

Contents

Page No.

<i>Abstract</i>	3
<i>List of Tables</i>	4
<i>List of Figures</i>	5
1 Introduction	6
1.1 Background of the Study	6
1.2 Problem Statement	6
1.3 Glossary	7
2 Literature Survey	8
3 Problem Definition	9
3.1 Scope	9
3.2 Exclusions	9
3.3 Assumptions.....	9
4 Project Planning.....	10
4.1 Software Life Cycle Model	10
4.2 Scheduling.....	12
4.3 Cost Analysis.....	13
5 Requirements Analysis.....	14
5.1 Functional Requirements.....	14
5.2 Non-functional Requirements.....	14
5.3 Platform Requirements.....	14
5.4 Software Requirements.....	15
5.5 Hardware Requirements.....	15
6 Detailed Design.....	16
6.1 Project Use-case Diagram.....	16
6.2 Project Block Diagram.....	17
6.3 Data Flow Diagram.....	18
6.4 Project Flowchart.....	18
6.5 Data-set Creation Flowchart.....	19

6.6 E-R Diagram.....	19
7 Required Algorithms.....	20
7.1 LBPH Algorithm.....	20
7.2 Haar Cascade Classifier.....	22
8 Implementation.....	24
8.1 System Installation.....	24
8.1.1 VS-Code Installation.....	24
8.1.2 Python-3.10.0 Installation.....	24
8.2 Software Implementation.....	25
8.2.1 Open-CV Installation.....	25
8.2.2 Haar Cascade Implementation.....	27
8.2.3 face_recognizer.yml Implementation.....	29
8.3 Code Implementation & Results.....	31
8.3.1 main_registration.py.....	32
8.3.2 traning.py.....	36
8.3.3 main_attendance.py.....	37
9 Conclusion.....	40
9.1 Projects Benefits.....	40
9.2 Future Scope for Improvements.....	40
10 References /Bibliography.....	41

Abstract

In colleges, universities, organizations, schools, and offices, taking attendance is one of the most important tasks that must be done on a daily basis. The majority of the time, it is done manually, such as by calling by name or by roll number. The main goal of this project is to create a Face Recognition-based attendance system that will turn this manual process into an automated one. This project meets the requirements for bringing modernization to the way attendance is handled, as well as the criteria for time management. This device is installed in the classroom, where student's information, such as name, roll number, class, sec, and photographs, is trained. The images are extracted using Open CV.

Before the start of the corresponding class, the student can approach the machine, which will begin taking pictures and comparing them to the qualified dataset. Logitech C270 web camera and PC were used in this project as the camera and processing system. The image is processed as follows: first, faces are identified using a Haar cascade classifier, then faces are recognized using the LBPH (Local Binary Pattern Histogram) Algorithm, histogram data is checked against an established dataset, and the device automatically labels attendance. An Excel sheet is developed, and it is updated every hour with the information from the respective class instructor.

Keywords: *Face Detection, LBPH algorithm Face Recognition, Haar-Cascade classifier, Face-Recognizer.yml, attendance system, Correlation*

List of Tables

Table	Title	Page
1	Project Scheduling & Work progress	12
2	Cost Analysis	13
3	Hardware Requirements	15

List of Figures

Figure Title		Page
Figure 1	Agile model of the Project.	12
Figure 2	Use-case Diagram: Registration.	16
Figure 3	Use -case Diagram: Attendance.	16
Figure 4	Project Block Diagram.	17
Figure 5	Data Flow Diagram.	18
Figure 6	Project Flowchart.	18
Figure 7	Dataset Creation flowchart	19
Figure 8	E-R Diagram.	19
Figure 9	LBPH Operation.	20
Figure 10	LBPH Radus Change	21
Figure 11	Extracting the Histogram.	21
Figure 12	Euclidean Distance Formula.	22
Figure 13	Image Training.	23
Figure 14	Face Detection.	34
Figure 15	Dataset Sample.	35
Figure 16	Registration Datasheet.	35
Figure 17	Face Recognize.	39
Figure 18	Generated Attendance Datasheet.	39

1. Introduction:

This project delves into the exciting realm of automated face recognition. We aim to develop a system that can intelligently identify individuals using facial features. Imagine applications like security systems, attendance monitoring, or even personalized experiences in stores. By leveraging computer vision and machine learning, this project has the potential to revolutionize the way we interact with technology.

1.1 Background of the Study:

Face recognition is essential in daily life for identifying familiar faces such as family and friends. Though it may seem instantaneous, several steps are involved in this process. Human intelligence enables us to receive and interpret visual information. Light, a form of electromagnetic waves, is reflected from objects and projected onto the retina of our eyes. According to Robinson-Riegler and Robinson-Riegler (2008), the human visual system processes this information by classifying the shape, size, contour, and texture of objects. This analyzed information is then compared to stored representations in our memory to recognize faces.

Building an automated system with human-like face recognition capabilities is challenging. Humans have limited memory and can struggle to remember every individual's face, such as in a university with many students of diverse backgrounds. Computers, with their vast memory and high processing power, can overcome these limitations.

Face recognition is a biometric method that identifies individuals by comparing real-time captured images with stored images in a database (Margaret Rouse, 2012). This technology is now widespread due to its simplicity and high performance. It is used in airport security, criminal investigations by the FBI, and tracking suspects, missing children, and drug activities.

1.2 Problem Statement:

Traditional methods of marking student attendance, such as calling names or passing attendance sheets, disrupt teaching and distract students, especially during exams and in large classes. These methods are cumbersome and can lead to issues such as fraudulent sign-ins. A face recognition attendance system offers a simpler and more efficient alternative by automating attendance tracking, reducing distractions, and preventing fraud.

This automated system ensures real-time processing, meaning it must identify students within defined time constraints to avoid omissions. It must also maintain high accuracy and fast

computation, despite changes in background, illumination, pose, and facial expressions. This technology eliminates the need for manual attendance checks, allowing lecturers to focus on teaching without repeatedly counting students.

1.3 Glossary:

- Face Detection: Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images.
- Face Recognition: It is a type of biometric software application that can identify a specific individual in a digital image by analyzing and comparing patterns.
- LBPH Algorithm: A widely used method for real-time object detection. OpenCV (Open-Source Computer Vision): is a library of programming functions mainly aimed at real-time computer vision.
- RTC: Real-Time-Capture, capturing an image or video data at the time it occurs, often used for face detection in security applications.
- Correlation: Correlation refers to any of a broad class of statistical relationships involving dependence.

2. Literature Survey:

Automated attendance systems using face recognition have gained increasing popularity in recent years. They are an effective way to streamline attendance tracking and eliminate the need for manual processes. A literature survey of automated attendance systems using face recognition reveals a significant amount of research in this area.

One study conducted by Kumar et al. (2021) proposed a facial recognition-based attendance system using deep learning techniques. They used a convolutional neural network (CNN) to extract facial features and recognize students' identities. Their results showed that their system achieved an accuracy of 97.5%.

Another study by Bhardwaj et al. (2021) proposed an automated attendance system based on the fusion of deep learning and computer vision techniques. They used a combination of face detection and recognition algorithms to identify students and track their attendance. Their system achieved an accuracy of 99.4%.

A study by Patil and Swami (2020) proposed a face recognition-based attendance system using a Raspberry Pi and OpenCV. They used the Eigenface algorithm to recognize faces and track attendance. Their results showed that their system achieved an accuracy of 92.5%.

In another study, Singh et al. (2020) proposed an automated attendance system based on a hybrid deep learning model. They used a combination of CNN and long short-term memory (LSTM) networks to recognize faces and track attendance. Their system achieved an accuracy of 98.5%. Finally, a study by Zhang et al. (2019) proposed a deep learning-based attendance system that can recognize faces in real-time. They used a Siamese neural network to extract facial features and track attendance. Their system achieved an accuracy of 98.8%.

Overall, the literature survey indicates that automated attendance systems using face recognition have achieved high accuracy rates and can be a valuable tool for educational institutions and organizations to streamline attendance tracking processes.

3. Problem Definition:

This project aims to develop a system that automatically locates human faces within images or videos (detection). This system should function accurately in real-time for various applications (security, social media, etc.).

3.1 Scope:

We are setting up to design a system comprising of three modules.

