

FACE RECOGNITION SYSTEM

A PROJECT REPORT

Mini Project

(KCA353)Session

(2023-24)

Submitted by

MANMEET CHAUHAN

(2200290140085)

MANAN SHARMA

(2200290140084)

Submitted in partial fulfilment of the

Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Dr. Amit Kumar Gupta

(Professor)



Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET Group of Institutions, Delhi

NCR. Ghaziabad Uttar Pradesh-

201206

January, 2024

DECLARATION

I hereby declare that the work presented in report entitled “Face Recognition System” was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, that are not my original contribution. I have used quotation marks to identify verbatim sentences and give credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Manmeet Chauhan (2200290140085)

Name: Manan Sharma (2200290140084)

(Candidate Signature)

CERTIFICATE

Certified that **Manmeet Chauhan 2200290140085, Manan Sharma 2200290140084** have carried out the project work having “**Face Recognition System**” (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Manmeet Chauhan 2200290140085

Manan Sharma 2200290140084

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Amit Kumar Gupta

Professor

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Dr. Arun Kumar Tripathi

Head

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

ABSTRACT

With every passing day, we are becoming more and more dependent upon technology to carry out even the most basic of our actions. Facial detection and Facial recognition help us in many ways, be it sorting of photos in our mobile phone gallery by recognizing pictures with their face in them or unlocking a phone by a mere glance to adding biometric information in the form of face images in the country's unique ID database (Aadhaar) as an acceptable biometric input for verification.

This project lays out the basic terminology required to understand the implementation of Face Detection and Face Recognition using Intel's Computer Vision library called 'OpenCV'.

It also shows the practical implementation of the Face Detection and Face Recognition using OpenCV with Python embedding on both Windows as well as macOS platform. The aim of the project is to implement Facial Recognition on faces that the script can be trained for. The input is taken from a webcam and the recognized faces are displayed along with their name in real time.

This project can be implemented on a larger scale to develop a biometric attendance system which can save the time-consuming process of manual attendance system.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr Amit Kumar Gupta** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Manmeet Chauhan
(2200290140085)

Manan Sharma
(2200290140084)

TABLE OF

1	Introduction	1-5
1.1	Overview	1
1.2	Motivation	2
1.3	Problem Statement	2
1.4	Objective And Scope	3
1.5	Feasibility Study	5
2	Literature Review	6-8
3	Design	9-21
3.1	Data Flow Diagram	9-10
3.1.1	Level 0 DFD	10
3.1.2	Level 1 DFD	11
3.2	Data Dictionary	12-13
3.3	Sequence Diagram	13-17
3.3.1	Sequence Diagram of Registration	13
3.3.2	Sequence Diagram of Login Process	14
3.3.3	Sequence Diagram of Add bookings	15
3.3.4	Sequence Diagram of Add Services	16
3.4	Use Case Diagram	17-20
3.4.1	Use Case Description of Registration	17
3.4.2	Use Case Description of Login	18
3.4.3	Use Case Description Of Create Report	19
3.4.4	Use Case Description of Add Services	20
4	Proposed Work	22-28
4.1	Database Design	22
4.2	Class Diagram	24
4.3	Approach Used	25
4.4	Dependencies Required	26

4.5	Algorithms and Flowchart	27
5	Results	28-32
5.1	Screens and Explanation	28-32
6	Discussions	33-36
6.1	Performance	33
6.2	Limitations of the System	33
6.3	Testing Of Systems	34
7	Conclusion	37-38
8	References & Bibliography	39-40
8.1	Online Websites	39
8.2	Reference Books	40

List Of Figures

Figure No.	Name of Figure	Page No.
1.1	Data Flow Diagram	10
2.1	Level 0 DFD	11
2.2	Level 1 DFD	12
3.1	Sequence Diagram of Registration	13
3.2	Sequence Diagram of Login	14
3.3	Sequence Diagram Of Add Bookings	15
3.4	Sequence Diagram of Add Services	17
3.5	Use Case Diagram for Owner	19
3.6	Use Case Diagram for Salon Admin	22
3.7	Use Case Diagram for Customer	24
4.1	Entity relationship diagram (ERD)	28
4.2	Class Diagram	29
4.3	Activity Diagram for System Login	30
4.4	Activity Diagram for Add New Service	31
4.5	Flowchart of Signup	33

CHAPTER 1

INTRODUCTION

1.1 Overview

A face recognition system is a technology that identifies or verifies individuals by analyzing and comparing their facial features. It is a subset of biometric systems that use unique physiological or behavioral characteristics for identification purposes. Face recognition has gained significant popularity and applications in various fields, including security, surveillance, access control, and user authentication.

1.2 Purpose:

Face recognition systems serve various purposes across different domains, providing solutions to numerous practical challenges. There are some of few purpose.

1.2.1 Access Control:

Face recognition is commonly used for access control in secure environments. It replaces traditional methods like key cards or PINs, offering a more convenient and secure means of entry.

1.2.2 Authentication And Authorization:

Face recognition serves as a biometric authentication method for user access to devices, systems, or applications. It enhances security by verifying the identity of individuals based on their unique facial features.

1.2.3 Identity Verification:

Face recognition is used for identity verification in various applications, such as online account access, e-commerce transactions, and financial services. It provides an additional layer of security in digital interactions.

1.2.4 Attendance Tracking:

Educational institutions and workplaces use face recognition for automated attendance tracking. This streamlines the attendance management process and reduces the need for manual record-keeping.

Face recognition is a biometric technology that has gained significant traction in recent years due to its wide range of applications across various industries. It is a method of identifying or verifying individuals based on their unique facial features. The primary objective of face recognition systems is to accurately recognize and distinguish between different faces in images or videos.

This technology finds applications in diverse fields, including security and surveillance, access control, law enforcement, digital authentication, and personalized user experiences. It offers numerous benefits, such as improved security, convenience, and

efficiency. However, it also raises concerns related to privacy, data protection, and ethical considerations, prompting the need for responsible deployment and regulation.

1.3 Technology Used

Face recognition systems use a combination of hardware and software technologies to accurately detect and identify faces. The technology has evolved over the years, and modern face recognition systems often leverage sophisticated methods, including.

1.3.1 Image Capture Devices:

Cameras: High-resolution cameras are essential for capturing clear and detailed facial images. These can range from standard RGB cameras to specialized depth-sensing cameras, infrared cameras, or 3D cameras for capturing additional depth information.

1.3.2 Face Facial Detection:

This technology is used to detect the presence of a face within an image or video frame. It locates the position and size of faces within the input data.

1.3.3 Feature Extraction:

Feature extraction techniques are employed to extract distinctive features from the detected facial images. These features could include the shape of the face, the distance between the eyes, the size of the nose, etc.

1.3.4 Machine Learning and Pattern Recognition:

Machine learning algorithms, such as deep learning neural networks, are often utilized to learn patterns and characteristics from facial images. These algorithms are trained on large datasets of labeled facial images to recognize and classify faces accurately.

