my platforms.

4.1.3 OpenCV

OpenCV is an open-source machine learning and computer vision library. OpenCV is a cross-platform library and is free to use. It was launched in 1999. Intel launched OpenCV to advance CPU-intensive applications. It was developed in C++. It provides bindings for Java and Python programming languages. It runs in different operating systems such as Linux, Windows, OSx, etc. It focuses mainly on video capturing, image processing, and analysis. It has face detection and objects detection features. OpenCV can be used to read and write images and capture and save videos. It can perform feature detection like faces, cars, images, etc. Many established companies like Yahoo, Google, Microsoft, Intel, and many others use the library.

4.2 Methodology

This section describes how LBPH is used for face recognition. First, a dataset is collected for images and each image is labeled with a unique id. The images are divided into an 8X8 grid and converted into grayscale. A 3X3 matrix of each pixel containing its intensity (0~255) is extracted from the image. The threshold of the central value of this matrix is taken which is used to determine the neighboring value of the matrix. Each neighboring value is compared with the central value. If the neighboring value is greater or equal to the threshold value, it is set to 1. If the neighboring value is less than the threshold value, it is set to 0. Then, the matrix value will contain binary values only. The decimal value is calculated using the given formula:

$$LBP(x_c, y_c) = \sum_{n=0}^7 (i_n - i_c) 2^n$$

In the above formula, 'n' is the 8 neighbors of the central pixel, i_c , and i_n are the grey level value of the central pixel and the surrounding pixel, respectively. $S(x)$ is

1 if x is greater than or equal to the threshold. S(x) is 0 if x is less than the threshold.

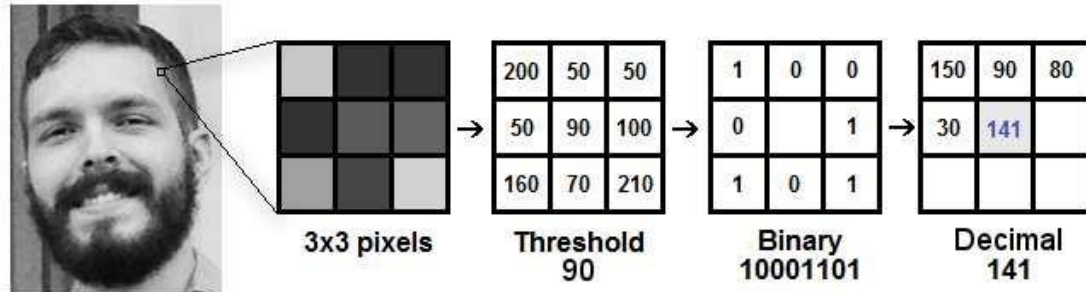


Figure 3 Extracting 3X3 matrix of a pixel.

The calculated decimal value is replaced with the central value. Hence, we obtain the characteristics of the original image in a new image. Once all the processes are complete, a histogram is extracted from each grid and are concatenated. This process is repeated for all the images and a histogram is generated. To compare two images, histograms are compared at a time. The comparison is done by Histogram Intersection. Its formula is given below:

$$\sum_{j=1} \min(I_j, M_j)$$

Here, j is the bin number and I and M are histogram 1 and histogram 2. If the intersection value is greater than 80% then, the image is successfully recognized.[9]

4.3 Implementation

This section describes how the algorithm was implemented to design the system and the testing of the system. The application was created using Python's Django framework. Both the front-end and back-end of the project were done using Django. This project implements the tools and technologies mentioned in section 4.1.

4.3.1 System Design

The project follows three-layered architecture, which is described below.

Presentation Layer

This layer is responsible for the user interface. All the components that users see and interact with within the application are in this layer.

Application Layer

Application layer controls the overall functionality of the system. Functionality such as logging into the system, facial detection, and recognition is all done in this layer.

Data Layer

In this layer, Data and Information are stored and retrieved in the database. The names, images of students as datasets, teachers are stored in the database. Once the face is matched, marking of attendance in the database. See figure 4.

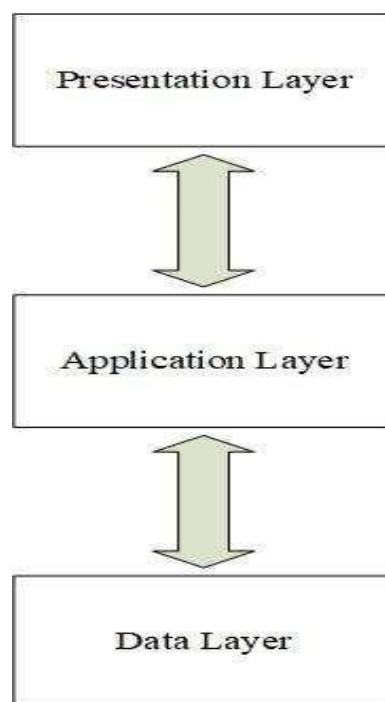


Figure 4 System Design

4.3.2 Database Design

For this project, the default Django SQLite was used to create the database. The tables are created by Django's model. It provides an ORM to the underlying database. ORM makes it easy to work with relational databases. The models in Django are Object, which is mapped to the database. When a model is created, Django creates the corresponding table without having to write any SQL code.

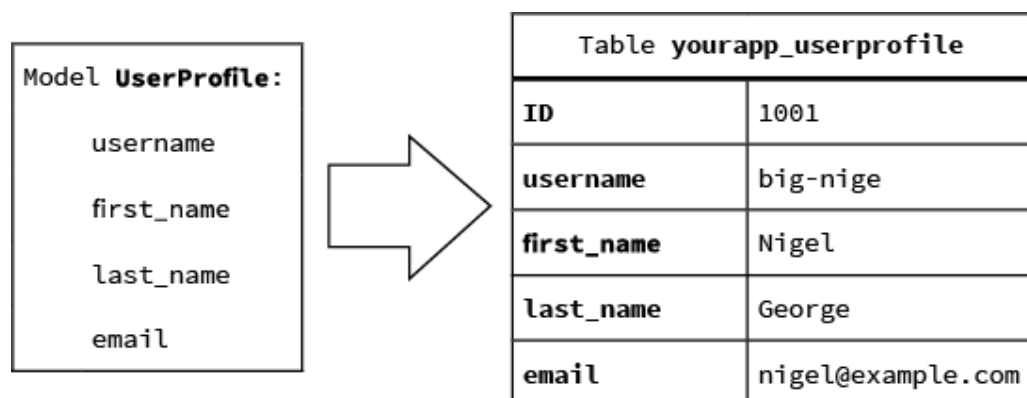


Figure 5 Django model and its corresponding table in the database

Creating a model in Django is easy. It contains essential fields needed for our data and specifies the behavior of the data. Each model is mapped to a single table in the database. Models in python are classes that subclass Django.db.models.Model.

```
from django.db import models
from django.contrib.auth.models import User

gender_list = (('Male', 'Male'), ('Female', 'Female'))

class Teacher(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    gender = models.CharField(max_length=10, choices=gender_list)
    room = models.CharField(max_length=5)
    is_deleted = models.BooleanField(default=False)
```

Figure 6 Teacher/Student Model

The teacher model contains fields like a user, gender, room, and is_deleted. The user field has a one-to-one relation with the User class, which is imported from the Django library as shown in figure 6.

```
class Student(models.Model):
    roll_no = models.IntegerField(unique=True, help_text='Roll No of the student')
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    gender = models.CharField(max_length=20, choices=choices.gender_list)
    room = models.CharField(max_length=5, choices=choices.room_list)
    image_path = models.CharField(max_length=200)
    is_deleted = models.BooleanField(default=False)
    #image field

    def __str__(self):
        return str(self.roll_no)
```

Figure 7 Student model

This is a model to record the student information. The student model contains fields such as first_name, last_name, and gender, as shown in figure 7. In the image_path field, images of a particular student are stored as the dataset. These images are used to compare the detected face during facial recognition.

```
class Attendance(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    teacher = models.ForeignKey(User, on_delete=models.CASCADE)
    date = models.DateField(blank=True)
    time = models.TimeField(blank=True)
    attended = models.BooleanField(default=False)

    def __str__(self):
        return str(self.attended)
```

Figure 8 Attendance model

The attendance model keeps the attendance record. It has student and teacher field, which has a foreign key attribute with the Student model and Teacher model, respectively. It has a date, time, and attended fields to keep the date and time record of the attendance.

4.3.3 Interface Design

The user interface in Django was created by implementing Django templates. It is the primary tool to create a user interface in Django. Django templates create HTML interface, which is rendered by Django view. In Django, views can be created as a class and a function. View created by function is called function-based view and view created by class is called a class-based view.

```
def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)

            if request.user.is_superuser and request.user.is_authenticated:
                return redirect('admins:home')
            else:
                return redirect('teacher:home')

    form = AuthenticationForm()
    return render(request, 'home.html', {'form': form})

def signout(request):
    if request.method == 'POST':
        logout(request)
        return redirect('home')
```

Figure 8 Function-based view for Login and Logout.

Figure 8 shows an example of function-based views. It has two functions login_view and signout view. In login_view, if the user is an admin and authenticated then, it redirects the user to the admin's home page. If the user is a teacher and authenticated then, the user is redirected to the teacher's home page. The user is redirected to the application's home page when signed out.


```
{% extends 'base.html' %}
{% block title %} Admin Login {% endblock %}

{% block content %}
    {% load widget_tweaks %}
    <h1 class="text-center header">Admin Login</h1>

    <div class="row">
        <div class="col-md-3"></div>
        <div class="col-md-6">
            <form action="" method="POST" autocomplete="off">
                {% csrf_token %}
                <div class="form-group">
                    <label for="">Username : </label>
                    {{ form.username | add_class:"form-control" }}
                </div>

                <div class="form-group">
                    <label for="">Password : </label>
                    {{ form.password | add_class:"form-control" }}
                </div>

                <div class="form-group">
                    <input type="submit" class="btn btn-block btn-success" value="Login">
                </div>
            </form>
        </div>
        <div class="col-md-3"></div>
    </div>
{% endblock %}
```

Figure 9 Admin login template HTML

Figure 9 shows an HTML template file for login. This HTML file would be rendered by Django views.

Attendance System
Help

Face Recognition Based Attendance System

19-June-2022 | 20:57:32

For Already Registered

Take Attendance

Attendance

ID	NAME	DATE	TIME

Quit

For New Registrations

Enter ID Clear

Enter Name Clear

1)Take Images >>> 2)Save Profile

Take Images

Save Profile

Total Registrations till now : 4

The form is titled "For New Registrations" in a green header bar. It has a light blue background. The first section is labeled "Enter ID" and contains a white text input field followed by a red "Clear" button. The second section is labeled "Enter Name" and contains a white text input field followed by a red "Clear" button. Below these is a text label "1)Take Images >>> 2)Save Profile". This is followed by two large blue buttons with white text: "Take Images" and "Save Profile". At the bottom, it displays "Total Registrations till now : 4".