- The first module (face detector) is a software program, which is basically captures student faces and stores them in (.jpg) file using computer vision face detection algorithms and face extraction techniques.
- The second module (face data training) is another software program that takes the face images and store trained data as (.yml) file.
- The third module is a desktop application that does face recognition, marks the students register and then stores the results in a database for future analysis.

3.2 Exclusions:

- Assumes basic hardware setup and excludes extensive hardware maintenance.
- Limited to on-site attendance, not for remote participants.
- Does not use other biometric methods (fingerprint, iris, voice).
- No integration with existing HR or payroll systems.

3.3 Assumptions:

The successful implementation of the Face Recognition Attendance System is based on certain assumptions to be considered throughout its deployment. Firstly, it is assumed that the hardware components supporting the system, including the webcam and computing infrastructure, will remain operational without frequent failures. While efforts have been made to create a resilient system, unexpected crashes or malfunctions in hardware are inherent risks that are assumed to be mitigated through routine maintenance and prompt technical support. Another assumption integral to the Face Recognition Attendance System is the expectation that users will comply with recommended guidelines during attendance marking. It is assumed that individuals will position themselves one at a time in front of the camera to optimize face recognition accuracy. The effectiveness of the system depends on users adhering to this protocol, ensuring that the facial recognition process occurs in a controlled and organized manner. Additionally, the system assumes a controlled and well-lit environment during attendance marking. It is expected that users will choose locations with adequate lighting, minimizing variations that could potentially impact the accuracy of the facial recognition algorithm. This assumption underscores the importance of environmental conditions in achieving optimal results

4. Project Planning:

A project work plan allows you to outline the requirements of a project, project planning steps, goals, and team members involved in the project. Within each goal, you're going to outline the necessary Key Action Steps in project planning, the requirements, and who's involved in each action step.

4.1 Software Life Cycle Model:

The Agile software life cycle model is well-suited for projects that require flexibility and continuous improvement. It allows for iterative development, where each phase can be revisited based on feedback and changing requirements. Here is a detailed overview of the phases involved in the Agile model for our face recognition and attendance system project:

1. Planning

Purpose: Understand and document user needs and expectations.

Activities:

- Stakeholder Engagement: Gather requirements from all stakeholders.
- User Stories: Capture specific needs through user stories.
- Requirement Prioritization: Prioritize requirements by importance.
- Documentation: Outline functional and non-functional requirements.

Outcome: A clear set of requirements for the project.

2. Design

Purpose: Create the system's blueprint, focusing on technical architecture and user interface.

Activities:

- System Architecture Design: Define data flow, algorithms, and integration points.
- Algorithm Design: Develop the face recognition algorithm.
- Technology Selection: Choose appropriate technologies and tools.

Outcome: Detailed design documents and prototypes.

3. Development

Purpose: Develop and integrate the system components.

Activities:

- Development: Code the face recognition and attendance systems.
- Integration: Ensure seamless interaction between system components.

- Incremental Delivery: Develop in iterative sprints, delivering functional increments.

Outcome: Working software incrementally improved.

4. Testing

Purpose: Ensure the system functions correctly and meets requirements.

Activities:

- Unit Testing: Test individual components.
- Integration Testing: Test component interactions.
- System Testing: Test overall functionality and performance.
- User Acceptance Testing (UAT): Test with actual users.

Outcome: A thoroughly tested system ready for deployment.

5. Deployment

Purpose: Release the system into the real-world environment.

Activities:

- Environment Setup: Prepare the deployment infrastructure.
- System Installation: Install and configure the system.
- Training and Support: Provide user training and support.

Outcome: An operational system with trained users.

6. Review

Purpose: Keep the system running smoothly and improve it continuously.

Activities:

- Performance Monitoring: Monitor system performance.
- Bug Fixes: Address issues and bugs.
- Updates and Enhancements: Regularly update the system.
- Adaptation: Adapt to changing requirements.

Outcome: An effective and evolving system aligned with user needs.

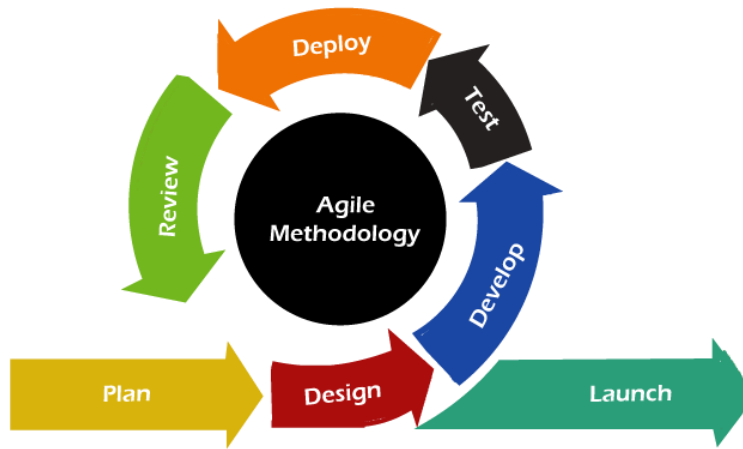


Figure1: Agile Methodology

4.2 Scheduling:

In order to develop this system, we gave enormous importance to scheduling because we believed if we want to provide the best of quality in a given period of time then we must give due importance to scheduling which also helped us to achieve a better result. we observe the entire work structure, meaning how the scheduling was maintained throughout the developmental phase. We shall also see the financial foundation of this project and furthermore the feasibility study should be also discussed.

Month	Activity	status
June	Selection of project area and Study of the related work.	Completed
August	Literature Survey and Study of Journals related to the work	Completed
September	Study on the software implementation works python and image processing	Completed
October	Study of project related works like face recognition and detection techniques	Completed
November	Study of the Image processing in python and Open Computer Vision	Completed
December	Study of hardware and selection of components	Completed
January	Study of hardware implementation and installation OS	Completed
February	Study related to creating the environments and working platform	Completed
March	Study of packages/tools and installation of packages	Completed
April	Implementation of algorithm in Software.	Completed
May	Implementation of code in Software	Complete

Table 1: Project Scheduling & Work progress

4.3 Cost Analysis:

Financial Plan identifies the Project Finance needed to meet specific objectives. The Financial Plan defines all of the various types of expenses that a project will incur (equipment, materials and administration costs) along with an estimation of the value of each expense. The Financial Plan also summarizes the total expense to be incurred across the project and this total expense becomes the project budget. As part of the Financial Planning exercise, a schedule is provided which states the amount of money needed during each stage of the project.

Components	Price (₹)
High Resolution Camera	1500/-
Computer/Server	20,000 – 30,000/-
Face Recognition Software	Free
Development Tools	Free
Power Supply	500/-
Miscellaneous	1000/-

Table2: Cost Analysis

5. Requirement Analysis:

Requirements analysis for an automated face recognition project involves several key steps and considerations to ensure the system meets user needs and operates effectively. Here's a detailed overview:

5.1 Functional Requirements:

1. Face Detection and Recognition:

- The system must detect faces in real-time from images or video streams.
- It should recognize and match faces against a stored database with high accuracy.

2. Database Integration:

- The system must integrate with existing databases for identity verification.
- It should support adding, updating, and deleting user data.

3. User Interface:

- Provide an intuitive interface for users to interact with the system.
- Include features for administrators to manage the database and system settings.

4. Alerts and Notifications:

- Generate alerts for successful and unsuccessful recognition attempts.
- Notify relevant personnel in case of security breaches or mismatches

5.2 Non-Functional Requirements:

1. Performance:

- Ensure low latency in face detection and recognition processes.

2. Scalability:

- The system should scale to handle increasing numbers of users and larger databases.

3. Security and Privacy:

- Implement strong encryption for data storage and transmission.
- Ensure compliance with data protection regulations like GDPR.

4. Reliability and Availability:

- Ensure high system uptime and robustness against failures.
- Implement redundancy and failover mechanisms.