1.3.5 Liveness Detection:

To prevent spoofing attacks (wherein someone tries to fool the system with a photo or video of a face), liveness detection techniques are used to verify that the face being presented is from a live person. This can involve analyzing facial movements or asking the user to perform specific actions.

1.3.6 Database Management:

Face recognition systems often require efficient database management techniques to store and retrieve facial templates or features quickly, especially in large-scale deployments.

1.3.7 Facial Detection:

Algorithms are used to detect and locate human faces within images or video frames. This involves identifying patterns such as the presence of eyes, nose, mouth, etc.

1.3.8 Machine Learning And Deep Learning:

These technologies play a crucial role in face recognition by training models to recognize patterns and features from vast amounts of facial data. Convolutional Neural Networks (CNNs) are commonly used for this purpose.

CHAPTER 2

LITERATURE REVIEW

A literature review on face recognition systems encompasses a comprehensive analysis of existing research, methodologies, and advancements in the field. The review typically covers a range of topics, including algorithms, techniques, applications, challenges, and future directions. Keep in mind that the information provided here is a general overview, and you may need to refer to specific sources for more detailed information.

This system also used machine learning algorithm which are usually used in computer vision. Also, HAAR CLASSIFIERS used to train the images from the camera capturing. The face snap by the camera capturing will convert to grayscale and do subtraction on the images; then the image is transferred to store on the server and processing later. In Request Matching, the first step is open the camera and snap the photo after the extraction the frontal face. The next step is recognizing the face with the training data and project the extracted face onto the Principal Component Analysis.

The final step displays the nearest face with the acquired images. Apart from that, adding a new face into the database is snap the photo after that extract the frontal face images and then perform the Haar cascade Method to find the perform the Principal Component Analysis Algorithm. The final step is storing the information.

Face detection is a computer technology that determines the location and size of human face in arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc. are ignored from the digital image. It can be regarded as a specific case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection, can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). Basically there are two types of approaches to detect facial part in the given image i.e. feature base and image base approach. Feature base approach tries to extract features of the image and match it against the knowledge of the face features. While image base approach tries to get best match between training and testing images.

Active Shape Model Active shape models focus on complex non-rigid features like actual physical and higher level appearance of features Means that Active Shape Models (ASMs) are aimed at automatically locating landmark points that define the shape of any statistically modelled. object in an image. When of facial features such as the eyes, lips, nose, mouth and eyebrows. The training stage of an ASM involves the building of a statistical.

2.1 Face Tracking:

Face tracking refers to identifying the features which are then used to detect a Face In this case the example method includes the receiving or we can say that it gets the first image and the second images of a face of a user who is being taken into consideration, where

one or both of the images which were used to sort of look for a match have been granted a match by the facial recognition system which also proves the correct working of the system. “The technique includes taking out a second sub- image coming from the second image, where the second sub-image includes a representation of the at least one corresponding facial landmark, detecting a facial gesture by determining whether a sufficient difference exists between the second sub - image and first sub-image to indicate the facial gesture, and determining, based on detecting the facial gesture, whether to deny authentication to the user with respect to accessing functionalities controlled by the computing”.

“Linear discriminant analysis (LDA) has been successfully used as a dimensionality reduction technique to many classification problems, such as speech recognition, face recognition, and multimedia information retrieval”. The objective is to a projection A that maximizes the ratio of between-class scatter against within-class scatter S (Fisher's criterion).

Basically what we see in this paper is that it presents an extension and a new way of perception of the author's theory for human visual information processing, which The method includes extracting a second sub-image from the second image, where the second sub-image includes a representation of the at least one corresponding facial landmark. “In turn detecting a facial gesture by determining whether a sufficient difference exists between the second sub-image and first sub-image to indicate the facial gesture, and determining, based on detecting the facial gesture, whether to deny authentication to the user with respect to the human recognition system and same was applied Several indispensable techniques are implicated: encoding of visible photographs into neural patterns, detection of easy facial features, measurement standardization, discount of the neural patterns in dimensionality.

Few procedures to computerized facial consciousness have employed geometric size of attribute points of a human face. Eye spacing dimension has been recognized as an essential step in reaching this goal. Measurement of spacing has been made by means of software of the Hough radically change method to discover the occasion of a round form and of an ellipsoidal form which approximate the perimeter of the iris and each the perimeter of the sclera and the form of the place under the eyebrows respectively. Both gradient magnitude and gradient direction were used to handle the noise contaminating the feature space. “Results of this application indicate that measurement of the spacing by detection of the iris is the most accurate of these three methods with measurement by detection of the position of the eyebrows the least accurate. However, measurement by detection of the eyebrows' position is the least constrained method.

CHAPTER 3

SYSTEM DEVELOPEMENT

3.1 Image:

An image refers a 2D light intensity function $f(x, y)$, where (x, y) denotes spatial coordinates and the value of f at any point (x, y) is proportional to the brightness or gray levels of the image at that point. A digital image is an image $f(x, y)$ that has been discretized both in spatial coordinates and brightness. The elements of such a digital array are called image elements or pixels.

3.2 A Simple Image Model:

To be suitable for computer processing, an image $f(x, y)$ must be digitalized both spatially and in amplitude. Digitization of the spatial coordinates (x, y) is called image sampling. Amplitude digitization is called gray-level quantization. The storage and processing requirements increase rapidly with the spatial resolution and the number of gray levels. Example: A 256 gray-level image of size 256×256 occupies 64k bytes of memory.

3.3 Recognizing The Model By Listing Out The Applications:

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Face detection can be regarded as a specific case of object class detection. In object class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars. Face detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process. Face Detection has found its application in various fields such as.

Facial motion capture: Facial motion capture is the process of electronically converting the movements of a person's face into a digital database using cameras or laser scanners. This database may then be used to produce CG (computer graphics) computer animation for movies, games, or real-time avatars. Because the motion of CG characters is derived from the movements of real people, it results in more realistic and nuanced computer character animation than if the animation were created manually.

Facial recognition: Facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape. It is also used in video surveillance, human computer interface and image database management.

3.4 Photography:

Some recent digital cameras use face detection for autofocus. Face detection is also

useful for selecting regions of interest in photo slideshows that use a pan-and- scale Ken Burns effect.

Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by.

3.5 Types Of Image Processing:

Low level processing

Medium level processing

High level processing

Low level processing means performing basic operations on images such as reading an image, resize, rotate, RGB to gray level conversion, histogram equalization etc..., The output image obtained after low level processing is raw image. Medium level processing means extracting regions of interest from output of low level processed image. Medium level processing deals with identification of boundaries edges. This process is called segmentation. High level processing deals with adding of artificial intelligence to medium level processed signal.