For New Registrations

Enter ID

Clear

Enter Name

Clear

1)Take Images >>> 2)Save Profile

Take Images

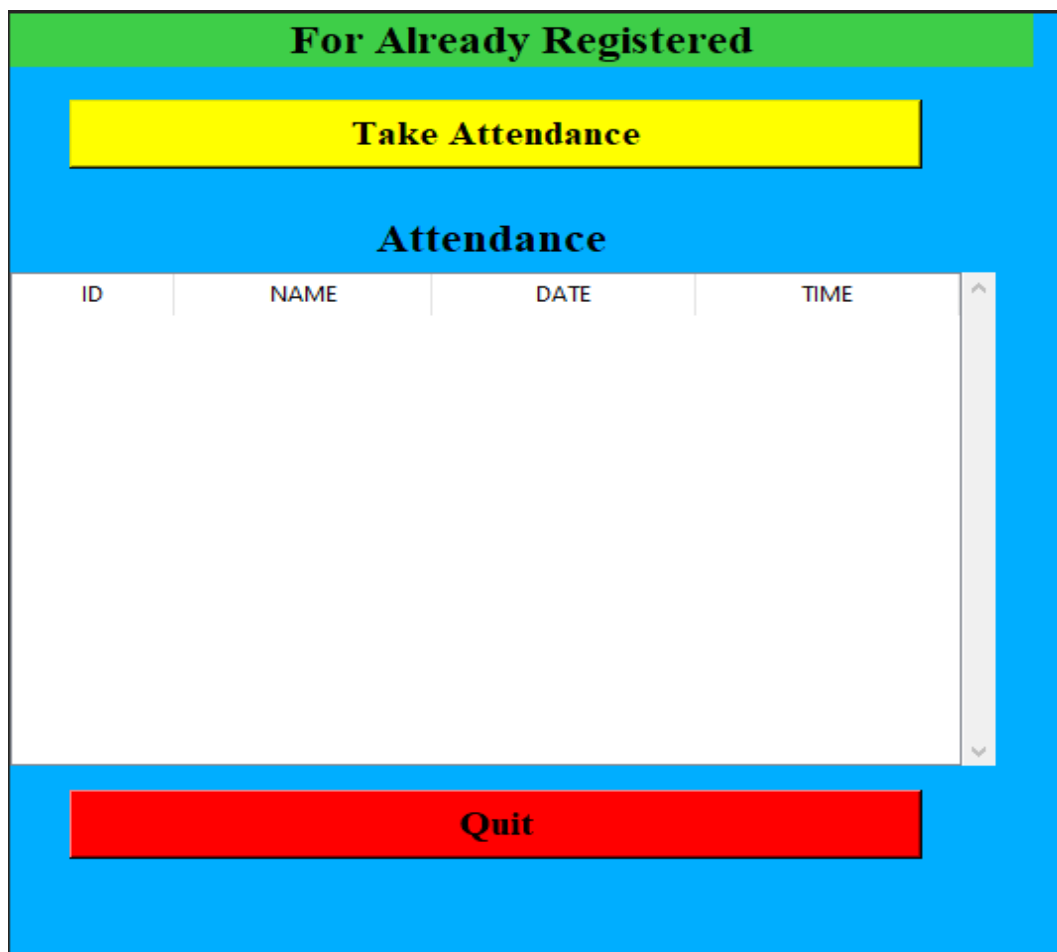
Save Profile

Total Registrations till now : 4

Figure 12 Add student by the admin.

In figure 12, we can see the form that the admin needs to fill to add a teacher to the system. Teacher information like their first name, last name, email, username, and password should be filled in. The teacher is also assigned a class from here.

Similarly, as shown in figure 12, the admin can also add students to the system. The admin needs to provide student's information. In the image path of the data set images of a student is provided. The scanned image is compared with the images stored in the data set from this path.



The screenshot displays a web application interface for a teacher. At the top, a green header bar contains the text "For Already Registered". Below this, a yellow button labeled "Take Attendance" is centered. The main content area has a blue background and features the word "Attendance" in bold black text. Below the title is a table with four columns: "ID", "NAME", "DATE", and "TIME". The table is currently empty. To the right of the table is a vertical scrollbar. At the bottom of the interface, a red button labeled "Quit" is centered.

ID	NAME	DATE	TIME
----	------	------	------

Figure 14 Teacher Interface

Figure 14 shows the teacher interface after the teacher has logged into the system. Here teacher can see the previous attendance record of the students. The teacher can also take new attendance of the students. The teacher can see the student information and their attendance report.

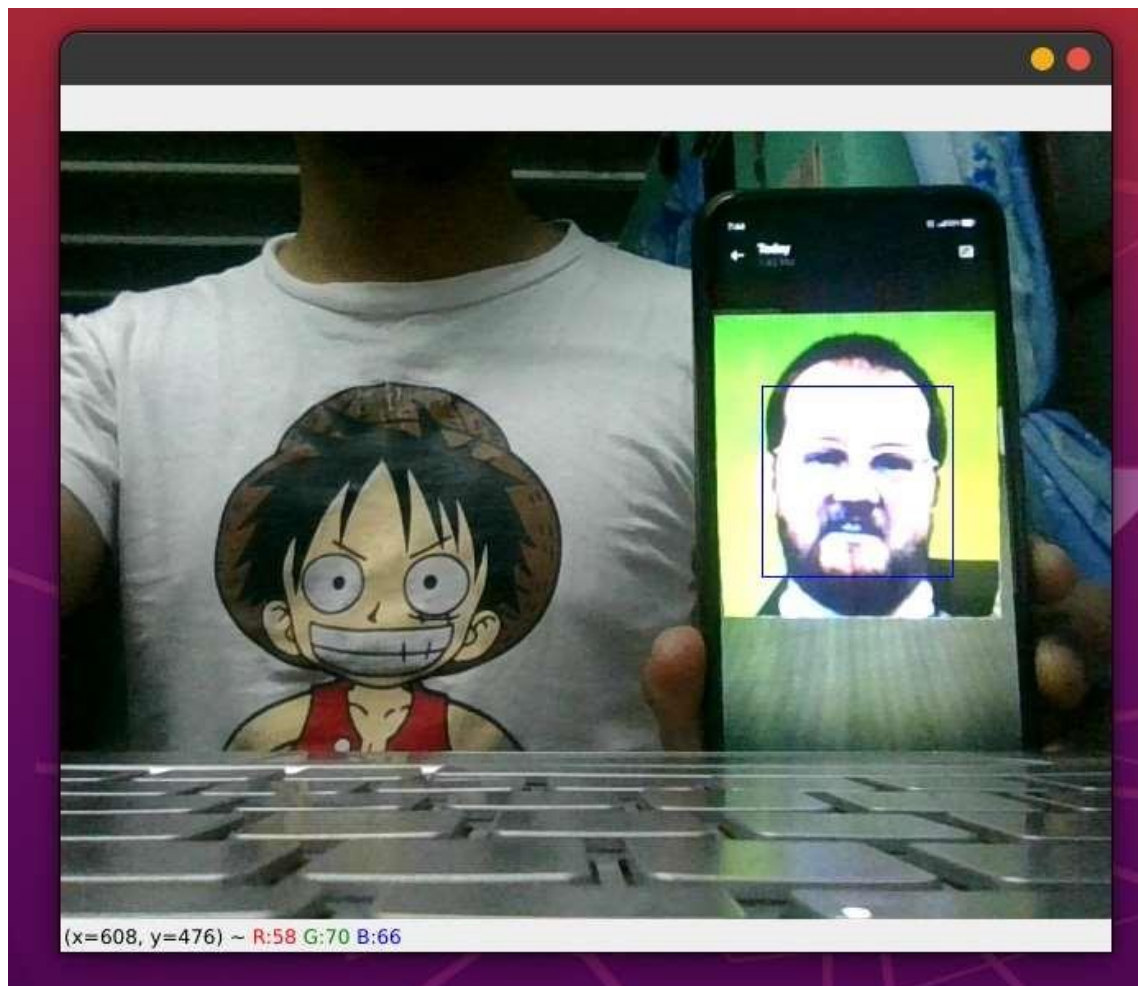


Figure 15 Face Scan.

Figure 15 shows the camera windows when taking new attendance. When the teacher takes new attendance, the camera windows open to scan the face. Once the camera detects the face, it covers the face with a blue square. Once the face has been scanned, it is compared with the images stored in the dataset. If it matches with the images in the dataset, it marks the students as present. The student report can be seen on the teacher's main page as shown in figure 14.

5 Result and Discussion

As this was a small-scale project, data structure and implementation did not have many problems. However, it took the author many effort with research and study with different technologies needed as these tools and technologies were new to the author. This caused a delay in the development of the project.

Despite the delay and difficulties, the author was able to incorporate those tools and technologies and complete the project. However, the success rate of facial recognition was not as expected. The success rate depended upon the quality of the camera, lighting, and sufficient dataset in the database. When these factors were to be managed properly, the success rate of face recognition increased.

The effort that went to learn and research about LBPH and Django and other tools and technologies was worth it. While the process of researching and implementing was overwhelming, it started to be interesting as the project started to show some results. This project gave the author first-hand experience in working on a project using Django and found Django easier and more scalable.

6 Face Detection

The problem of face recognition is all about face detection. This is a fact that seems quite bizarre to new researchers in this area. However, before face recognition is possible, one must be able to reliably find a face and its landmarks. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step.