5.3 Platform Requirements:

Supportive Operating Systems:

- Windows XP/7/8/10
- Mac OS
- Linux Ubuntu/Mint
- Android/iOS/Windows Mobile

5.4 Software Requirements:

- OpenCV 2.4.11 with gtk+
- python 3.3.0
- Visual Studio 2013(Community Edition)
- Microsoft Excel/CSV

5.5 Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

Components	Minimum	Recommended
Processor	Intel Core i3-6Gen,1.20GHz	Intel Core i5-10Gen, 2.10GHz
RAM	4GB	8GB
Camera	2 Mega-Pixel	8 Mega-Pixel
Disk	256 GB	516 GB
Network	1 MB/s	3 MB/s

Table3: Hardware Requirements

6. Detailed Design:

6.1 Project Use Case Diagram:

A use case diagram is a visual representation of how a system interacts with external entities, known as actors, to accomplish specific tasks or goals. It illustrates the functional requirements of a system from an external user's perspective, focusing on what the system does without detailing how it achieves these tasks. Actors can be users, other systems, or external entities. Use cases represent the system's functionalities or services, demonstrating various scenarios and interactions between actors and the system. This diagram provides a high-level overview of the system's behavior, aiding in understanding, communication, and analysis of the system's functionality and user interactions.

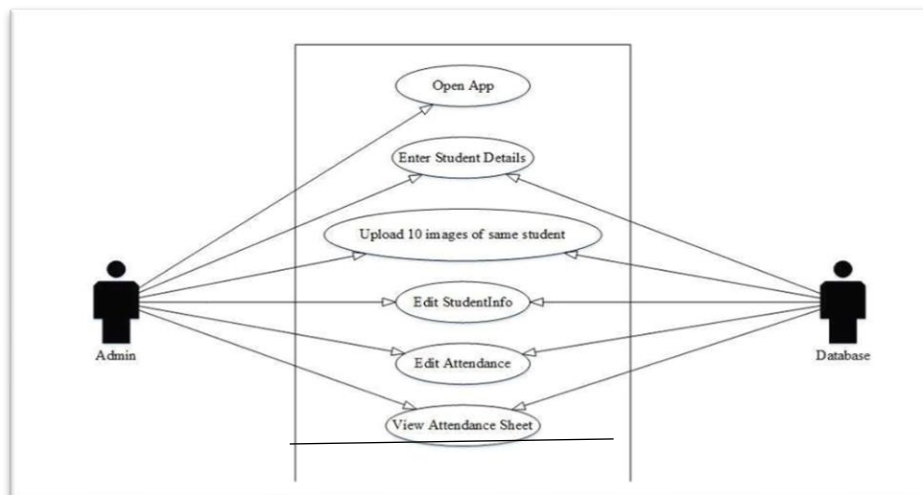


Figure 2: Usecase Diagram- Registration

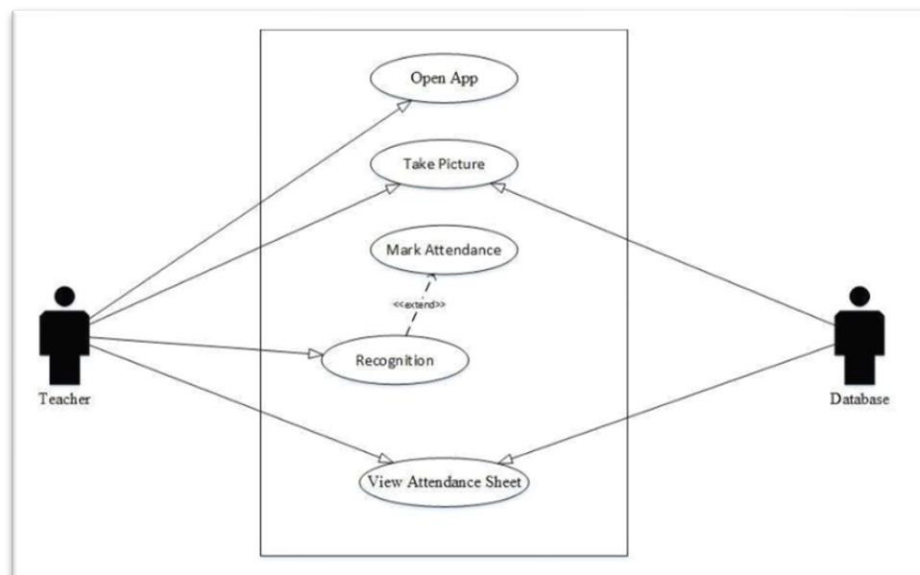


Figure 3: Usecase Diagram- Attendance

6.2 Project Block Diagram:

A block diagram is a graphical representation of a system or a process, using blocks to represent its different components and lines to indicate their relationships or interactions. It is a high-level representation of a complex system or process that breaks it down into simpler components and shows how they are connected or related to each other. The blocks in a block diagram can represent physical components such as hardware devices, subsystems, or software modules, as well as abstract concepts or processes. The lines between the blocks indicate the flow of data, signals, or other inputs and outputs between the components, and can be used to describe the logic or functionality of the system or process. Block diagrams are widely used in engineering, science, and technology to model and analyze complex systems or processes, and to communicate their design and operation to others in a clear and concise way.