3.5 Viola-Jones Algorithm:

The Viola-Jones algorithm is a widely used mechanism for object detection. The main property of this algorithm is that training is slow, but detection is fast. This algorithm uses Haar basis feature filters, so it does not use multiplications.

The efficiency of the Viola-Jones algorithm can be significantly increased by first generating the integral image.

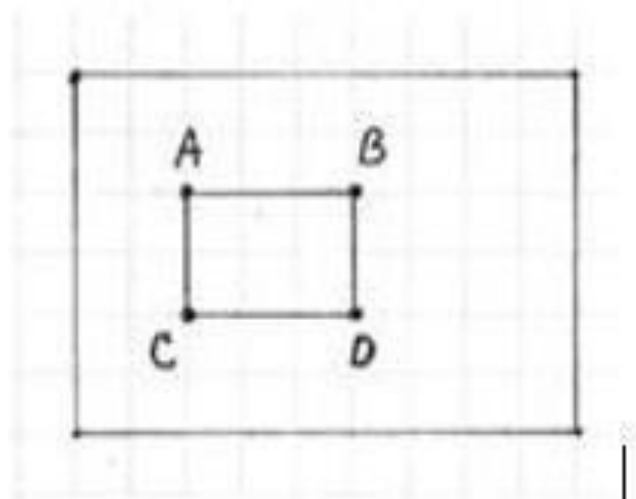


Figure 3.1 Rectangle used

Detection happens inside a detection window. A minimum and maximum window size i

s chosen, and for each size a sliding step size is chosen. Then the detection window is moved across the image as follows:

Set the minimum window size, and sliding step corresponding to that size.

For the chosen window size, slide the window vertically and horizontally with the same step. At each step, a set of N face recognition filters is applied. If one filter gives a positive answer, the face is detected in the current widow.

If the size of the window is the maximum size stop the procedure. Otherwise increase the size of the window and corresponding sliding step to the next chosen size and go to the step 2. Each face recognition filter (from the set of N filters) contains a set of cascade-connected classifiers. Each classifier looks at a rectangular subset of the detection window and determines if it looks like a face. If it does, the next classifier is applied. If all classifiers give a positive answer, the filter gives a positive answer and the face is recognized.

Each classifier is composed of Haar feature extractors (weak classifiers). Each Haar feature is the weighted sum of 2D integrals of small rectangular areas attached to each other. The weights may take values ± 1 . Fig.2 shows examples of Haar features relative to then closing detection window. Gray areas have a positive weight and white areas have a negative weight. Haar feature extractors are scaled with respect to the detection window size.

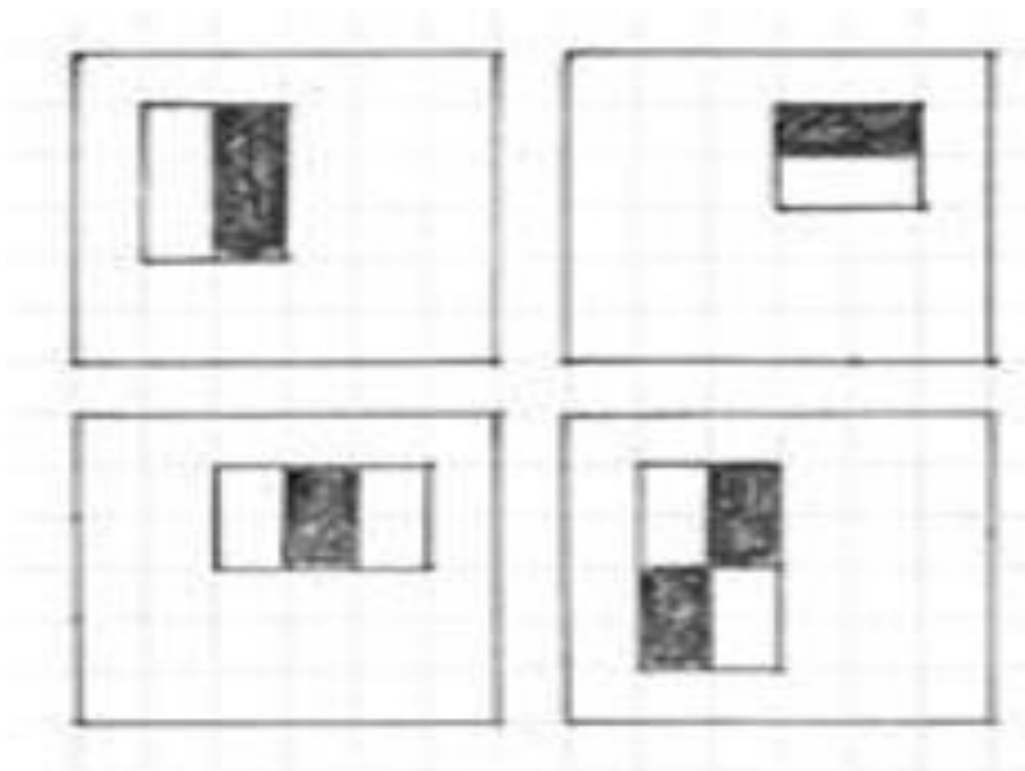


Figure 3.2 Classifiers

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection is using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

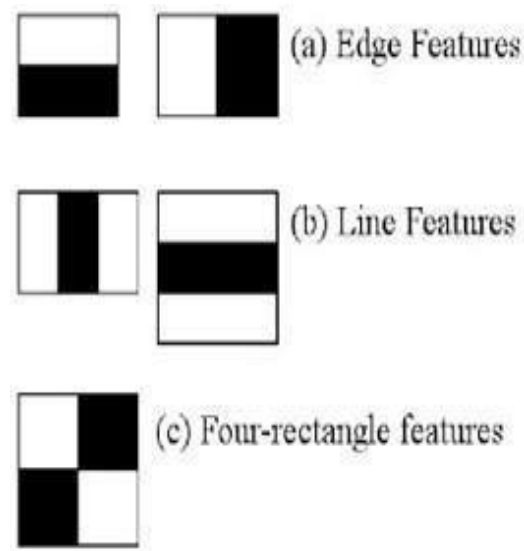


Figure 3.3 Feature Detection

Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applied to cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved.

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain). So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Isn't it a little inefficient and time consuming? Yes, it is. The authors have a good solution for that. In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region.

If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions. For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window.


3.6 Output:

The screenshot shows the landing page of the FAMS (Face Recognition Based Attendance Management System). The title bar at the top reads "FAMS-Face Recognition Based Attendance Management System". The main heading is "Face-Recognition-Based-Attendance-Management-System". Below the heading, there are two input fields for enrollment: "Enter Enrollment :" and "Enter Name :", each with a corresponding "Clear" button. To the right of these fields is a button labeled "Check Registered students". At the bottom, there are four large blue buttons: "Take Images", "Train Images", "Automatic Attendance", and "Manually Fill Attendance". The Windows taskbar is visible at the bottom, showing the search bar and various application icons.