There are two types of face detection problems:

- 1) Face detection in images and
- 2) Real-time face detection

6.1 FACE DETECTION IN IMAGES

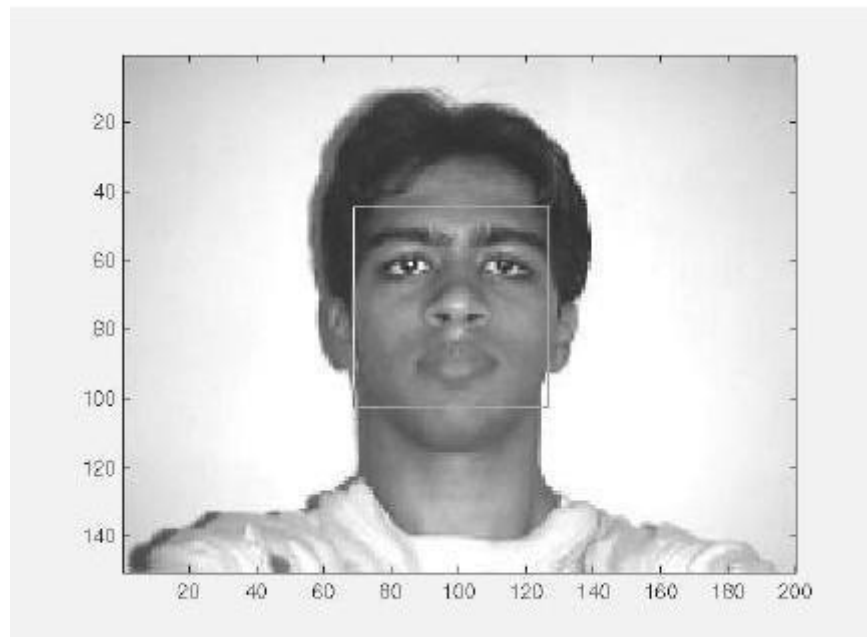


Figure 6.1 A successful face detection in an image with a frontal view of a human face.

Most face detection systems attempt to extract a fraction of the whole face, thereby eliminating most of the background and other areas of an individual's head such as hair that are not necessary for the face recognition task. With static images, this is often done by running a across the image. The face detection system then judges if a face is present inside the window (Brunelli and Poggio, 1993). Unfortunately, with static images there is a very large search space of possible locations of a face in an image.

Most face detection systems use an example based learning approach to decide whether or not a face is present in the *window* at that given instant (Sung and Poggio, 1994 and Sung, 1995). A neural network or some other classifier is trained using supervised learning with 'face' and 'non-face' examples, thereby enabling it to classify an image (*window* in face detection system) as a 'face' or 'non-face'. Unfortunately, while it is relatively easy to find face examples, how would one find a representative sample of images which represent non-faces (Rowley et al., 1996)? Therefore, face detection systems using example based learning need thousands of 'face' and 'non-face' images for effective training. Rowley, Baluja, and Kanade (Rowley et al., 1996) used 1025 face images and 8000 non-face images (generated from 146,212,178 sub-images) for their training set!

There is another technique for determining whether there is a face inside the face detection system's *window* - using Template Matching. The difference between a fixed target pattern (face) and the window is computed and thresholded. If the window contains a pattern which is close to the target pattern (face) then the window is judged as containing a face. An implementation of template matching called Correlation Templates uses a whole bank of fixed sized templates to detect facial

features in an image (Bichsel, 1991 & Brunelli and Poggio, 1993). By using several templates of different (fixed) sizes, faces of different scales (sizes) are detected. The other implementation of template matching is using a deformable template (Yuille, 1992). Instead of using several fixed size templates, we use a deformable template (which is non-rigid) and there by change the size of the template hoping to detect a face in an image.

A face detection scheme that is related to template matching is image invariants. Here the fact that the local ordinal structure of brightness distribution of a face remains largely unchanged under different illumination conditions (Sinha, 1994) is used to construct a spatial template of the face which closely corresponds to facial features. In other words, the average grey-scale intensities in human faces are used as a basis for face detection. For example, almost always an individuals eye region is darker than his forehead or nose. Therefore an image will match the template if it satisfies the 'darker than' and 'brighter than' relationships (Sung and Poggio, 1994).

6.2 REAL-TIME FACE DETECTION

Real-time face detection involves detection of a face from a series of frames from a video-capturing device. While the hardware requirements for such a system are far more stringent, from a computer vision stand point, real-time face detection is actually a far simpler process than detecting a face in a static image. This is because unlike most of our surrounding environment, people are continually moving. We walk around, blink, fidget, wave our hands about, etc.



Figure 6.2.1: Frame 1 from camera



Figure 6.2.2: Frame 2 from camera



Figure 6.2.3: Spatio-Temporally filtered image

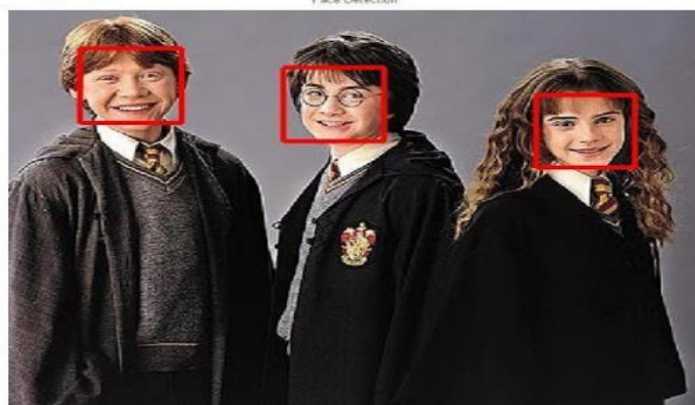
Since in real-time face detection, the system is presented with a series of frames in which to detect a face, by using spatio-temporal filtering (finding the difference between subsequent frames), the area of the frame that has changed can be identified and the individual detected (Wang and Adelson, 1994 and Adelson and Bergen 1986). Further more as seen in Figure exact face locations can be easily identified by using a few simple rules, such as,

- 1) the head is the small blob above a larger blob -the body
- 2) head motion must be reasonably slow and contiguous -heads won't jump around erratically (Turk and Pentland 1991a, 1991b).

Real-time face detection has therefore become a relatively simple problem and is possible even in unstructured and uncontrolled environments using these very simple image processing techniques and reasoning rules.

6.3 FACE DETECTION PROCESS

Fig 5.3 Face detection



It is process of identifying different parts of human faces like eyes, nose, mouth, etc... this process can be achieved by using MATLAB code. In this project the author will attempt to detect faces in still images by using image invariants. To do this it would be useful to study the grey-scale intensity distribution of an average human face. The following 'average human face' was constructed from a sample of 30 frontal view human faces, of which 12 were from females and 18 from males. A suitably scaled colormap has been used to highlight grey-scale intensity differences.

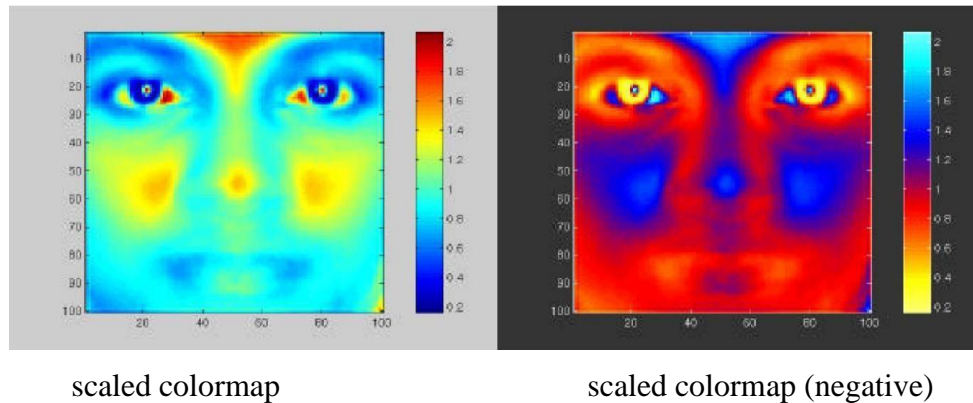


Figure 6.3.1 Average human face in grey-scale

The grey-scale differences, which are invariant across all the sample faces are strikingly apparent. The eye-eyebrow area seem to always contain dark intensity (low) gray-levels while nose forehead and cheeks contain bright intensity (high) gray-levels. After a great deal of experimentation, the researcher found that the following areas of the human face were suitable for a face detection system based on image invariants and a deformable template.

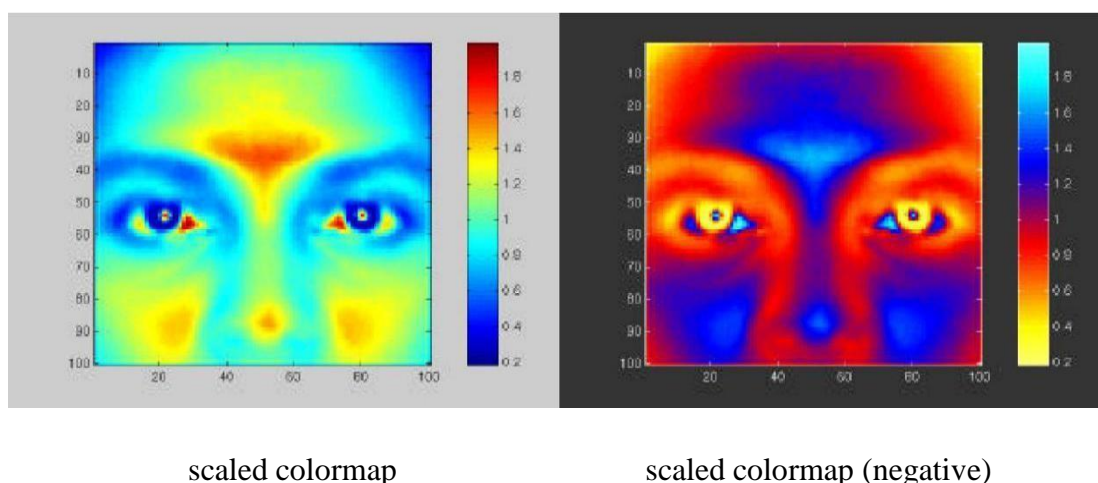


Figure 6.3.2 Area chosen for face detection (indicated on average human face in gray scale)

The above facial area performs well as a basis for a face template, probably because of the clear divisions of the bright intensity invariant area by the dark intensity invariant regions. Once this pixel area is located by the face detection system, any particular area required can be segmented based on the proportions of the average human face. After studying the above images it was subjectively decided by the author to use the following as a basis for dark intensity sensitive and bright intensity sensitive templates. Once these are located in a subject's face, a pixel area 33.3% (of the width of the square window) below this.

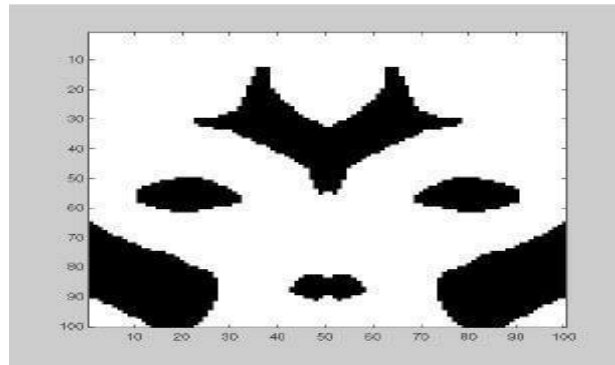


Figure 6.3.3: Basis for a bright intensity invariant sensitive template.