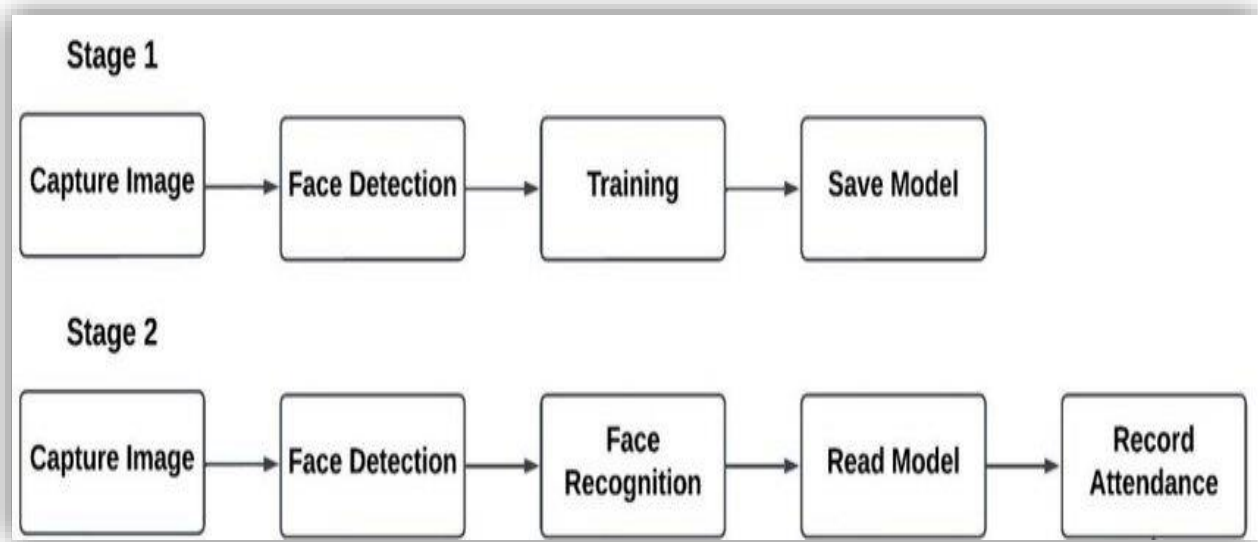


Figure 4: Project Block Diagram

6.3 Project Data Flow Diagram:

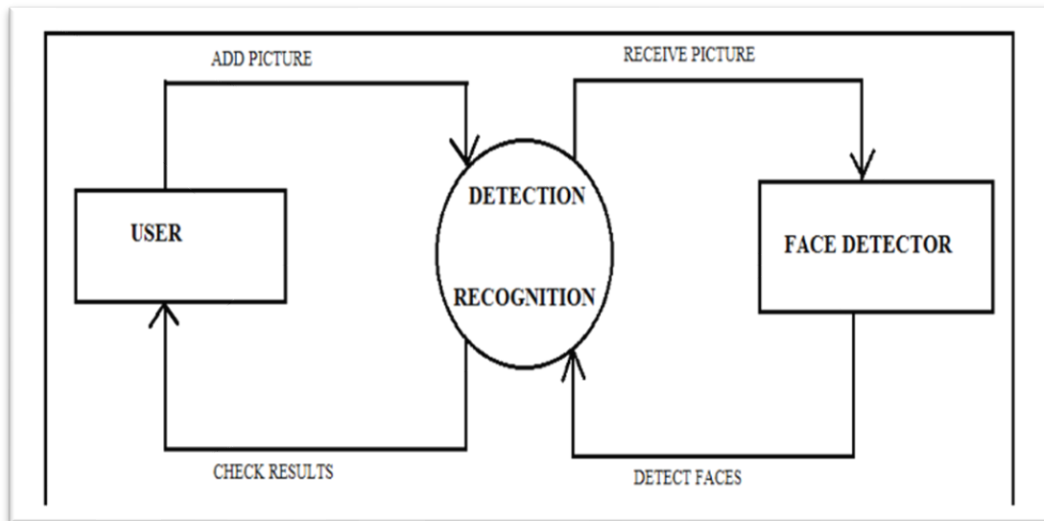


Figure 5: Data Flow Diagram

6.4 Project Flow Chart:

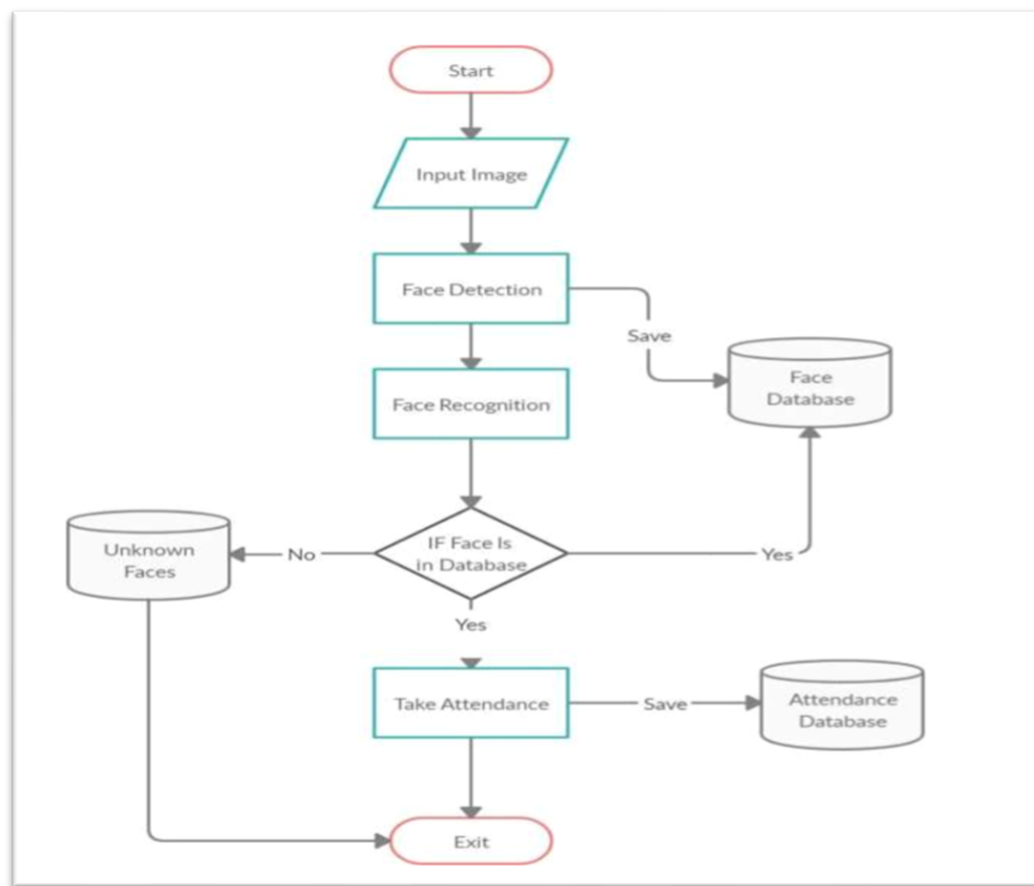


Figure 6: Project Flowchart

6.5 Dataset Creation Flowchart:



Figure 7: Dataset Creation Flowchart

6.6 Project E-R Diagram:

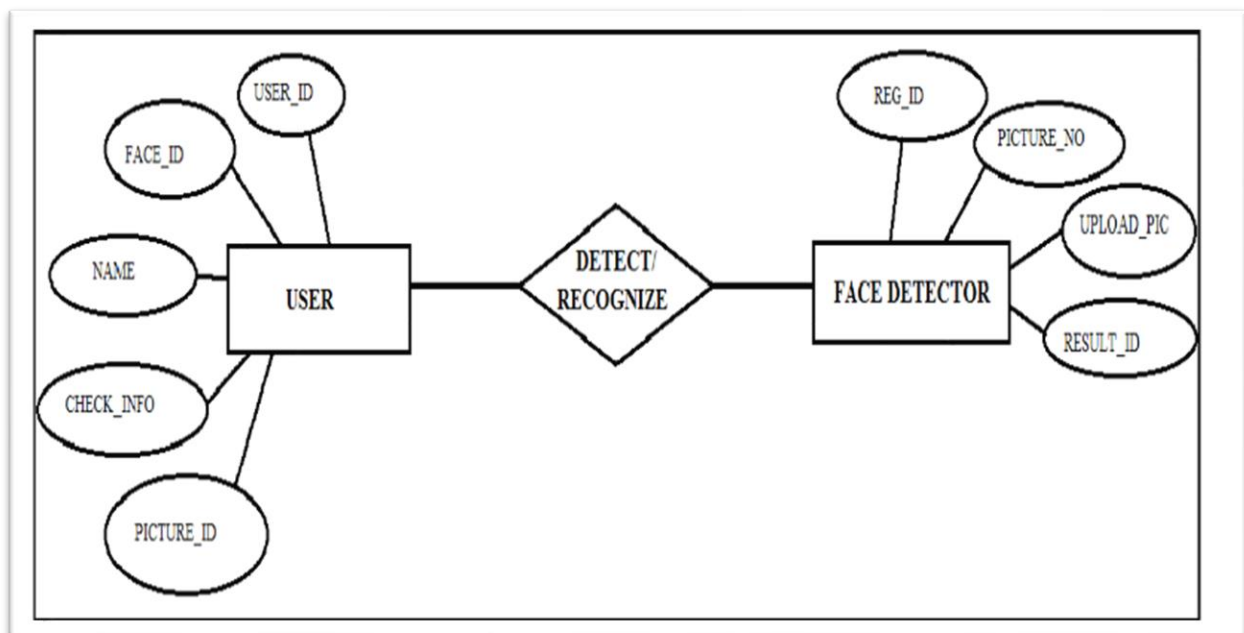


Figure 8: E-R Diagram

7. Required Algorithms:

7.1 Local Binary Pattern Histogram (LBPH) Algorithm:

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

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on Some datasets. Using the LBP combined with histogram we can represent the face images with a simple data vector.

LBPH algorithm work step by step:

LBPH algorithm work in 5 steps.

1. Parameters: the LBPH uses 4 parameters:

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

2. Training the Algorithm:

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

3. Applying the LBP Operation:

The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way. By highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameter's radius and neighbors.

The image below shows this procedure:

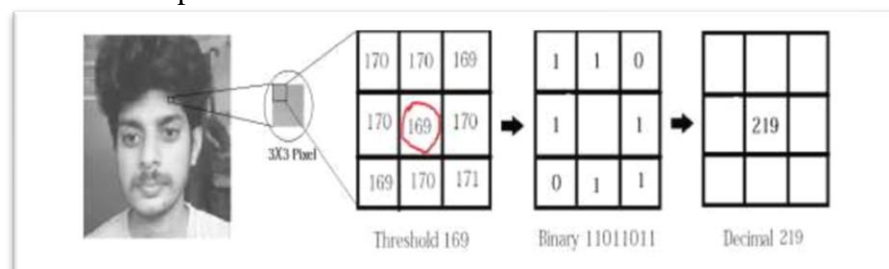


Figure 9: LBPH Operation

Based on the image above, let's break it into several small steps so we can understand it easily:

- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0-255)
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values form the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold a 0 for values lower than the threshold.
- Now the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