Figure 3.4 Landing page

The screenshot shows the login page of the FAMS system. The title bar at the top reads "Login". The main heading is "Login". Below the heading, there are two input fields for login: "Enter username :" and "Enter password :", each with a corresponding "Clear" button. Below these fields is a large blue button labeled "LogIn".

Figure 3.4 Login page

 Enter subject name...

Enter Subject :

Fill Attendance

Figure 4.2 Automatic attendance

Manually attendance of os

Enter Enrollment Clear

Enter Student name Clear

Enter Data Convert to CSV

Check Sheets

Type here to search

14:16 27-02-2024

Figure 4.2 Manual fill attendance

CHAPTER 4

TRAINING

4.1 Training In OpenCV:

In OpenCV, training refers to providing a recognizer algorithm with training data to learn from. The trainer uses the same algorithm (LBPH) to convert the images cells to histograms and then computes the values of all cells and by concatenating the histograms, feature vectors can be obtained. Images can be classified by processing with an ID attached. Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face Eigen face and Fisher face compute the dominant features of the whole training set while LBPH analyses them individually.

To do so, firstly, a Dataset is created. You can either create your own dataset or start with one of the available face databases.

Yale Face Database
AT & T Face Database

The .xml configuration file is made from the several features extracted from your dataset with the help of the Face Recognizer Class and stored in the form of feature vectors.

4.2 Training The Classifiers:

OpenCV enables the creation of XML files to store features extracted from datasets using the Face Recognizer Class. The stored images are imported, converted to Grayscale and saved with IDs in two lists with same indexes. Face Recognizer objects are created using Face Recognizer class. Each recognizer can take in parameters described below.
`cv2.face.createEigenFaceRecognizer()`.

Takes in the threshold in recognizing faces. If the distance to the likeliest Eigen face is above this threshold, the function will return a -1, that can be used state the face is unrecognizable that is `cv2.face.createFisherFaceRecognizer()`.

The first argument is the number of components for the LDA for the creation of Fisher faces. OpenCV mentions it to be kept 0 if uncertain.

Similar to Eigen face threshold. -1 if the threshold is passed that is `cv2.face.createLBPHFaceRecognizer()`.

The radius from the center pixel to build the local binary pattern. The Number of sample points to build the pattern. Having a considerable number will slow down the computer. The Number of Cells to be created in X axis. The number of cells to be created in Y axis. A threshold value similar to Eigen face and Fisher face. if the threshold is passed the object will return Recognizer objects are created and images are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is gathered from splitting the file name, and stored in another vector. By using

FaceRecognizer.train(NumpyImage, ID) all three of the objects are trained. It must be noted that resizing the images were required only for Eigenface and Fisherface, not for LBPH. The configuration model is saved as XML using the function: FaceRecognizer.save(FileName).Recognizer class. The stored images are imported, converted to grayscale and saved with IDs in two lists with same indexes. Face Recognizer objects are created using face recognizer class.

4.3 Train() Function:

Trains a Face Recognizer with given data and associated labels. Parameters:

The training images, that means the faces you want to learn. The data has to be given as a vector<Mat>. labels The labels corresponding to the images have to be given either as a vector<int> or any other data type.

4.4 Dataset:

This is the code that will be used to create a dataset. It will turn the camera and take number of pictures for few seconds. Given below is the code for face_dataset.py. After running the dataset code, we will get number of pictures in a folder named dataset. Now these photos will be used to train. The more the pics the greater the accuracy of the trainer.

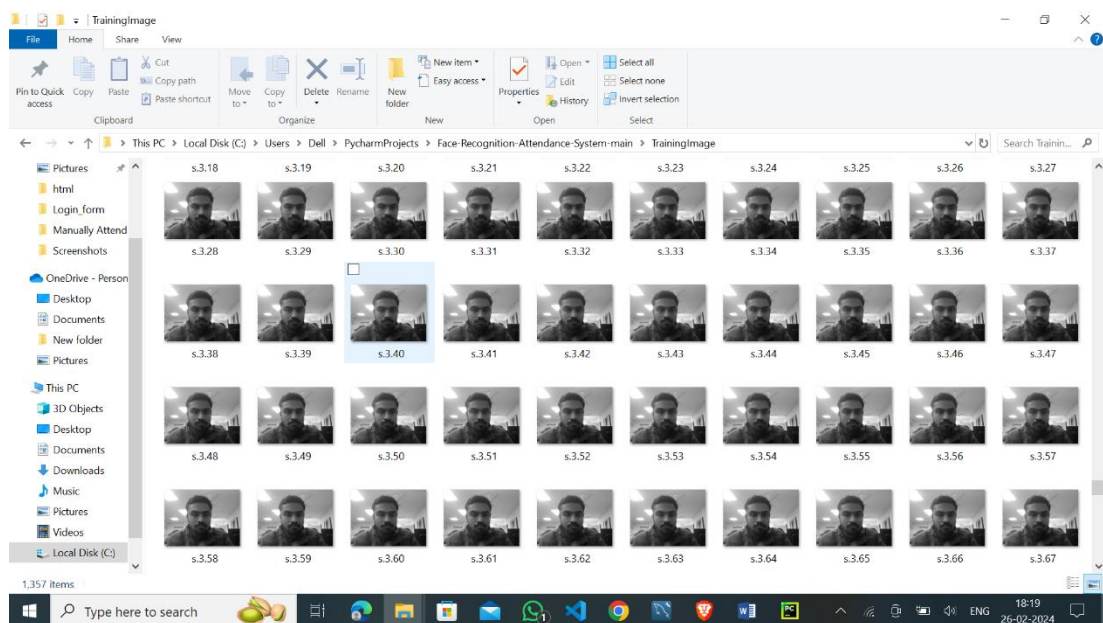


Figure 4.2 Manual fill attendance

4.5 Facerecognizer Class:

All face recognition models in OpenCV are derived from the abstract base class FaceRecognizer, which provides a unified access to all face recognition algorithms in OpenCV.