Note the slight differences which were made to the bright intensity invariant sensitive template which were needed because of the pre-processing done by the system to overcome irregular lighting (chapter six). Now that a suitable dark and bright intensity invariant templates have been decided on, it is necessary to find a way of using these to make 2 A-units for a perceptron, i.e. a computational model is needed to assign neurons to the distributions displayed .



Fig 6.3.4 Scanned image detection

- San window over image
- Clasify window as either

6.3.1 Face

6.3.2 Non - Face

6.4 FACE DETECTION ALGORITHM

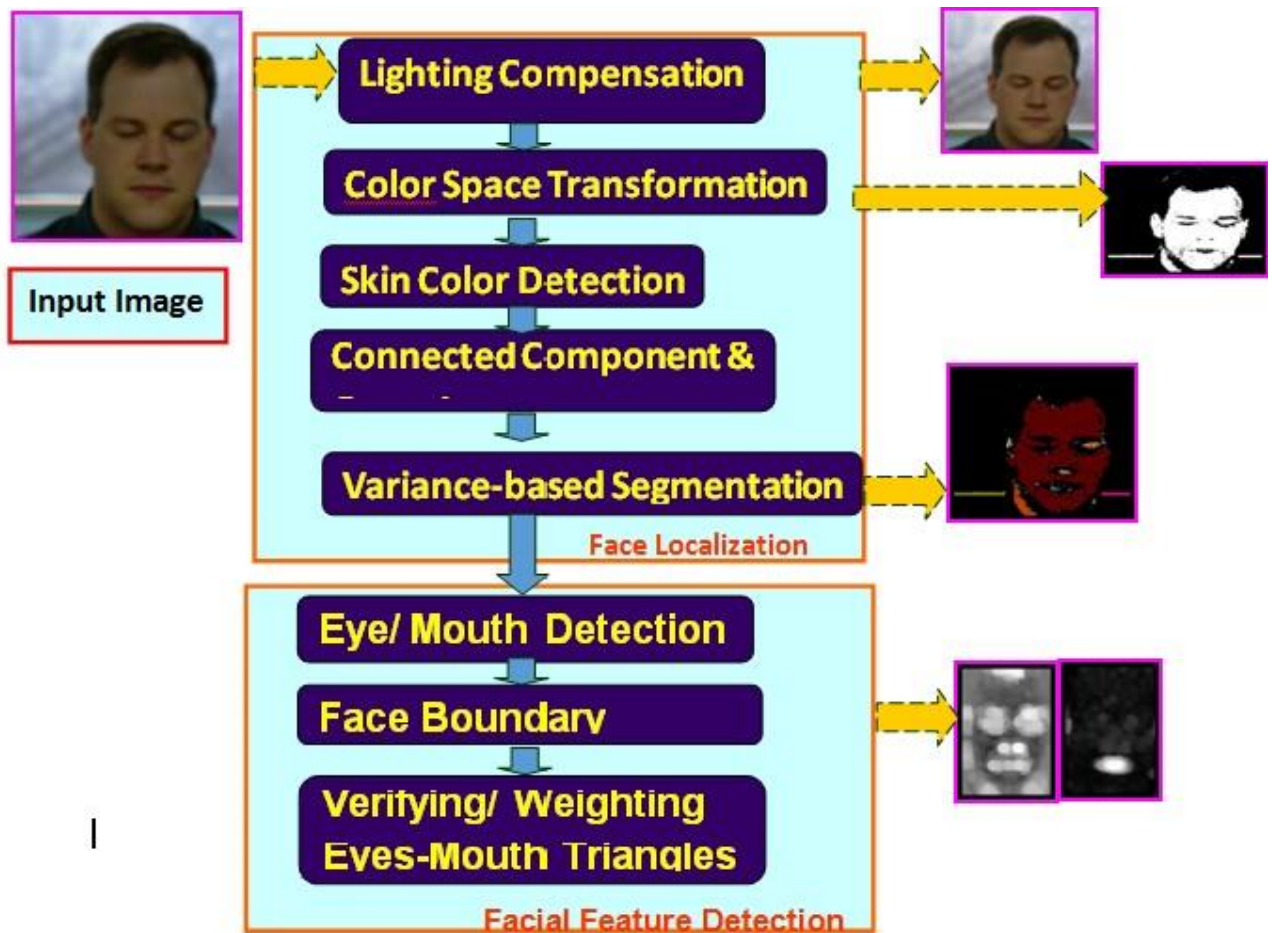


Fig 6.4 Face detection algorithm



Fig 6.4.1 mouth detection



Fig 6.4.2 Noise detection

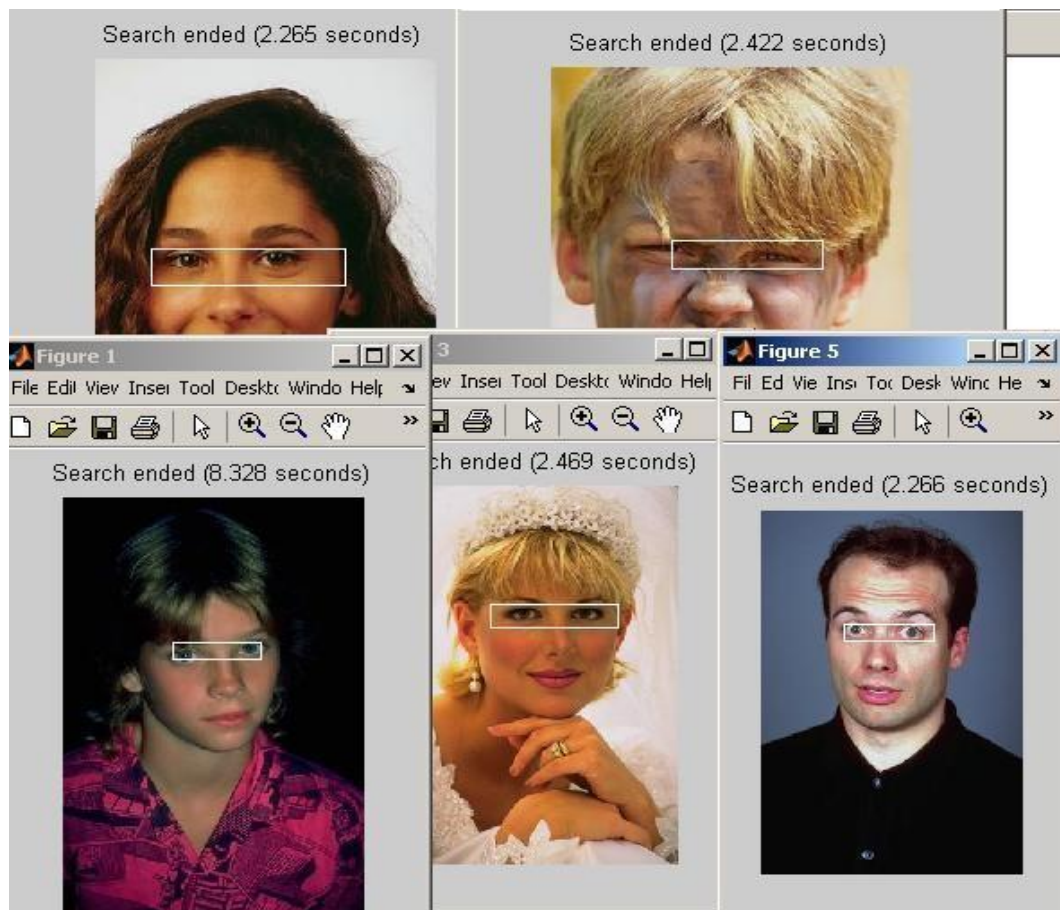


Fig 6.4.3 Eye detection

7 FACE RECOGNITION

Over the last few decades many techniques have been proposed for face recognition. Many of the techniques proposed during the early stages of computer vision cannot be considered successful, but almost all of the recent approaches to the face recognition problem have been creditable. According to the research by Brunelli and Poggio (1993) all approaches to human face recognition can be divided into two strategies:

- (1) Geometrical features and
- (2) Template matching.

7.1 FACE RECOGNITION USING GEOMETRICAL FEATURES

This technique involves computation of a set of geometrical features such as nose width and length, mouth position and chin shape, etc. from the picture of the face we want to recognize. This set of features is then matched with the features of known individuals. A suitable metric such as Euclidean distance (finding the closest vector) can be used to find the closest match. Most pioneering work in face recognition was done using geometric features (Kanade, 1973), although Craw et al. (1987) did relatively recent work in this area.

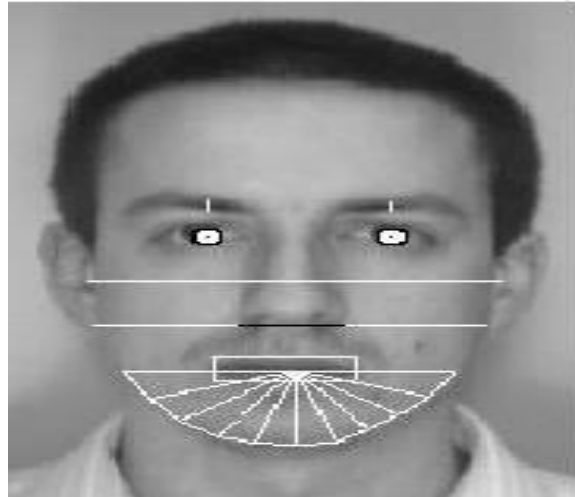


Figure 7.1 Geometrical features (white) which could be used for face recognition

The advantage of using geometrical features as a basis for face recognition is that recognition is possible even at very low resolutions and with noisy images (images with many disorderly pixel intensities). Although the face cannot be viewed in detail its overall geometrical configuration can be extracted for face recognition. The technique's main disadvantage is that automated extraction of the facial geometrical features is very hard. Automated geometrical feature extraction based recognition is also very sensitive to the scaling and rotation of a face in the image plane (Brunelli and Poggio, 1993). This is apparent when we examine Kanade's (1973) results where he reported a recognition rate of between 45-75 % with a database of only 20 people. However if these features are extracted manually as in Goldstein et al. (1971), and Kaya and Kobayashi (1972) satisfactory results may be obtained.

7.1.1 Face recognition using template matching

This is similar the template matching technique used in face detection, except here we are not trying to classify an image as a 'face' or 'non-face' but are trying to recognize a face.