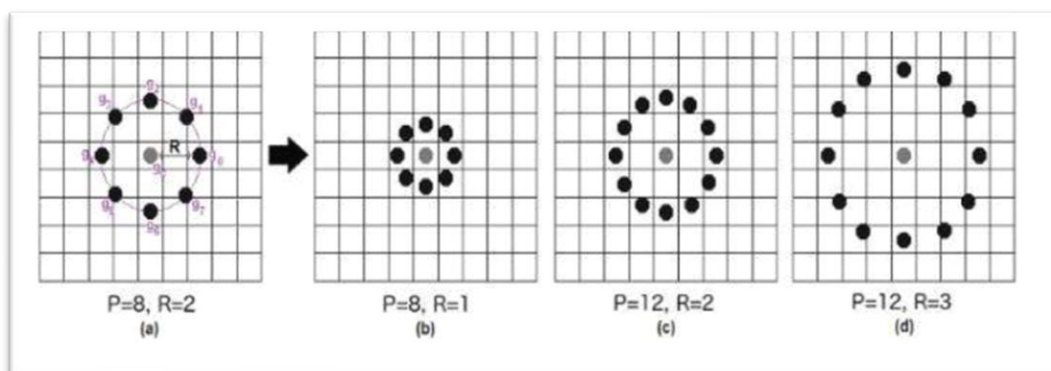


Figure 10: LBPH Operation Radius Change

It can be done by using bilinear interpolation. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4.Extraction the Histograms:

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

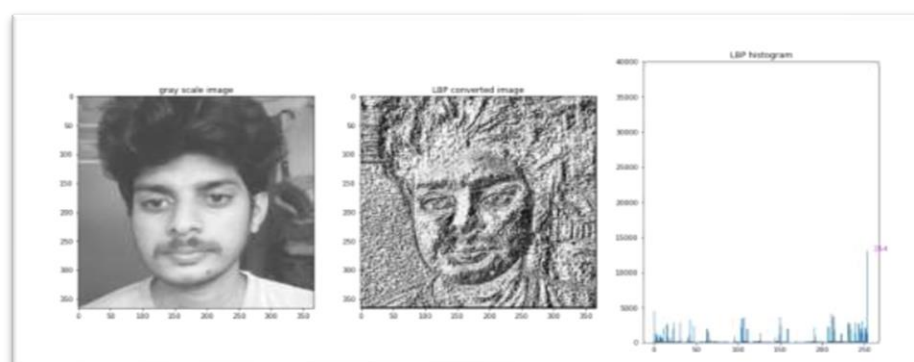


Figure 11: Extracting the Histogram

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

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0-255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have $8 \times 8 \times 256 = 16.384$ positions in the final histogram. The final histogram represents the characteristics of the image original image.

5.Performing the face recognition:

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

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

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

Figure 12: Euclidean Distance Formula

- So, the algorithm output is the ID from the image with the closet histogram. The algorithm should also return the calculate distance, which can be used as a ‘confidence’ measurement.
- We can then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

7.2 Haar Cascade Classifier:

The Haar cascade classifier is another computer vision algorithm that is commonly used for face detection, which is often a precursor to face recognition tasks.

1. Algorithm Basics:

The Haar cascade classifier uses Haar-like features, which are digital image features used in object recognition.

These features are reminiscent of the Haar wavelet and are used to detect the presence of objects (e.g., faces) in images.

2. Training:

A Haar cascade classifier is trained with a set of positive images (images of faces) and negative images (images without faces).

The training process involves selecting the most relevant features from a large set and using them to create a cascade of classifiers, each stage of the cascade filtering out non-face regions.



Figure 13: Image Training

3. Detection:

During detection, the classifier scans the input image at multiple scales and locations.

Each stage of the cascade classifies whether a region is likely to contain a face. If a region passes all stages, it is considered to contain a face.

4. Application in Face Recognition:

The Haar cascade classifier is often used to detect faces in an image before applying a face recognition algorithm like LBPH.

In practice, the image is first processed by the Haar cascade classifier to locate faces, and the detected face regions are then passed to the LBPH recognizer for identification.

8. Implementation:

8.1 System Installation:

8.1.1 VS-Code Installation:

Visual Studio Code is the most popular code editor and the IDEs provided by Microsoft for writing different programs and languages. It allows the users to develop new code bases for their applications and allow them to successfully optimize them.

For its high popularity, individuals opt to **Install Visual Studio Code on Windows** over any other IDE. **Installation of Windows Visual Studio Code** is not a difficult matter. The Installation process starts with **Downloading the Visual Studio Code EXE** file to some on-screen instructions.

Features of VS-Code:

- VS Code is a very user-friendly code editor and it is supported on all the different types of OS.
- It has support for all the languages like C, C++, Java, Python, JavaScript, React, Node JS, etc.
- It allows users with different types of in-app installed extensions.
- It allows the programmers to write the code with ease with the help of these extensions.
- Also, Visual Studio Code has a great vibrant software UI with amazing night mode features.
- It suggests auto-complete code to the users which suggests the users complete their code with full ease

Install Visual Studio Code on Windows:

- Download the VS Code file from the Official Website.
- Execute the download file.
- Accept the Terms & Conditions.
- Click on the Install button.
- Wait for the installation to complete.
- Click on the Launch button to start it.

8.1.2 Python - 3.10.0 Installation:

Python is a widely used high-level programming language. To write and execute code in python, we first need to install Python on our system.

Installing Python on Windows takes a series of few easy steps:

- Select Version of Python to Install.
- Download Python Executable Installer.
- Run Executable Installer.
 - We downloaded the Python 3.9.1 Windows 64-bit installer.

- Verify Python is installed on Windows.
 - To ensure if Python is successfully installed on your system. Follow the given steps –
 - Open the command prompt.
 - Type 'python' and press enter.
 - The version of the python which you have installed will be displayed if the python is successfully installed on your windows.
- Verify Pip was installed.

Pip is a powerful package management system for Python software packages. Thus, make sure that you have it installed.

- To verify if pip was installed, follow the given steps –
 - Open the command prompt.
 - Enter pip -V to check if pip was installed.
 - The following output appears if pip is installed successfully.

We have successfully installed python and pip on our Windows system.

8.2 Software Implementation:

8.2.1 Open-CV Installation:

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image patterns and its various features we use vector space and perform mathematical operations on these features. To install OpenCV, one must have Python and PIP, preinstalled on their system.

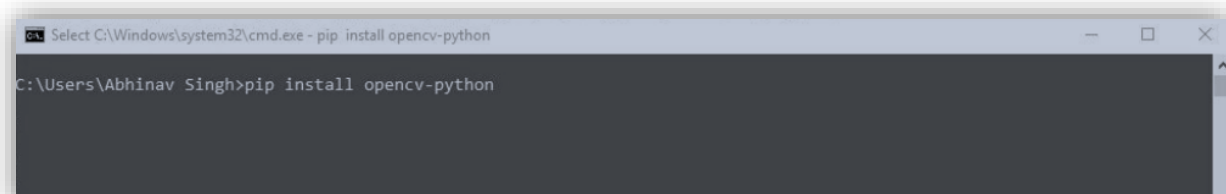
Downloading and Installing OpenCV:

OpenCV can be directly downloaded and installed with the use of pip (package manager). To install OpenCV, just go to the command-line and type the following command:

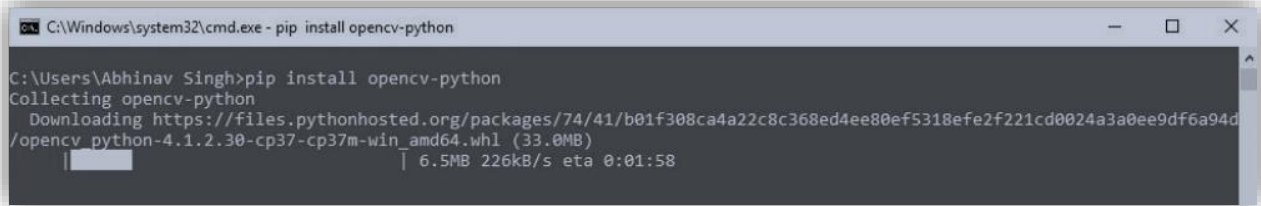
```
pip install opencv-python
```

Beginning with the installation:

- Type the command in the Terminal and proceed:



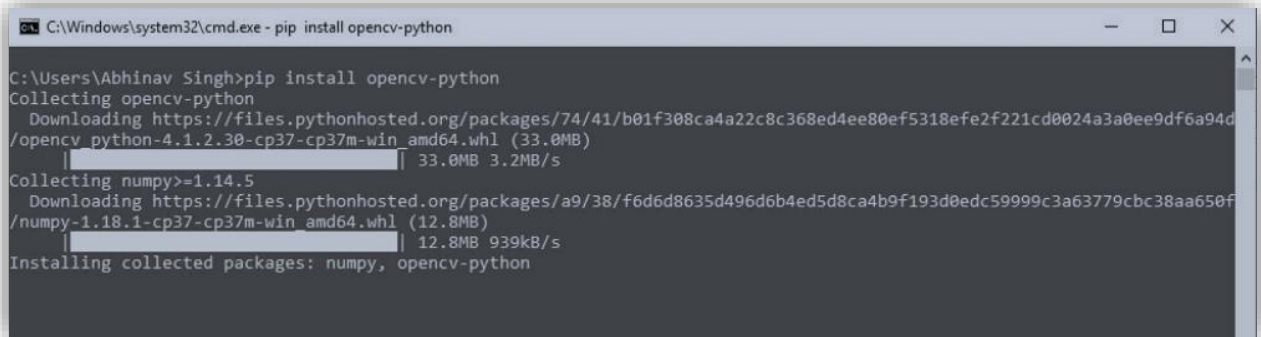
- Collecting Information and downloading data:



```
C:\Windows\system32\cmd.exe - pip install opencv-python

C:\Users\Abhinav Singh>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/74/41/b01f308ca4a22c8c368ed4ee80ef5318efe2f221cd0024a3a0ee9df6a94d/opencv_python-4.1.2.30-cp37-cp37m-win_amd64.whl (33.0MB)
    | 6.5MB 226kB/s eta 0:01:58
```

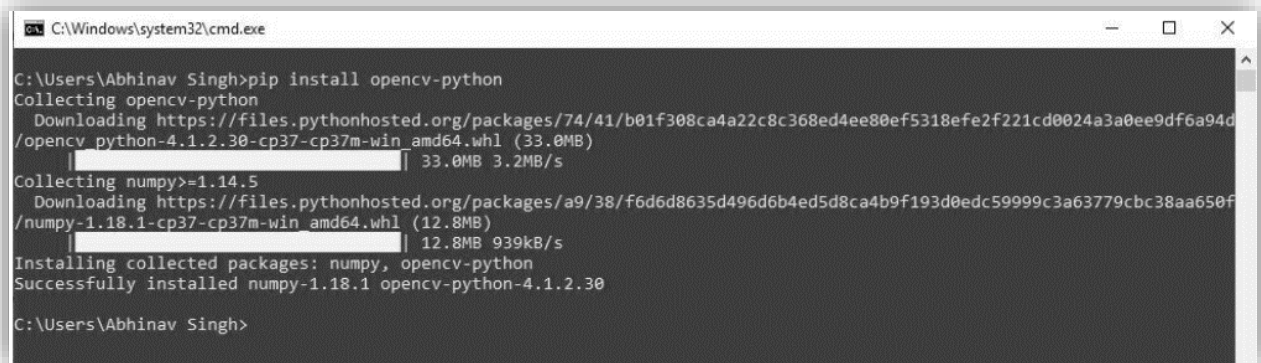
- Installing Packages:



```
C:\Windows\system32\cmd.exe - pip install opencv-python

C:\Users\Abhinav Singh>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/74/41/b01f308ca4a22c8c368ed4ee80ef5318efe2f221cd0024a3a0ee9df6a94d/opencv_python-4.1.2.30-cp37-cp37m-win_amd64.whl (33.0MB)
    | 33.0MB 3.2MB/s
Collecting numpy>=1.14.5
  Downloading https://files.pythonhosted.org/packages/a9/38/f6d6d8635d496d6b4ed5d8ca4b9f193d0edc59999c3a63779cbc38aa650f/numpy-1.18.1-cp37-cp37m-win_amd64.whl (12.8MB)
    | 12.8MB 939kB/s
Installing collected packages: numpy, opencv-python
```

- Finished Installation:



```
C:\Windows\system32\cmd.exe

C:\Users\Abhinav Singh>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/74/41/b01f308ca4a22c8c368ed4ee80ef5318efe2f221cd0024a3a0ee9df6a94d/opencv_python-4.1.2.30-cp37-cp37m-win_amd64.whl (33.0MB)
    | 33.0MB 3.2MB/s
Collecting numpy>=1.14.5
  Downloading https://files.pythonhosted.org/packages/a9/38/f6d6d8635d496d6b4ed5d8ca4b9f193d0edc59999c3a63779cbc38aa650f/numpy-1.18.1-cp37-cp37m-win_amd64.whl (12.8MB)
    | 12.8MB 939kB/s
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.18.1 opencv-python-4.1.2.30

C:\Users\Abhinav Singh>
```

8.2.2 Haar Cascade Implementation:

To implement Haar Cascade in Python, we will use the OpenCV library, which provides pre-trained Haar Cascade classifiers for facial detection. We will use the `cv2.CascadeClassifier` class to create a classifier, and the `detectMultiscale` method to detect faces in an image.

- Installing the Required Libraries

We will start by installing the required libraries. OpenCV and TensorFlow can be installed using `pip`. To install these libraries, open a terminal or command prompt and type the following commands:

```
sh

pip install opencv-python
```

- Implementing Haar Cascade for Face Detection

We will start with Haar Cascade and NumPy for more accurate detection. Let's start by importing the necessary libraries:

```
python

import cv2
import numpy as np
```

Next, we will create a *Cascade Classifier* object and load the pre-trained classifier for facial detection:

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

The `haarcascade_frontalface_default.xml` file contains the pre-trained classifier for facial detection.

Now, let's load an image and convert it to grayscale:

```
img = cv2.imread('face.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

We will use the *cv2.imread* method to load the image, and the *cv2.cvtColor* method to convert it to grayscale. Grayscale images are easier to process than color images, and they can improve the accuracy of facial detection.

Next, we will apply the detect Multiscale method to detect faces in the image:

```
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
```

The detect Multiscale method takes the grayscale image as input and returns a list of rectangular coordinates for each detected face. The scale Factor parameter controls the scale factor used to resize the image during the scanning process. The minNeighbors parameter controls the minimum number of neighboring windows required for a potential face to be considered a face.

Finally, we will draw rectangles around the detected faces:

```
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

The *cv2.rectangle* method is used to draw rectangles around each detected face. The (x, y) coordinate specifies the top-left corner of the rectangle, while the (x+w, y+h) coordinate specifies the bottom-right corner. The (0, 255, 0) parameter specifies the color of the rectangle, and the 2 parameter specifies the thickness of the rectangle.

Now we define the IMAGE_PATH and detect the faces using our function above.

```
# Define the path to the image
IMAGE_PATH = 'test_image.jpg'

# Detect faces in the image
detect_objects(IMAGE_PATH, detection_graph, category_index, min_score_thresh=.5)
```

8.2.3 Face_recognizer.yml Implementation:

The face_recognizer.yml file is a serialized representation of the trained face recognition model. Here's a detailed explanation of its role and contents:

Role of face_recognizer.yml:

1. Model Storage:

- The file stores the state of the trained Local Binary Patterns Histograms (LBPH) face recognizer model. This includes the learned parameters and data that the model needs to perform face recognition tasks.

2. Portability:

- By saving the trained model to a file, you can easily load it later without needing to retrain the model. This is particularly useful for deployment, sharing, or continuing training at a later time.

3. Efficiency:

- Loading a pre-trained model from a file is much faster than training the model from scratch every time it is needed. This improves the efficiency of applications that require face recognition

Contents of face_recognizer.yml:

The file contains various elements essential for the LBPH algorithm to function properly after loading. Here's what it typically includes:

1. Model Parameters:

- Parameters like radius, neighbors, grid_x, and grid_y which define the LBPH operator settings.

2. Histograms:

- The histograms of the Local Binary Patterns (LBP) computed for each face in the training set.

3. Labels:

- The mapping between labels and the corresponding histograms. This allows the model to recognize which histogram corresponds to which individual.

4. Metadata:

- Additional metadata that may include information about the training process, such as the number of training images and any preprocessing steps applied.