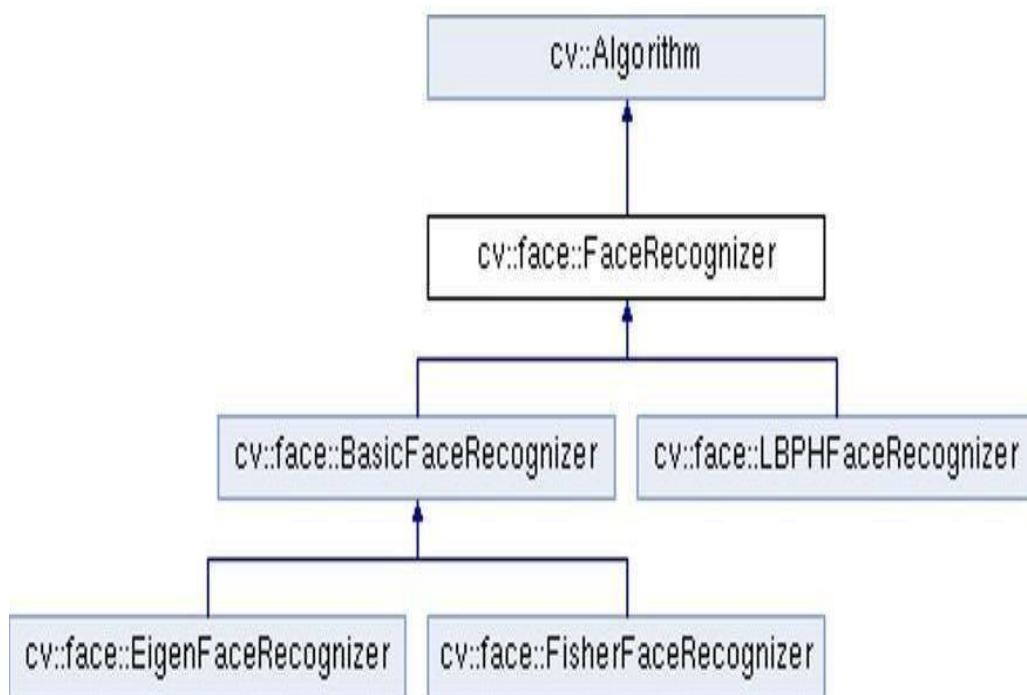


Figure 4.8: Flowchart Representation

It doesn't look like a powerful interface at first sight. But: Every FaceRecognizer is an Algorithm, so you can easily get/set all model internals (if allowed by the implementation). Algorithm is a relatively new OpenCV concept, which is available since the 2.4 release.

Algorithm provides the following features for all derived classes:

So called "virtual constructor". That is, each Algorithm derivative is registered at program start and you can get the list of registered algorithms and create instance of a particular algorithm by its name (see `Algorithm::create`). If you plan to add your own algorithms, it is good practice to add a unique prefix to your algorithms to distinguish them from other algorithms.

If you used video capturing functionality from OpenCV high module, you are probably familiar with `cv::cvSetCaptureProperty`, `opencvGetCaptureProperty`, `VideoCapture::set` and `VideoCapture::get`. Algorithm provides similar method where instead of integer id's you specify the parameter names as text Strings.

Reading and writing parameters from/to XML or YAML files. Every Algorithm derivative can store all its parameters and then read them back. There is no need to re-implement it each time.

CHAPTER 5

TESTING

7.1 Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but it could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist in testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

7.1.1 Benefits of Unit Testing

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it offers several benefits.

Find Problems Early:

Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

Facilitates Change:

Unit testing allows the programmer to refactor code or upgrade system libraries later, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes that may break a design contract.

Simplifies Integration:

Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

Documentation:

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in the development unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

7.2 Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify the functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, with success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. Some different types of integration testing are big-bang, top- down, and bottom-up, mixed (sandwich) and risky- hardest. Other Integration Patterns are collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

7.2.1 Big Bang

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing because it expects to have few problems with the individual components.

The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

7.2.2 Top-Down and Bottom-Up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready.

This method also helps to determine the levels of software developed and makes it easier to report

testing progress in the form of a percentage. Top-down testing is an approach to integrated testing

where the top integrated modules are tested, and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top-down testing with bottom-up testing.

7.3 Black-Box Testing

Black box testing is a technique used to test the functionality of a software application without having knowledge of its internal structure or implementation details. It focuses on the inputs and outputs of the system and verifies if the expected outputs match the desired results. Here are some examples of black box testing techniques that can be applied to the KIET Event Management App:

Equivalence Partitioning:

- Identify different categories of inputs for the app, such as valid and invalid inputs, and divide them into equivalence classes.
- Test representative values from each equivalence class to ensure the app behaves consistently within each class.

Boundary Value Analysis:

- Identify the boundaries or limits for inputs in the app, such as minimum and maximum values, and test values at those boundaries.
- Test values just above and below the boundaries to verify the app's behaviour at critical points.

Decision Table Testing:

- Identify the different conditions and rules that govern the behaviour of the app.
- Create a decision table with combinations of conditions and corresponding expected results.

- Test different combinations of conditions to validate the app's decision-making process.

State Transition Testing:

- Identify the different states that the app can transition between.
- Define the valid and invalid transitions between states.
- Test different sequences of state transitions to verify the app's behaviour.

Error Guessing:

- Use experience and intuition to guess potential errors or issues in the app.
- Create test cases based on those guesses to verify if the app handles the errors correctly.

Compatibility Testing:

- Test the app on different platforms, browsers, or devices to ensure compatibility.
- Verify that the app functions correctly and displays appropriately across different environments.

Usability Testing:

- Evaluate the app's user interface and interactions from the perspective of an end-user.
- Test common user scenarios and assess the app's ease of use, intuitiveness, and overall user experience.

Security Testing:

- Test the app for potential security vulnerabilities or weaknesses.
- Verify if the app handles user authentication, data encryption, and access control appropriately.

Performance Testing:

- Test the app's performance under different load conditions, such as a high number of concurrent users or large data sets.
- Verify if the app responds within acceptable time limits and performs efficiently.

During black box testing, test cases are designed based on the app's specifications, requirements, and user expectations. The focus is on validating the functionality, user interactions, and expected outputs without considering the internal implementation details of the app.

7.4 White-Box Testing

White box testing, also known as structural testing or glass box testing, is a software testing technique that examines the internal structure and implementation details of the application. It aims to ensure that the code functions as intended and covers all possible execution paths. Here are some examples of white box testing techniques that can be applied to the KIET Event Management App:

Unit Testing:

- Test individual units or components of the app, such as functions or methods, to verify their correctness.
- Use techniques like code coverage analysis (e.g., statement coverage, branch coverage) to ensure that all code paths are exercised.

Integration Testing:

- Test the interaction between different components or modules of the app to ensure they work together seamlessly.
- Verify the flow of data and control between the modules and check for any integration issues or errors.

Path Testing:

- Identify and test different paths or execution flows through the app, including both positive and negative scenarios.
- Execute test cases that cover all possible paths within the code to ensure complete coverage.

Decision Coverage:

- Ensure that every decision point in the code (e.g., if statements, switch cases) is tested for both true and false conditions.
- Validate that the app makes the correct decisions based on the specified conditions.

Code Review:

- Analyze the code and its structure to identify any potential issues or vulnerabilities.
- Review the adherence to coding standards, best practices, and potential optimizations.

Performance Testing:

- Assess the app's performance from a code perspective, such as identifying any bottlenecks or inefficient algorithms.
- Measure the execution time of critical code sections and evaluate resource usage.

Security Testing:

- Review the code for potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), or authentication weaknesses.
- Verify the implementation of secure coding practices, data encryption, and access control mechanisms.