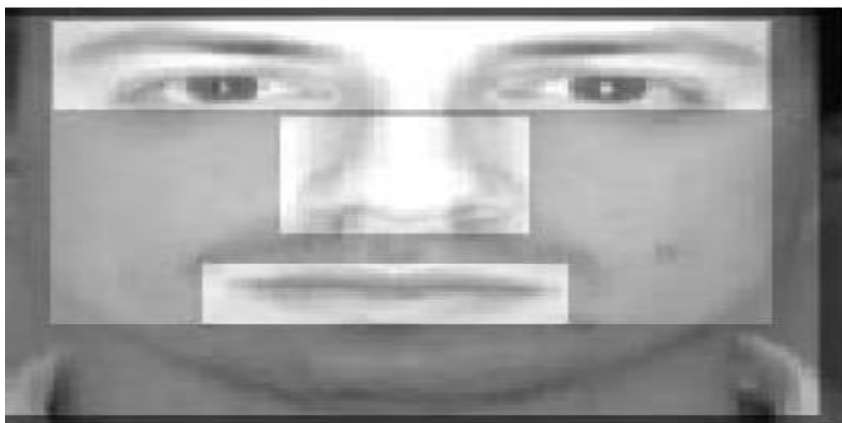


Figure .7.11 Face recognition using template matching

Whole face, eyes, nose and mouth regions which could be used in a template matching

strategy. The basis of the template matching strategy is to extract whole facial regions (matrix of pixels) and compare these with the stored images of known individuals. Once again Euclidean distance can be used to find the closest match. The simple technique of comparing grey-scale intensity values for face recognition was used by Baron (1981). However there are far more sophisticated methods of template matching for face recognition. These involve extensive pre-processing and transformation of the extracted grey-level intensity values. For example, Turk and Pentland (1991a) used Principal Component Analysis, sometimes known as the eigenfaces approach, to pre-process the gray-levels and Wiskott et al. (1997) used Elastic Graphs encoded using Gabor filters to pre-process the extracted regions. An investigation of geometrical features versus template matching for face recognition by Brunelli and Poggio (1993) came to the conclusion that although a feature based strategy may offer higher recognition speed and smaller memory requirements, template based techniques offer superior recognition accuracy.

7.2 PROBLEM SCOP AND SYSTEM SPECIFICATION

The following problem scope for this project was arrived at after reviewing the literature on face detection and face recognition, and determining possible real-world situations where such systems would be of use. The following system(s) requirements were identified

- 1 A system to detect frontal view faces in static images
- 2 A system to recognize a given frontal view face
- 3 Only expressionless, frontal view faces will be presented to the face detection&recognition
- 4 All implemented systems must display a high degree of lighting invariency.
- 5 All systems must posses near real-time performance.
- 6 Both fully automated and manual face detection must be supported
- 7 Frontal view face recognition will be realised using only a single known image
- 8 Automated face detection and recognition systems should be combined into a fully automated face detection and recognition system. The face recognition sub-system must display a slight degree of invariency to scaling and rotation errors in the segmented image extracted by the face detection sub-system.
- 9 The frontal view face recognition system should be extended to a pose invariant face recognition system.

Unfortunately although we may specify constricting conditions to our problem domain, it may not be possible to strictly adhere to these conditions when implementing a system in the real-world.

7.3 BRIEF OUT LINE OF THE IMPLEMENTED SYSTEM

Fully automated face detection of frontal view faces is implemented using a deformable template algorithm relying on the image invariants of human faces. This was chosen because a similar neural-network based face detection model would have needed far too much training data to be implemented and would have used a great deal of computing time. The main difficulties in implementing a deformable template based technique were the creation of the bright and dark intensity sensitive templates and designing an efficient implementation of the detection algorithm.

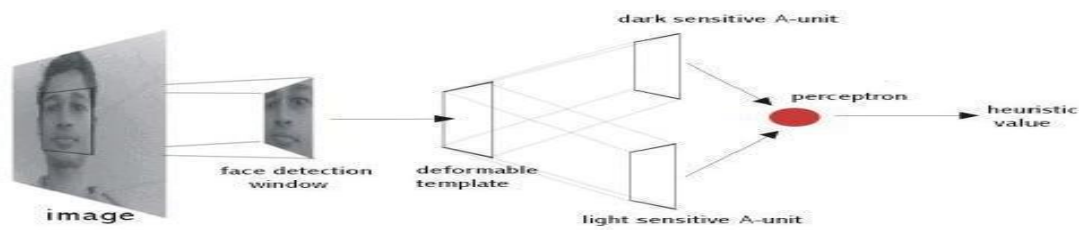


Figure 7.3 Implemented fully automated frontal view face detection model

A manual face detection system was realised by measuring the facial proportions of the average face, calculated from 30 test subjects. To detect a face, a human operator would identify the locations of the subject's eyes in an image and using the proportions of the average face, the system would segment an area from the image

A template matching based technique was implemented for face recognition. This was because of its increased recognition accuracy when compared to geometrical features based techniques and the fact that an automated geometrical features based technique would have required complex feature detection pre-processing.

Of the many possible template matching techniques, Principal Component Analysis was chosen because it has proved to be a highly robust in pattern recognition tasks and because it is relatively simple to implement. The author would also liked to have implemented a technique based on Elastic Graphs but could not find sufficient literature about the model to implement such a system during the limited time available for this project.

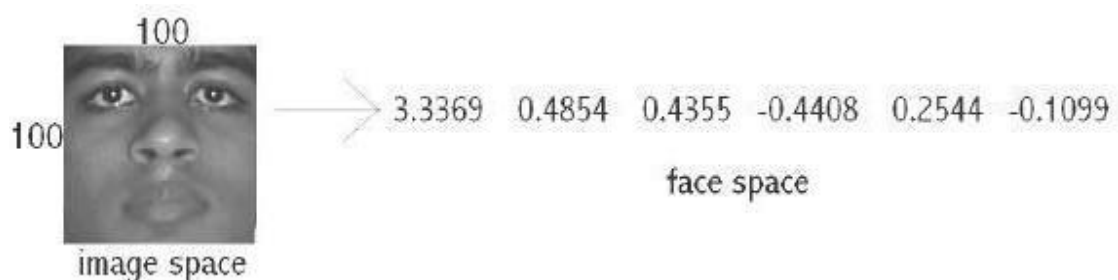


Figure 7.3.1: Principal Component Analysis transform from 'image space' to 'face space'.

Using Principal Component Analysis, the segmented frontal view face image is transformed from what is sometimes called 'image space' to 'face space'. All faces in the face database are transformed into face space. Then face recognition is achieved by transforming any given test image into face space and comparing it with the training set vectors. The closest matching training set vector should belong to the same individual as the test image. Principal Component Analysis is of special interest because the transformation to face space is based on the variation of human faces (in the training set). The values of the 'face space' vector correspond to the amount certain 'variations' are present in the test image.

Face recognition and detection system is a pattern recognition approach for personal identification purposes in addition to other biometric approaches such as fingerprint recognition, signature, retina and so forth. Face is the most common biometric used by humans applications ranges from static, mug-shot verification in a cluttered background.



Fig 7.3.2 Face Recognition

7.4 FACE RECOGNITION DIFFICULTIES

1. Identify similar faces (inter-class similarity)
2. Accommodate intra-class variability due to
 - 2.1 head pose
 - 2.2 illumination conditions
 - 2.3 expressions

2.4 facial accessories

2.5 aging effects

3. Cartoon faces

7.4.1 Inter - class similarity:

- Different persons may have very similar appearance



Fig 7.4.1 Face recognition twins and other and Son

Face recognition and detection system is a pattern recognition approach for personal identification purposes in addition to other biometric approaches such as fingerprint recognition, signature, retina and so forth. The variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

7.4.2 Inter – class variability

Faces with intra-subject variations in pose, illumination, expression, accessories, color, occlusions, and brightness.

7.5 PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (or Karhunen-Loeve expansion) is a suitable strategy for face recognition because it identifies variability between human faces, which may not be immediately obvious. Principal Component Analysis (hereafter PCA) does not attempt to categorise faces using familiar geometrical differences, such as nose length or eyebrow width. Instead, a set of human faces is analysed using PCA to determine which 'variables' account for the variance of faces. In face recognition, these variables are called eigen faces because when plotted they display an eerie resemblance to human faces. Although PCA is used extensively in statistical analysis, the pattern recognition community started to use PCA for classification only relatively recently. As described by Johnson and Wichern (1992), 'principal component analysis is concerned with explaining the variance- covariance structure through a few linear combinations of the original variables.' Perhaps PCA's greatest strengths are in its ability for data reduction and interpretation. For example a 100x100 pixel area containing a face can be very accurately represented by just 40 eigen values. Each eigen value describes the magnitude of each eigen face in each image. Furthermore, all interpretation (i.e. recognition) operations can now be done using just the 40 eigen values to represent a face instead of the manipulating the 10000 values contained in a 100x100 image. Not only is this computationally less demanding but the fact that the recognition information of several thousand.

7.6 UNDERSTANDING EIGENFACES

Any grey scale face image $I(x,y)$ consisting of a $N \times N$ array of intensity values may also be considered as a vector of N^2 . For example, a typical 100x100 image used in this thesis will have to be transformed into a 10000 dimension vector!

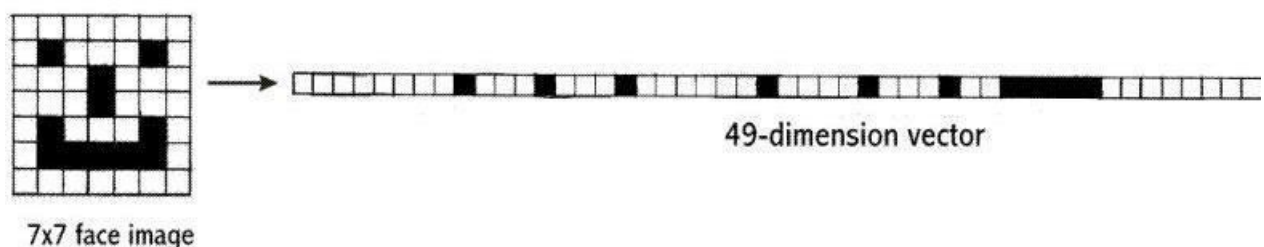


Figure 7.6.0 A 7x7 face image transformed into a 49 dimension vector

This vector can also be regarded as a point in 10000 dimension space. Therefore, all the images of subjects' whose faces are to be recognized can be regarded as points in 10000 dimension space. Face recognition using these images is doomed to failure because all human face images are quite similar to one another so all associated vectors are very close to each other in the 10000-dimension space.