How the File is Used

1. Saving the Model:

```
python
model.save('face_recognizer.yml')
```

This command writes the trained model's data into the 'face_recognizer.yml' file.

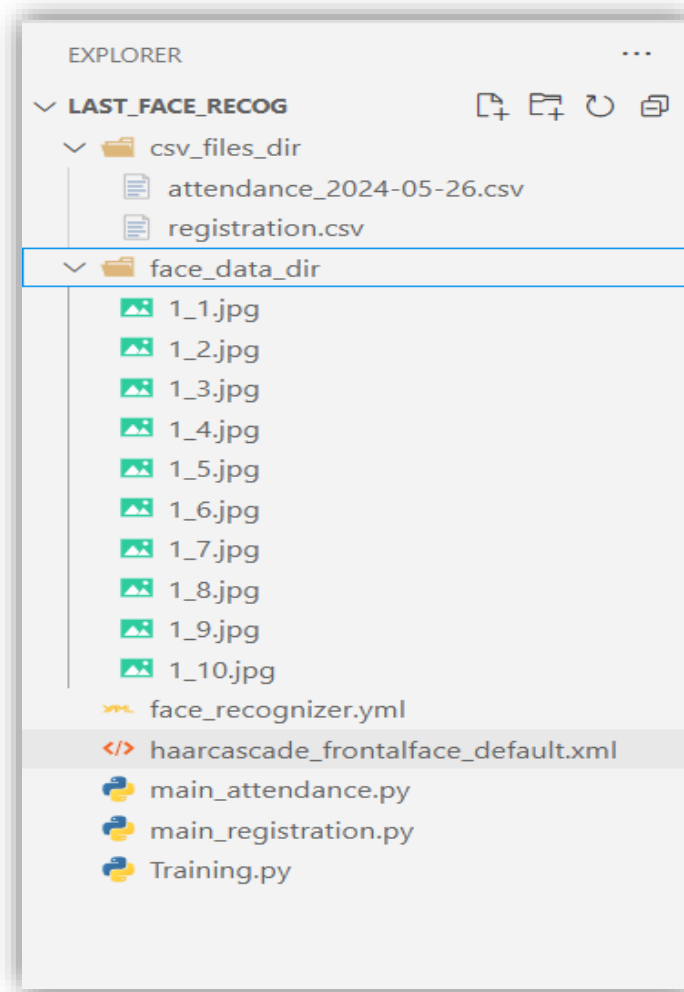
2. Loading the Model:

```
python
model = cv2.face.LBPHFaceRecognizer_create()
model.read('face_recognizer.yml')
```

This command reads the model data from the file, allowing you to use the trained model without retraining.

8.3 Code Implementation & Results:

All our code is written in Python language. First here is our project directory structure and files-



All those files shown above are in the project directory.

[CSV_files_dir] => It stores all CSV files (registration.csv and attendance.csv) inside the folder.

[face_data_dir] => It stores all captured photos inside the folder.

[main_registration.py] => It's a python code that registered students details & take photos of that student.

[haarcascade_frontalface_default.xml] => It's a pre-trained classifier file used for detecting faces in images.

[Training.py] => It's a python code that trained the photos.

[face_rocognizer.yml] => It's a file that is serialized representation of the trained face recognition model.

[main_attendance.py] => It's a python code for recognition and mark attendance.

8.3.1 main_registration.py

This Python code performs several tasks, including registering a student's details, capturing their face data, and storing both the details and face data for future use.

```
1 import csv
2 import cv2
3 import os
4 from datetime import datetime
5
6 # Directories for storing CSV files and face data
7 csv_file_dir = 'csv_files_dir'
8 face_data_dir = 'face_data_dir'
9
10 # Create the directories if they don't exist
11 os.makedirs(csv_file_dir, exist_ok=True)
12 os.makedirs(face_data_dir, exist_ok=True)
13
14 # Path for the CSV file
15 csv_file_path = os.path.join(csv_file_dir, 'registration.csv')
16
17 # Check if the CSV file exists, if not, create it with headers
18 if not os.path.exists(csv_file_path):
19     with open(csv_file_path, mode='w', newline='') as file:
20         writer = csv.writer(file)
21         writer.writerow(["ID", "Student Name", "Department", "Year of Admission", "Father Name", "Address", "Phone Number", "Email ID"])
22
23 # Input student details from the user
24 student_id = input("Enter Student ID: ")
25 student_name = input("Enter Student Name: ")
26 department = input("Enter Department: ").upper()
27 year_of_admission = input("Enter Year of Admission (4 digit integer): ")
28 father_name = input("Enter Father's Name: ")
29 address = input("Enter Address: ")
30 phone_number = input("Enter Phone Number (10 digit integer): ")
31 email_id = input("Enter Email ID: ").lower()
32
33 # Append the student details to the CSV file
34 with open(csv_file_path, mode='a', newline='') as file:
35     writer = csv.writer(file)
36     writer.writerow([student_id, student_name, department, year_of_admission, father_name, address, phone_number, email_id])
37
38 # Display the student information
39 print("\n-----\n")
40 print("Student Information:")
41 print("ID:", student_id)
42 print("Student Name:", student_name)
43 print("Department:", department)
44 print("Year of Admission:", year_of_admission)
45 print("Father's Name:", father_name)
46 print("Address:", address)
47 print("Phone Number:", phone_number)
48 print("Email ID:", email_id)
49 print("\nStudent registered successfully.\n")
50 print("\n-----\n")
```



```

51
52 # Wait for user input to start face capture
53 while True:
54     digit = int(input("Enter 1 to capture face: "))
55     if digit == 1:
56         print("Face recognition started.")
57         break
58     else:
59         continue
60
61 # Initialize the face classifier
62 face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
63
64 # Function to extract faces from an image
65 def face_extractor(img):
66     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
67     faces = face_classifier.detectMultiScale(gray, 1.3, 5)
68
69     if len(faces) == 0:
70         return None, None
71
72     for (x, y, w, h) in faces:
73         cropped_face = img[y:y+h, x:x+w]
74         return cropped_face, (x, y, w, h)
75
76     return None, None
77
78 # Start video capture
79 cap = cv2.VideoCapture(0)
80 count = 0
81
82 while True:
83     ret, frame = cap.read()
84     face, rect = face_extractor(frame)
85
86     if face is not None:
87         count += 1
88         face = cv2.resize(face, (350, 350)) # Resize the face image
89         face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY) # Convert to grayscale
90
91         # Save the face image
92         file_name_path = os.path.join(face_data_dir, f'{student_id}_{count}.jpg')
93         cv2.imwrite(file_name_path, face)
94
95         if rect:
96             x, y, w, h = rect
97             cv2.putText(frame, f'Count: {count}', (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
98             cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
99
100         cv2.imshow('Capture Facedata', frame)
101
102     else:
103         print("Face not found.")
104         pass
105
106     # Exit the loop if 'Enter' key is pressed or 100 faces are captured
107     if cv2.waitKey(1) == 13 or count == 100:
108         break
109
110 # Release the video capture and close the windows
111 cap.release()
112 cv2.destroyAllWindows()
113 print('Face data collection completed.')

```

Console Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS E:\Workspace\Clg_Project\last_face_recog\last_face_recog> python -u "e:\Workspace\Clg_Project\last_face_recog\last_face_recog\main_registration.py"
Enter Student ID: 1
Enter Student Name: Sandip Karmakar
Enter Department: CSE
Enter Year of Admission (4 digit integer): 2021
Enter Father's Name: Dilip Karmakar
Enter Address: Bishnupur
Enter Phone Number (10 digit integer): 9064176331
Enter Email ID: abc@gmail.com

-----

Student Information:
ID: 1
Student Name: Sandip Karmakar
Department: CSE
Year of Admission: 2021
Father's Name: Dilip Karmakar
Address: Bishnupur
Phone Number: 9064176331
Email ID: abc@gmail.com

Student registered successfully.