Error Handling Testing:

- Test how the app handles and recovers from unexpected errors or exceptions.
- Validate that error messages are clear, meaningful, and do not expose sensitive information.

Code Coverage Analysis:

- Use tools to measure the code coverage achieved by the tests, such as statement coverage, branch coverage, or path coverage.
- Aim for high code coverage to ensure that all parts of the code are exercised.

During white box testing, the tester has access to the application's internal code, allowing for a more detailed examination of its behaviour.

7.5 System Testing

System testing is a level of software testing that evaluates the complete system as a whole, rather than focusing on individual components or modules. It ensures that all components of the KIET Event Management App work together seamlessly and meet the specified requirements. Here are some examples of system testing techniques that can be applied to the app:

Functional Testing:

- Verify that all functional requirements of the app are met.
- Test various functionalities such as event creation, registration, club directory search, user log in and registration, event notifications, etc.
- Validate that the app behaves as expected and produces the correct outputs based on different inputs.

User Interface Testing:

- Test the graphical user interface (GUI) of the app for usability, consistency, and responsiveness.
- Check the layout, navigation, buttons, forms, and other UI elements to ensure they are visually appealing and intuitive.
- Validate that the app adheres to the design guidelines and provides a seamless user experience.

Performance Testing:

- Evaluate the performance of the app under different load conditions.
- Measure response times, throughput, and resource utilization to ensure the app can handle the expected user load without significant degradation.
- Identify and address any performance bottlenecks or scalability issues.

Compatibility Testing:

- Test the app on different devices, platforms, and browsers to ensure compatibility.
- Verify that the app works correctly on various operating systems (e.g., iOS, Android) and different screen sizes.

- Validate that the app functions properly on different web browsers (if applicable).

Security Testing:

- Assess the app's security measures to protect user data and prevent unauthorized access.
- Perform vulnerability scanning, penetration testing, and authentication testing to identify and address any security vulnerabilities.
- Test the app's resilience against common security threats, such as cross-site scripting (XSS) and SQL injection.

Integration Testing:

- Test the integration of the app with external systems, such as databases, mapping services, or notification services.
- Validate that data is exchanged correctly between the app and external systems.
- Verify that the app's functionality remains intact when integrated with other systems.

CHAPTER 6

DATABASE DESIGN

6.1 Flowchart:

This appears to be a screenshot of the user interface for a facial recognition-based attendance system. Systems like this are used to track attendance automatically, often in schools or workplaces.

Key Features:

- **Face Registration:** The system allows you to enroll new faces with names.
- **Image Capture:** A camera is used to take pictures of individuals.
- **Training:** The system uses captured images to 'learn' to recognize faces.
- **Attendance Options:** Offers both automated attendance (likely using facial recognition against the database) and manual attendance entry options.

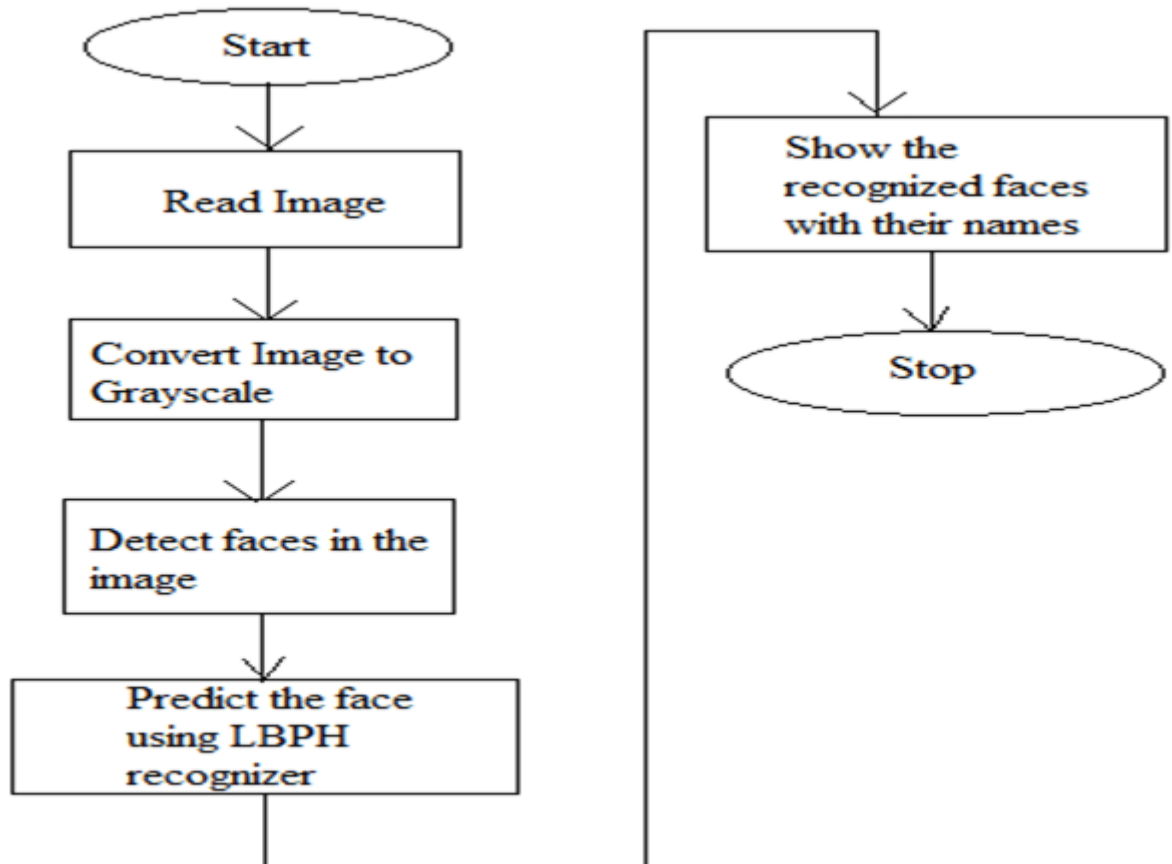


Figure 6.1 Data flow

6.2 Data Flow :

- **Python Scripts:** Files like "AMS Run.py", "faceee.py", "training.py" are the core code files responsible for the system's logic.
- **CSV File:** "StudentDetails.csv" probably stores student data (like enrollment numbers, names, and possibly images used for reference by the system).
- **Image Files:** "trainingimage", "Trainingimagelabel", "Screenshot" files likely contain images used to train the facial recognition model.
- **XML Files:** "haarcascade_frontalface_alt.xml" and similar files are pre-trained models used for general face detection.

6.3 System Functionality:

- **Facial Detection:** The system uses the camera to detect faces in a live video feed or still images.
- **Facial Recognition:** It attempts to match detected faces against the enrolled student database.
- **Attendance Marking:** If a match is found, the system can automatically mark the student as present, potentially with a timestamp and image for later verification if needed.