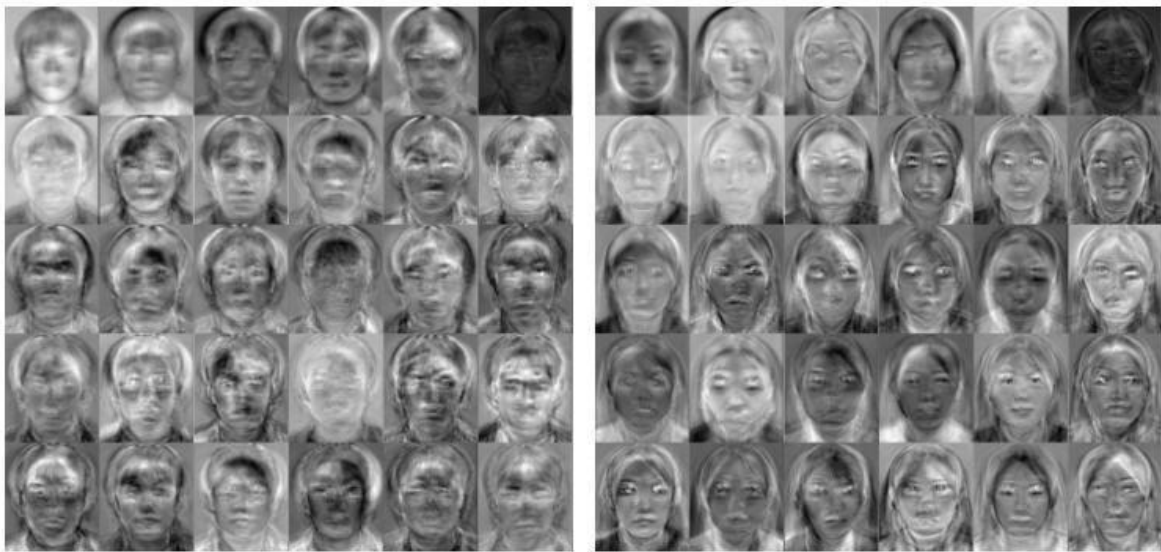
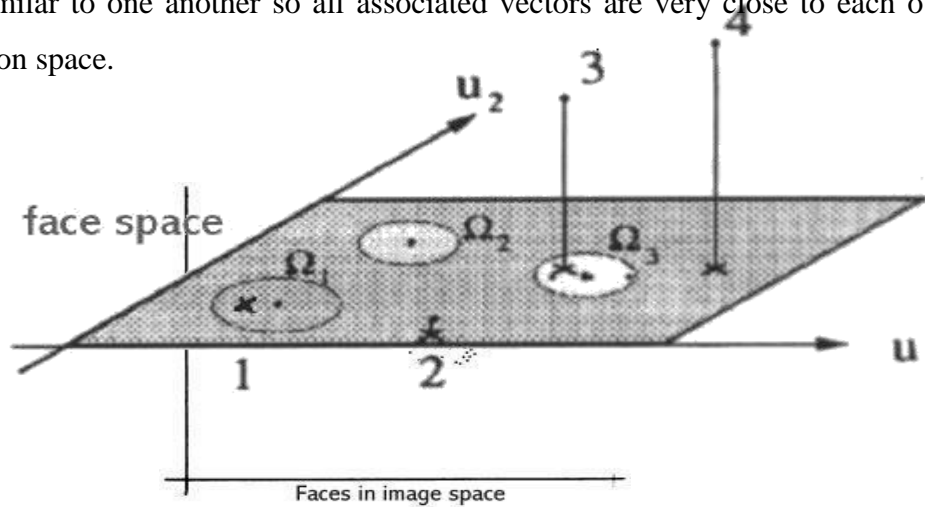


Fig 7.6.1 Eigenfaces

The transformation of a face from image space (I) to face space (f) involves just a simple matrix multiplication. If the average face image is A and U contains the (previously calculated) eigenfaces,

$$f = U * (I - A)$$

This is done to all the face images in the face database (database with known faces) and to the image (face of the subject) which must be recognized. The possible results when projecting a face into face space are given in the following figure.

There are four possibilities:

1. Projected image is a face and is transformed near a face in the face database
2. Projected image is a face and is not transformed near a face in the face database
3. Projected image is not a face and is transformed near a face in the face database
4. Projected image is not a face and is not transformed near a face in the face database

While it is possible to find the closest known face to the transformed image face by calculating the Euclidean distance to the other vectors, how does one know whether the image that is being transformed actually contains a face? Since PCA is a many-to-one transform, several vectors in the image space (images) will map to a point in face space (the problem is that even non-face images may transform near a known face image's faces space vector). Turk and Pentland (1991a), described a simple way of checking whether an image is actually of a face. This is by transforming an image into face space and then transforming it back (reconstructing) into image space. Using the previous notation,

$$I' = U^T * U * (I - A)$$

With these calculations it is possible to verify that an image is of a face and recognise that face. O'Toole et al. (1993) did some interesting work on the importance of eigen faces with large and small eigenvalues. They showed that the eigen vectors with larger eigenvalues convey information relative to the basic shape and structure of the faces. This kind of information is most useful in categorising faces according to sex, race etc. Eigen vectors with smaller eigenvalues tend to capture information that is specific to single or small subsets of learned faces and are useful for distinguishing a particular face from any other face. Turk and Pentland (1991a) showed that about 40 eigen faces were sufficient for a very good description of human faces since the reconstructed image have only about 2% RMS. pixel-by-pixel errors.

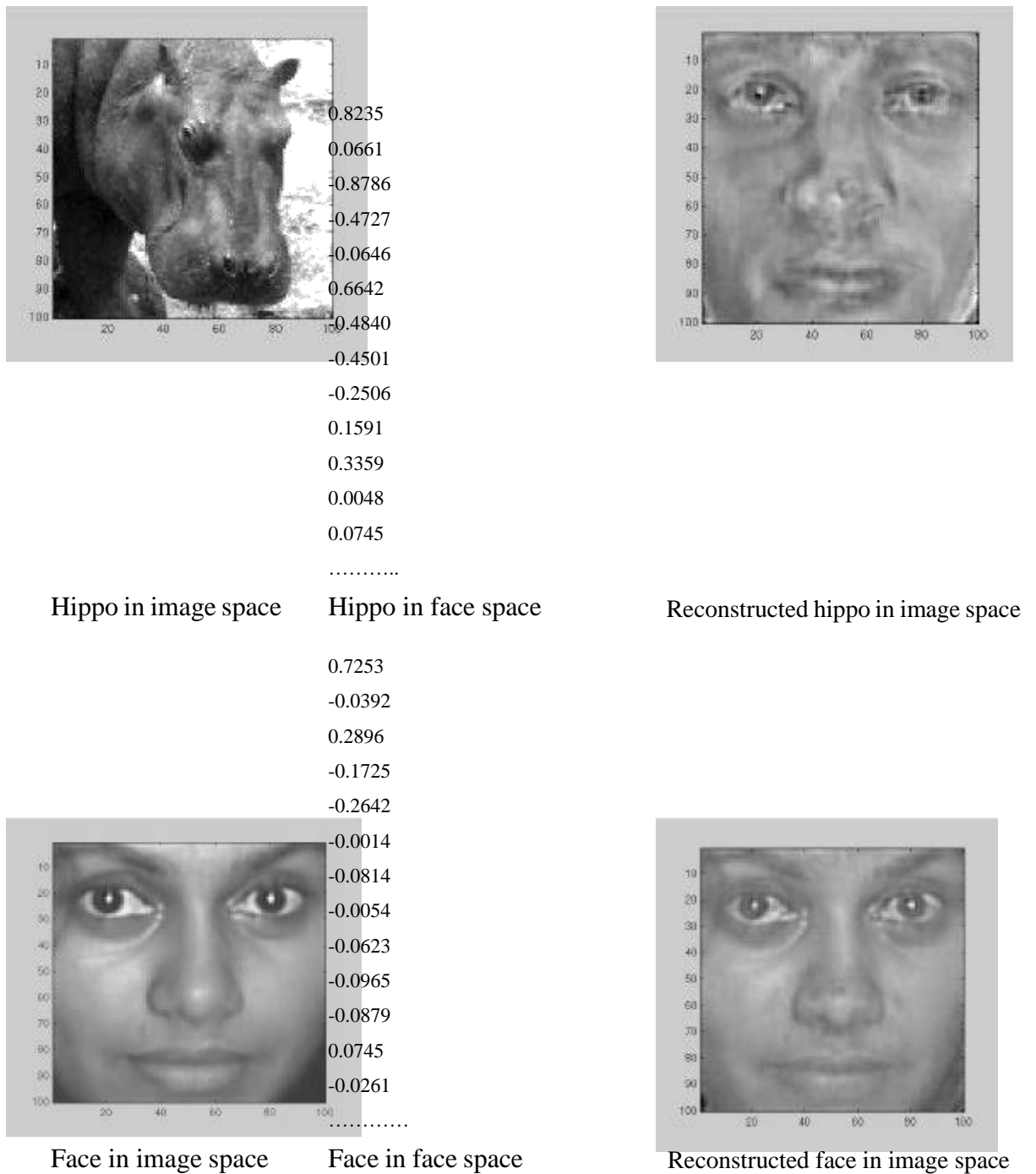
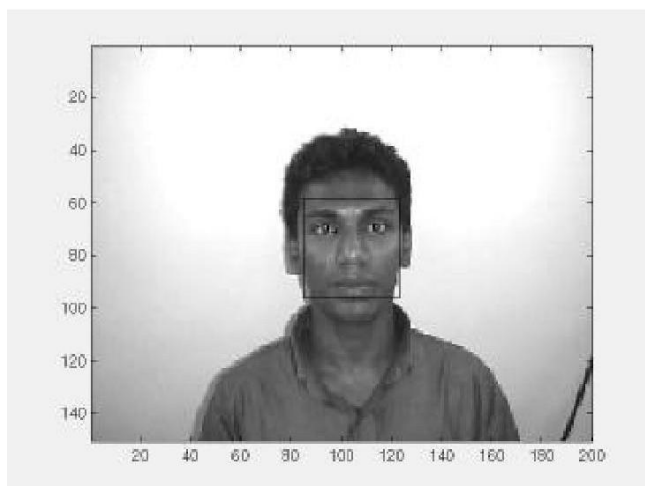
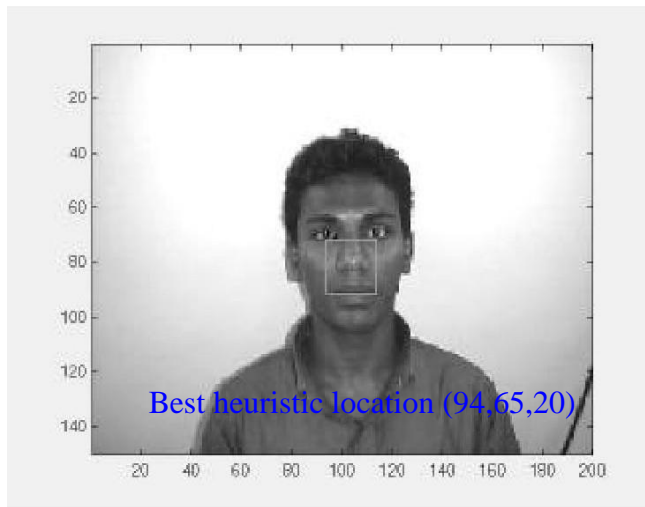


Figure 6.6.3 Images and their reconstruction. The Euclidean distance between a face image and its reconstruction will be lower than that of a non-face image.