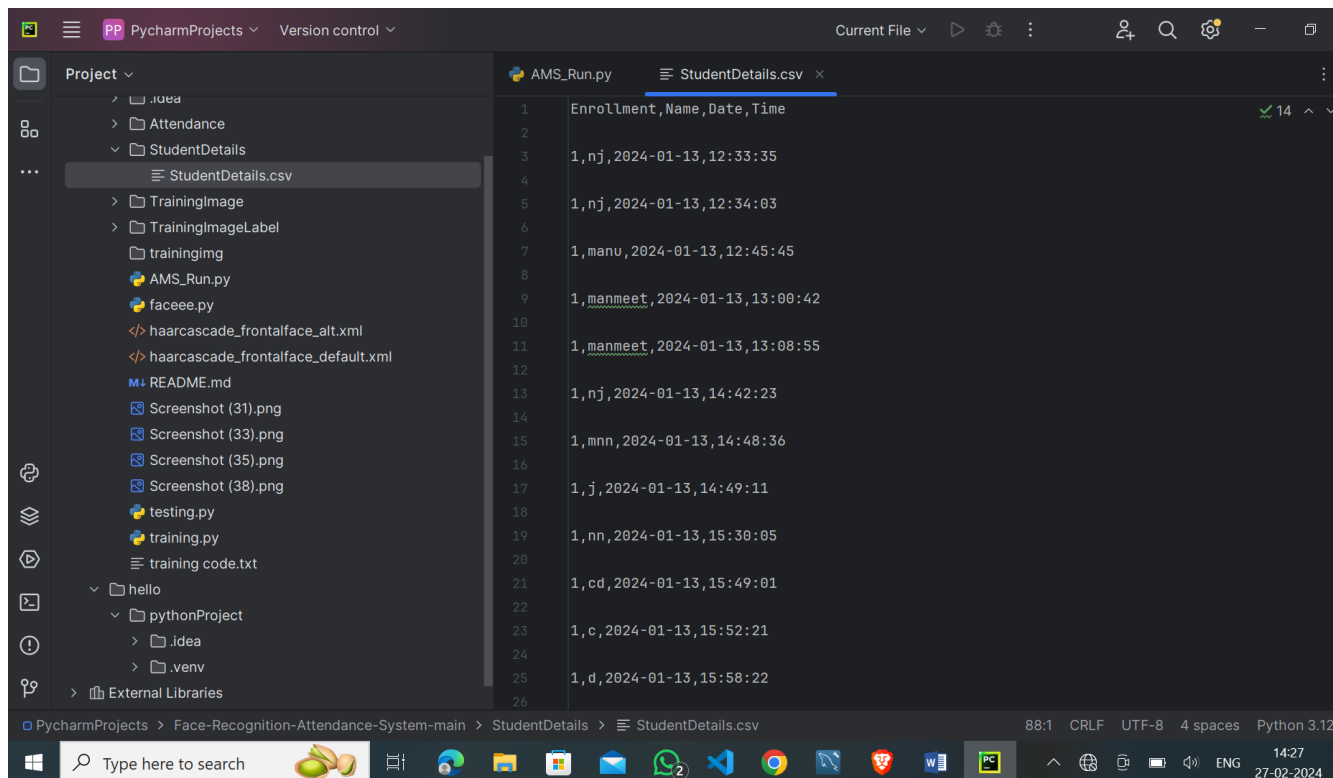


Figure 6.2 Database

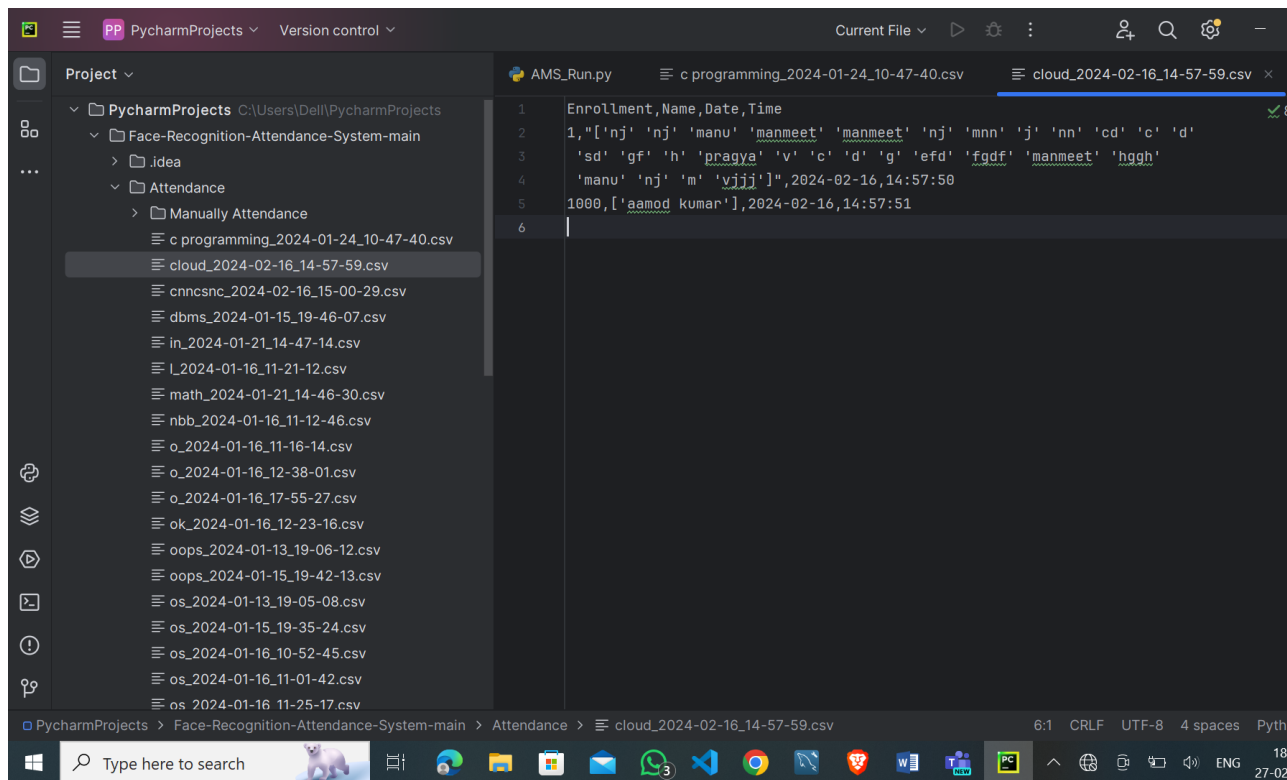


Figure 6.3 Database of subject attendance

Training of a FaceRecognizer with FaceRecognizer.train on a given set of images (your face database!).

Prediction of a given sample image, that means a face. The image is given as a Mat.

Loading/Saving the model state from/to a given XML or YAML.

Setting/Getting labels info, that is stored as a string. String labels info is useful for keeping names of the recognized people.