7.7 IMPROVING FACE DETECTION USING RECONSTRUCTING

Reconstruction cannot be used as a means of face detection in images in near real-time since it would involve resizing the face detection *window* area and large matrix multiplication, both of which are computationally expensive. However, reconstruction can be used to verify whether potential face locations identified by the deformable template algorithm actually contain a face. If the reconstructed image differs greatly from the face detection *window* then the *window* probably does not contain a face. Instead of just identifying a single potential face location, the face detection algorithm can be modified to output many high 'faceness' locations which can be verified using reconstruction. This is especially useful because occasionally the best 'faceness' location found by the deformable template algorithm may not contain the ideal frontal view face pixel area.



Output from Face detection system

Heuristic	x	y	width
978	74	31	60
1872	74	33	60
1994	75	32	58
2125	76	32	56
2418	76	34	56
2389	79	32	50
2388	80	33	48
2622	81	33	46
2732	82	32	44
2936	84	33	40
2822	85	58	38
2804	86	60	36
2903	86	62	36
3311	89	62	30
3373	91	63	26
3260	92	64	24
3305	93	64	22
3393	94	65	20

potential face locations that have been identified by the face detection system (the best face locations it found on its search) are checked whether they contain a face. If the threshold level (maximum difference between reconstruction and original for the original to be a face) is set correctly this will be an efficient way to detect a face. The deformable template algorithm is fast and can reduce the search space of potential face locations to a handful of positions. These are then checked using reconstruction. The number of locations found by the face detection system can be changed by getting it to output, not just the best face locations it has found so far but any location, which has a 'faceness' value, which for example is, at least 0.9 times the best heuristic value that has been found so far. Then there will be many more potential face locations to be checked using reconstruction. This and similar speed versus accuracy trade-off decisions have to be made keeping in mind the platform on which the system is implemented.

Similarly, instead of using reconstruction to check the face detection system's output, the output's correlation with the average face can be checked. The segmented areas with a high correlation probably contains a face. Once again a threshold value will have to be established to classify faces from non-faces. Similar to reconstruction, resizing the segmented area and calculating its correlation with the average face is far too expensive to be used alone for face detection but is suitable for verifying the output of the face detection system.

7.8 POSE INVARIANT FACE RECOGNITION

Extending the frontal view face recognition system to a pose-invariant recognition system is quite simple if one of the proposed specifications of the face recognition system is relaxed. Successful pose-invariant recognition will be possible if many images of a known individual are in the face database. Nine images from each known individual can be taken as shown below. Then if an image of the same individual is submitted within a 30° angle from the frontal view he or she can be identified.

Nine images in face database from a single known individual



Unknown image from same individual to be identified



Fig: 7.8 Pose invariant face recognition.

Pose invariant face recognition highlights the generalization ability of PCA. For example,

when an individual's frontal view and 30° left view known, even the individual's 15° left view can be recognized

8 Program Code

```
##### IMPORTING #####
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox as mess
import tkinter.simpledialog as tsd
import cv2,os
import csv
import numpy as np
from PIL import Image
import pandas as pd
import datetime
import time

##### FUNCTIONS #####

def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

##### DATE AND TIME #####

def tick():
    time_string = time.strftime('%H:%M:%S')
    clock.config(text=time_string)
```

```

clock.after(200,tick)

##### Help Contact #####

def contact():
    mess._show(title='Contact us', message="Please contact us on : 'jayakumar28062002@gmail.com' ")

##### Classifier File #####

def check_haarcascade():
    exists = os.path.isfile("haarcascade_frontalface_default.xml")
    if exists:
        pass
    else:
        mess._show(title='Some file missing', message='Please contact us for help')
        window.destroy()

##### Save Password #####

def save_pass():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        master.destroy()
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show=*)
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!! Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was registered successfully!!')
            return
    op = (old.get())
    newp= (new.get())
    nnewp = (nnew.get())
    if (op == key):
        if(newp == nnewp):
            txf = open("TrainingImageLabel\psd.txt", "w")
            txf.write(newp)
        else:
            mess._show(title='Error', message='Confirm new password again!!!')
            return
    else:
        mess._show(title='Wrong Password', message='Please enter correct old password.')
        return
    mess._show(title='Password Changed', message='Password changed successfully!!')
    master.destroy()

```

Change Password

```
def change_pass():
    global master
    master = tk.Tk()
    master.geometry("400x160")
    master.resizable(False,False)
    master.title("Change Password")
    master.configure(background="white")
    lbl4 = tk.Label(master,text='  Enter Old Password',bg='white',font=('times', 12, ' bold '))
    lbl4.place(x=10,y=10)
    global old
    old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('times', 12, ' bold '),show=*)
    old.place(x=180,y=10)
    lbl5 = tk.Label(master, text='  Enter New Password', bg='white', font=('times', 12, ' bold '))
    lbl5.place(x=10, y=45)
    global new
    new = tk.Entry(master, width=25, fg="black",relief='solid', font=('times', 12, ' bold '),show=*)
    new.place(x=180, y=45)
    lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('times', 12, ' bold '))
    lbl6.place(x=10, y=80)
    global nnew
    nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('times', 12, ' bold '),show=*)
    nnew.place(x=180, y=80)
    cancel=tk.Button(master,text="Cancel", command=master.destroy ,fg="black" ,bg="red"
,height=1,width=25 , activebackground = "white" ,font=('times', 10, ' bold '))
    cancel.place(x=200, y=120)
    save1 = tk.Button(master, text="Save", command=save_pass, fg="black", bg="#3ece48", height =
1,width=25, activebackground="white", font=('times', 10, ' bold '))
    save1.place(x=10, y=120)
    master.mainloop()
```

Password for Saving Details

```
def psw():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show=*)
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!! Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was registered successfully!!!')
            return
    password = tsd.askstring('Password', 'Enter Password', show=*)
```

```

if (password == key):
    TrainImages()
elif (password == None):
    pass
else:
    mess._show(title='Wrong Password', message='You have entered wrong password')

##### Clear the TextBox #####

def clear():
    txt.delete(0, 'end')
    res = "1)Take Images >>> 2)Save Profile"
    message1.configure(text=res)

def clear2():
    txt2.delete(0, 'end')
    res = "1)Take Images >>> 2)Save Profile"
    message1.configure(text=res)

##### Taking Image #####

def TakeImages():
    check_haarcascade()
    columns = ['SERIAL NO.', 'ID', 'NAME']
    assure_path_exists("StudentDetails/")
    assure_path_exists("TrainingImage/")
    serial = 0
    exists = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists:
        with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
            reader1 = csv.reader(csvFile1)
            for l in reader1:
                serial = serial + 1
            serial = (serial // 2)
            csvFile1.close()
    else:
        with open("StudentDetails\StudentDetails.csv", 'a+') as csvFile1:
            writer = csv.writer(csvFile1)
            writer.writerow(columns)
            serial = 1
            csvFile1.close()
    Id = (txt.get())
    name = (txt2.get())
    if ((name.isalpha()) or (' ' in name)):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        sampleNum = 0
        while (True):

```

```

ret, img = cam.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = detector.detectMultiScale(gray, 1.3, 5)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
    # incrementing sample number
    sampleNum = sampleNum + 1
    # saving the captured face in the dataset folder TrainingImage
    cv2.imwrite("TrainingImage\" + name + "." + str(serial) + "." + Id + '.' + str(sampleNum) +
".jpg",
                gray[y:y + h, x:x + w])
    # display the frame
    cv2.imshow('Taking Images', img)
    # wait for 100 milliseconds
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    # break if the sample number is morethan 100
    elif sampleNum > 100:
        break
cam.release()
cv2.destroyAllWindows()
res = "Images Taken for ID : " + Id
row = [serial, ", Id, ", name]
with open('StudentDetails\\StudentDetails.csv', 'a+') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerow(row)
csvFile.close()
message1.configure(text=res)
else:
    if (name.isalpha() == False):
        res = "Enter Correct name"
        message.configure(text=res)

##### Train Image #####

def TrainImages():
    check_haarcascade()
    assure_path_exists("TrainingImageLabel/")
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, ID = getImagesAndLabels("TrainingImage")
    try:
        recognizer.train(faces, np.array(ID))
    except:
        mess._show(title='No Registrations', message='Please Register someone first!!!')
    return
recognizer.save("TrainingImageLabel\\Trainer.yml")
res = "Profile Saved Successfully"
message1.configure(text=res)

```

```

message.configure(text='Total Registrations till now : ' + str(ID[0]))

##### Image Labels #####

def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    # create empty face list
    faces = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image
        ID = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(ID)
    return faces, Ids

##### Track Image #####

def TrackImages():
    check_haarcascadefile()
    assure_path_exists("Attendance/")
    assure_path_exists("StudentDetails/")
    for k in tv.get_children():
        tv.delete(k)
    msg = "
    i = 0
    j = 0
    recognizer = cv2.face.LBPHFaceRecognizer_create() # cv2.createLBPHFaceRecognizer()
    exists3 = os.path.isfile("TrainingImageLabel\Trainer.yml")
    if exists3:
        recognizer.read("TrainingImageLabel\Trainer.yml")
    else:
        mess._show(title='Data Missing', message='Please click on Save Profile to reset data!!')
        return
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);

    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', ' ', 'Name', ' ', 'Date', ' ', 'Time']
    exists1 = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists1:

```

```

df = pd.read_csv("StudentDetails\StudentDetails.csv")
else:
    mess._show(title='Details Missing', message='Students details are missing, please check!')
    cam.release()
    cv2.destroyAllWindows()
    window.destroy()
while True:
    ret, im = cam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, 1.2, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
        serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
        if (conf < 50):
            ts = time.time()
            date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
            timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
            aa = df.loc[df['SERIAL NO.'] == serial]['NAME'].values
            ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values
            ID = str(ID)
            ID = ID[1:-1]
            bb = str(aa)
            bb = bb[2:-2]
            attendance = [str(ID), ", ", bb, ", ", str(date), ", ", str(timeStamp)]

        else:
            Id = 'Unknown'
            bb = str(Id)
            cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)
    cv2.imshow('Taking Attendance', im)
    if (cv2.waitKey(1) == ord('q')):
        break
ts = time.time()
date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
exists = os.path.isfile("Attendance\Attendance_" + date + ".csv")
if exists:
    with open("Attendance\Attendance_" + date + ".csv", 'a+') as csvFile1:
        writer = csv.writer(csvFile1)
        writer.writerow(attendance)
    csvFile1.close()
else:
    with open("Attendance\Attendance_" + date + ".csv", 'a+') as csvFile1:
        writer = csv.writer(csvFile1)
        writer.writerow(col_names)
        writer.writerow(attendance)
    csvFile1.close()
with open("Attendance\Attendance_" + date + ".csv", 'r') as csvFile1:
    reader1 = csv.reader(csvFile1)
    for lines in reader1:
        i = i + 1

```

```

    if (i > 1):
        if (i % 2 != 0):
            iidd = str(lines[0]) + ' '
            tv.insert(" 0, text=iidd, values=(str(lines[2]), str(lines[4]), str(lines[6])))
csvFile1.close()
cam.release()
cv2.destroyAllWindows()

##### USED STUFFS #####

global key
key = ""

ts = time.time()
date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
day,month,year=date.split("-")

mont={'01':'January',
      '02':'February',
      '03':'March',
      '04':'April',
      '05':'May',
      '06':'June',
      '07':'July',
      '08':'August',
      '09':'September',
      '10':'October',
      '11':'November',
      '12':'December'
      }

##### GUI FRONT-END #####

window = tk.Tk()
window.geometry("1280x720")
window.resizable(True,False)
window.title("Attendance System")
window.configure(background='#262523')
frame1 = tk.Frame(window, bg="#00aeff")
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)
frame2 = tk.Frame(window, bg="#00aeff")
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)
message3 = tk.Label(window, text="Face Recognition Based Attendance System",
fg="white",bg="#262523",width=55,height=1,font=('times', 29, 'bold'))
message3.place(x=10, y=10)
frame3 = tk.Frame(window, bg="#c4c6ce")
frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)
frame4 = tk.Frame(window, bg="#c4c6ce")
frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

```



```

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+" | ", fg="orange",bg="#262523"
,width=55,height=1,font=('times', 22, ' bold '))
datef.pack(fill='both',expand=1)
clock = tk.Label(frame3,fg="orange",bg="#262523",width=55,height=1,font=('times', 22, ' bold '))
clock.pack(fill='both',expand=1)
tick()
head2 = tk.Label(frame2, text="                For New Registrations                ",
fg="black",bg="#3ece48",font=('times', 17, ' bold '))
head2.grid(row=0,column=0)
head1 = tk.Label(frame1, text="                For Already Registered                ",
fg="black",bg="#3ece48",font=('times', 17, ' bold '))
head1.place(x=0,y=0)
lbl = tk.Label(frame2, text="Enter ID",width=20,height=1,fg="black",bg="#00aeff",font=('times', 17, '
bold '))
lbl.place(x=80,y=55)
txt = tk.Entry(frame2,width=32,fg="black",font=('times', 15, ' bold '))
txt.place(x=30,y=88)
lbl2 = tk.Label(frame2, text="Enter Name",width=20,fg="black",bg="#00aeff",font=('times', 17, ' bold
'))
lbl2.place(x=80,y=140)
txt2 = tk.Entry(frame2,width=32,fg="black",font=('times', 15, ' bold '))
txt2.place(x=30,y=173)
message1 = tk.Label(frame2, text="1)Take Images >>> 2)Save Profile",bg="#00aeff",
fg="black",width=39,height=1,activebackground="yellow",font=('times', 15, ' bold '))
message1.place(x=7,y=230)
message = tk.Label(frame2, text="",bg="#00aeff",fg="black",width=39,height=1,activebackground =
"yellow",font=('times', 16, ' bold '))
message.place(x=7,y=450)
lbl3 = tk.Label(frame1, text="Attendance",width=20,fg="black",bg="#00aeff",height=1,font=('times',
17, ' bold '))
lbl3.place(x=100,y=115)
res=0
exists = os.path.isfile("StudentDetails\\StudentDetails.csv")
if exists:
    with open("StudentDetails\\StudentDetails.csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for l in reader1:
            res = res + 1
        res = (res // 2) - 1
        csvFile1.close()
else:
    res = 0
message.configure(text="Total Registrations till now : '+str(res))

##### MENUBAR #####

menubar = tk.Menu(window,relief='ridge')
filemenu = tk.Menu(menubar,tearoff=0)
filemenu.add_command(label='Change Password', command = change_pass)
filemenu.add_command(label='Contact Us', command = contact)

```

```
filemenu.add_command(label='Exit',command = window.destroy)
menubar.add_cascade(label='Help',font=('times', 29, ' bold '),menu=filemenu)
```

TREEVIEW ATTENDANCE TABLE

```
tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))
tv.column('#0',width=82)
tv.column('name',width=130)
tv.column('date',width=133)
tv.column('time',width=133)
tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)
tv.heading('#0',text = 'ID')
tv.heading('name',text = 'NAME')
tv.heading('date',text = 'DATE')
tv.heading('time',text = 'TIME')
```

SCROLLBAR

```
scroll=ttk.Scrollbar(frame1,orient='vertical',command=tv.yview)
scroll.grid(row=2,column=4,padx=(0,100),pady=(150,0),sticky='ns')
tv.configure(yscrollcommand=scroll.set)
```

BUTTONS

```
clearButton = tk.Button(frame2, text="Clear", command=clear ,fg="black" ,bg="#ea2a2a" ,width=11
,activebackground = "white" ,font=('times', 11, ' bold '))
clearButton.place(x=335, y=86)
clearButton2 = tk.Button(frame2, text="Clear", command=clear2 ,fg="black" ,bg="#ea2a2a" ,width=11 ,
activebackground = "white" ,font=('times', 11, ' bold '))
clearButton2.place(x=335, y=172)
takeImg = tk.Button(frame2, text="Take Images",
command=TakeImages ,fg="white" ,bg="blue" ,width=34 ,height=1, activebackground = "white"
,font=('times', 15, ' bold '))
takeImg.place(x=30, y=300)
trainImg = tk.Button(frame2, text="Save Profile", command=psw
,fg="white" ,bg="blue" ,width=34 ,height=1, activebackground = "white" ,font=('times', 15, ' bold '))
trainImg.place(x=30, y=380)
trackImg = tk.Button(frame1, text="Take Attendance",
command=TrackImages ,fg="black" ,bg="yellow" ,width=35 ,height=1, activebackground = "white"
,font=('times', 15, ' bold '))
trackImg.place(x=30,y=50)
quitWindow = tk.Button(frame1, text="Quit",
command=window.destroy ,fg="black" ,bg="red" ,width=35 ,height=1, activebackground = "white"
,font=('times', 15, ' bold '))
quitWindow.place(x=30, y=450)
```

END

```
window.configure(menu=menubar)
window.mainloop()
```

```
#####
```

9 Conclusion

The goal of the project was to build a facial recognition system for student's attendance. Concepts of facial recognition and LBPH is heavily discussed in this thesis. Similarly, web development with Django is also discussed, followed by examples of implementation and explanations.

The result of the project was a successful prototype of a facial recognition system where the admin can create a teacher account and add students and their information to the database. Teachers then can log in to the system and take attendance of the student. The student's face is detected by a camera and attendance is recorded in the database. Teachers and admin could see the attendance report of the students.

The fully automated face detection and recognition system was not robust enough to achieve a high recognition accuracy. The only reason for this was the face recognition subsystem did not display even a slight degree of invariance to scale, rotation or shift errors of the segmented face image.

Overall, the project was successful in its showcasing how LBPH can be implemented in Django to create a web application. Once implemented, it can be used to take attendance of students and keep track of their attendance records. This project has the potential for further development in the future by adding more features for students and teachers. More features such as assignments, results, and grades could be added.

References

1. John D. Woodward, Jr., Christopher Horn, Julius Gatune, and Aryn Thomas "Biometrics: A look at facial Recognition," 2003.
URL: <https://apps.dtic.mil/sti/pdfs/ADA414520.pdf>
2. Li Wang, Ali Akbar Siddique. "Facial recognition system using LBPHface recognizer for anti-theft and surveillance application based on drone technology" , Measurement and Control, 2020
URL: <https://journals.sagepub.com/doi/10.1177/0020294020932344>
3. A brief history of facial recognition. 2020.
URL: <https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/>
4. Mark Andrejevic & Neil Selwyn (2020) "Facial recognition technology inschools: critical questions and concerns, Learning, Media and Technology,"
URL: <https://www.tandfonline.com/doi/full/10.1080/17439884.2020.1686014>
5. Python Documentation [online].
URL: https://www.w3schools.com/python/python_intro.asp
6. Django Documentation [online].
URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
7. OpenCV Documentation
[online].URL:
<https://opencv.org/>

8. Farah Deeba, Aftab Ahmed, Hira Memon "Real Time Face Recognition of Human Faces by using LBPH and Viola Jones Algorithm." 2019. URL: <https://pdfs.semanticscholar.org/3255/0898eec9c4424932e70e5e32c98b0220a747.pdf>
9. Chatterjee, S., Jana, A., Ganguly, A., & Ghosh, A. Automated Attendance System Using Face Recognition Technique. International Journal of Engineering and Applied Sciences (IJEAS), 5(7). URL: <https://doi.org/10.31873/IJEAS.5.7.18>
10. X. Zhao and C. Wei, "A real-time face recognition system based on the improved LBPH algorithm," 2017
URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8124508&isnumber=8124489>
11. Hannan, Farwa Abdul et al "Comparative Analysis of Face Recognition Methodologies and Techniques," 2016.
12. A. K. Kaine, "The impact of facial recognition systems on business practices within an operational setting," Proceedings of the 25th International Conference on Information Technology Interfaces, 2003.
URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1225363>