6.4 Lbph Recognizer:

The approach that has been used in this project is, LBPH approach which uses the following algorithm to compute the feature vectors of the provided images in the dataset. Local Binary Patterns (LBP) is a type of visual descriptor used for classification in computer vision. LBP was first described in 1994 and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with the Histogram of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

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.

In this section, it is shown a step-by-step explanation of the LBPH algorithm: 1.First of all, we need to define the parameters (radius, neighbours, grid x and grid y) using the Parameters structure from the lbph package. Then we need to call the Init function passing the structure with the parameters. If we not set the parameters, it will use the default parameters as explained in the Parameters section.

Secondly, we need to train the algorithm. To do that we just need to call the Train function

passing a slice of images and a slice of labels by parameter. All images must have the same size. The labels are used as IDs for the images, so if you have more than one image of the same texture/subject, the labels should be the same.

The Train function will first check if all images have the same size. If at least one image has not the same size, the Train function will return an error and the algorithm will not be trained.

Then, the Train function will apply the basic LBP operation by changing each pixel based on its neighbours using a default radius defined by the user. The basic LBP operation can be seen in the following image (using 8 neighbours and radius equal to 1).

After applying the LBP operation we extract the histograms of each image based on the number of grids (X and Y) passed by parameter. After extracting the histogram of each region, we concatenate all histograms and create a new one which will be used to represent the image.

The images, labels, and histograms are stored in a data structure so we can compare all of it to a new image in the Predict function. To predict a new image we just need to call the Predict function passing the image as parameter. The Predictfunction will extract the histogram from the new image, compare it to the histograms stored in the data structure and return the label and distance corresponding to the closest histogram if no error has occurred. Note: It uses the Euclidian distance metric as the default metric to compare the histograms. The closer to zero is the distance, the greater is the confidence.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Future Scope

Government/ Identity Management: Governments all around the world are using face recognition systems to identify civilians. America has one of the largest face databases in the world, containing data of about 117 million people.

Emotion & Sentiment Analysis: Face Detection and Recognition have brought us closer to the technology of automated psyche evaluation. As systems now a days can judge the precise emotions frame by frame in order to evaluate the psyche.

Authentication systems: Various devices like mobile phones or even ATMs work using facial recognition, thus making getting access or verification quicker and hassle free.

Full Automation: This technology helps us become fully automated as there is very little to zero amount of effort required for verification using facial recognition.

High Accuracy: Face Detection and Recognition systems these days have developed very high accuracy and can be trained using very small data sets and the false acceptance rates have dropped down significantly.

7.2 Applications:

Security: Face Recognition can help in developing security measures, that is unlocking of a safe using facial recognition.

Attendance Systems: Face Recognition can be used to train a set of users in order to create and implement an automatic attendance system that recognizes the face of the individual and marks their attendance.

Access: Face Detection can be used to access sensitive information like your bank account and it can also be used to authorize payments.

Mobile Unlocking: This feature has taken the mobile phone industry by a storm and almost every smart phone manufacturing company has their flagship smartphones being unlocked using face recognition. Apple's Face ID is an excellent example.

Law Enforcement: This is a rather interesting way of using face detection and face recognition as it can be used to assess the features of a suspect to see if they are being truthful in their statements or not.

Healthcare: Face Recognition and Detection can be used in the healthcare sector to assess the illness of a patient by reading

7.3 Limitations

Data Storage: Extensive data storage is required for creating, training and maintaining big face databases which is not always feasible

Computational Power: The requirement of computational power also increases with increase in the size of the database. This becomes financially out of bounds for smaller organizations.

Camera Angle: The relative angle of the target's face with the camera impacts the recognition rate drastically. These conditions may not always be suitable, therefore creating a major drawback.

7.4 Conclusion

- Facial Detection and Recognition systems are gaining a lot of popularity these days. Most of the flagship smartphones of major mobile phone manufacturing companies use face recognition as the means to provide access to the user.
- This project report explains the implementation of face detection and face recognition using OpenCV with Python and also lays out the basic information that is needed to develop a face detection and face recognition software. The goal of increasing the accuracy of this project will always remain constant and new configurations and different algorithms will be tested to obtain better results. In this project, the approach we used was that of Local Binary Pattern Histograms that are a part of the FaceRecognizer Class of OpenCV.

CHAPTER 8

REFERENCES

- Predinger, Ishizuka, “The empathic companion: a character-based interface that addresses users' affective states”, 2007.
- Nunamaker Jr., C. Derrick, et al.: “Embodied Conversational Agent-Based Kiosk for Automated Interviewing”, 2011.
- Maras et al. “Ameliorating the disadvantage for autistic job seekers: An initial evaluation of adapted employment interview questions”, 2021.
- Robinson, Marica F., “Artificial Intelligence in Hiring: Understanding Attitudes and Perspectives of HR Practitioners”, 2019.
- Hosselet, P.F., “The acceptance of AI enabled decision support systems a project management perspective”, 2018.
- Tom Taulli, “Artificial Intelligence Basics: A non Technical Basics”, 2019.
- Crutsinger, Herrera, “Mock Interviews: Leveraging AI Resources To Enhance Professional Skills”, 2022.
- B. Powell et al., “An overview of mock interviews as a training tool for interviewers of children”, 2022.
- Chou et al., “An AI Mock-interview Platform for Interview Performance Analysis”, 2022.

BIBLIOGRAPHY

Reference Books

Following are the books that we had referred for our project Mock Interview System:

- "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig
This comprehensive textbook provides insights into various aspects of artificial intelligence, including machine learning and natural language processing.
- "Building Scalable and Responsive Web Applications with React" by Adam Freeman
Ideal for those working with this book covers building scalable and responsive web applications, which aligns with the frontend development aspect of your project.
- "Mastering React: Build Scalable and High-Performance Web Applications with React 17" by Tom Banks
This book delves into advanced concepts and best practices for mastering React, which can be valuable for optimizing the frontend of your application.
- "Practical Natural Language Processing with Python: A Comprehensive Guide to Building Real-World NLP Applications" by Dipanjan Sarkar
Focused on natural language processing, this book provides practical insights and hands-on examples for building NLP applications, aligning with the language processing aspect of your project.
- "Virtual Reality in Education: A Practical Guide for Teachers and Developers" by Charles Wankel and Patrick Blessinger
Explore the potential of virtual reality in education, which could be insightful for understanding the implications of VR integration in your mock interview system.
- "Web Analytics: An Hour a Day" by Avinash Kaushik
For those interested in understanding web analytics, this book provides a structured approach and best practices, which can be beneficial for analyzing user interactions on your platform.

Online Websites

The following are the AI enabled mock interview system websites that we had analyzed for ours:

- <https://www.greetai.co/practice>
- <https://beta.interviewai.in/>

- <https://www.acetheinterview.app/>
- <https://geekflare.com/ai-powered-interview-preparation-platforms/>