-----

Enter 1 to capture face: 1
Face recognition started.
Face data collection completed.
PS E:\Workspace\Clg_Project\last_face_recog\last_face_recog> []
```

Face Detection:

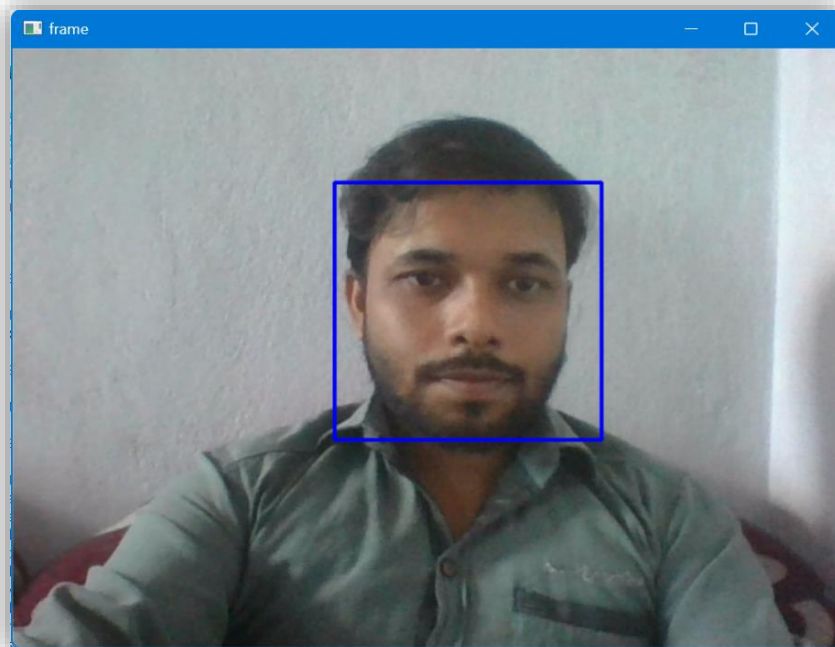


Figure 14: Face Detection

- Capturing cropped images of the student and stored into the folder named- [face_data_dir]

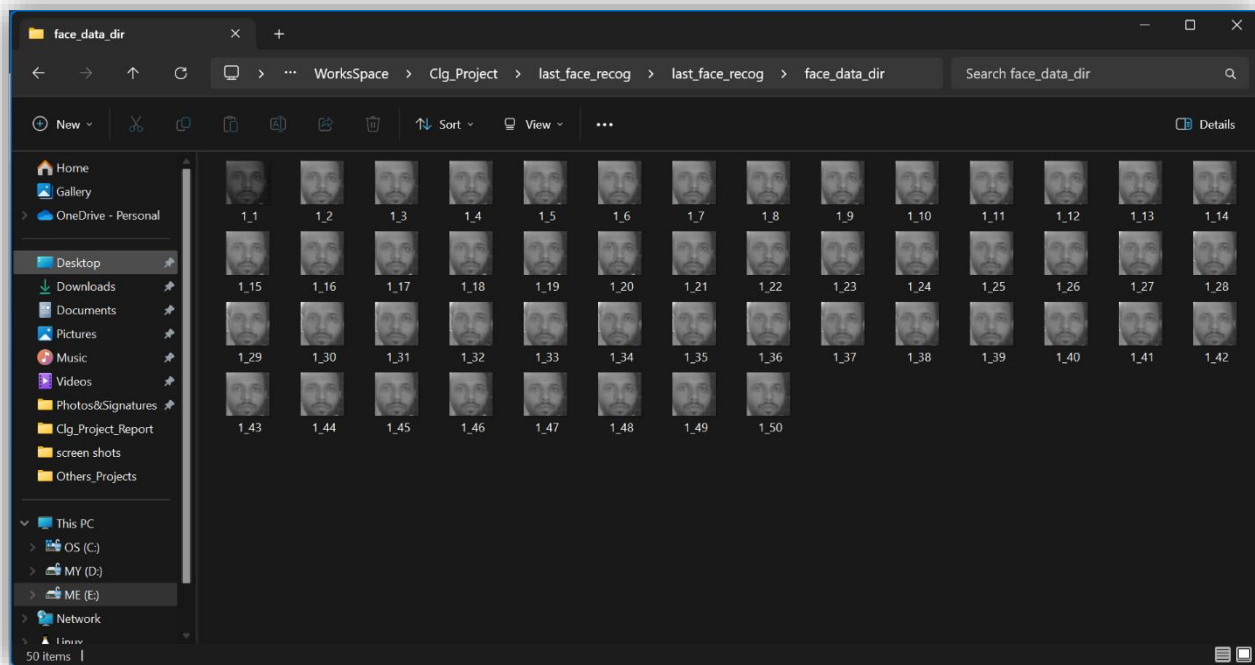


Figure 15: Dataset Sample

- And entered student details are saved as a .CSV format into the folder name- [CSV_files_dir].

registration - Excel

Sandip Karmakar

ID	Student Name	Department	Year of Admission	Father Name	Address	Phone Number	Email ID
1	Sandip Karmakar	CSE	2021	Dilip Karmakar	Bishnupur	9064176331	abc@gmail.com
2	Soumyadeep Karmakar	CSE	2021	Ranjit Karmakar	Bishnupur	8145728664	sk@gmail.com
3	Pragya Jalan	CSE	2021	Bijoy Jalan	Bishnupur	8972356049	pj@gmail.com

Figure 16: Registration Datasheet

8.3.2 Training.py:

This python code performs the task of training a face recognition model that is stored in a specified directory using the Local Binary Patterns Histograms (LBPH) algorithm in OpenCV.

```
1 import cv2
2 import numpy as np
3 import os
4
5 # Set the path to the dataset containing facial images for training
6 data_path = 'face_data_dir/'
7
8 # Get all file names in the directory and filter out the ones that are not image files
9 onlyfiles = [f for f in os.listdir(data_path) if os.path.isfile(os.path.join(data_path, f))]
10
11 # Initialize empty lists for training data and labels
12 Training_Data, Labels = [], []
13
14 # Loop through each image file, read the image in grayscale, and append it to the Training_Data list
15 # Append the index of the image file to the Labels list
16 for i, files in enumerate(onlyfiles):
17     image_path = os.path.join(data_path, files)
18     images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
19     if images is not None:
20         # Check if image reading is successful
21         Training_Data.append(np.asarray(images, dtype=np.uint8))
22         Labels.append(int(files.split('_')[0]))
23     else:
24         print(f"Failed to read image: {image_path}")
25
26 # Check if any images were read successfully
27 if len(Training_Data) == 0:
28     print("No training data available. Please check the dataset directory.")
29     exit()
30
31 # Convert the Labels list to a numpy array of type int32
32 Labels = np.asarray(Labels, dtype=np.int32)
33
34 # Create a new LBPH face recognition model
35 model = cv2.face.LBPHFaceRecognizer_create()
36
37 # Train the model using the training data and labels
38 model.train(np.asarray(Training_Data), np.asarray(Labels))
39
40 # Save the trained model
41 model.save('face_recognizer.yml')
42
43 print("Dataset Model Training Completed")
44
```

Console Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

```
PS E:\Workspace\Clg_Project\last_face_recog\last_face_recog> python -u "e:\Workspace\Clg_Project\last_face_recog\last_face_recog\tempCodeRunnerFile.py"
Dataset Model Training Completed
PS E:\Workspace\Clg_Project\last_face_recog\last_face_recog> █
```

8.3.3 Main_attendance.py:

```
1 import cv2
2 import csv
3 import os
4 import numpy as np
5 from datetime import datetime
6
7 # Load the pre-trained face recognizer
8 face_recognizer = cv2.face.LBPHFaceRecognizer_create()
9 face_recognizer.read('face_recognizer.yml')
10
11 # Load CSV file containing student information
12 csv_dir = 'csv_files_dir' # Directory where CSV files are stored
13 csv_file = os.path.join(csv_dir, 'registration.csv') # Path to the CSV file with student registration data
14 attendance_dir = csv_dir # Directory where attendance CSV files will be saved
15 student_data = {} # Dictionary to hold student information
16
17 # Ensure the CSV directory exists
18 if not os.path.exists(csv_dir):
19     os.makedirs(csv_dir)
20
21 # Read student data from the CSV file
22 with open(csv_file, mode='r') as file:
23     reader = csv.reader(file)
24     next(reader) # Skip the header row
25     for row in reader:
26         student_id = row[0]
27         student_name = row[1]
28         department = row[2]
29         student_data[student_id] = {"name": student_name, "department": department}
30
31 # Debugging print to check loaded student data
32 print("Loaded student data:", student_data)
33
34 def mark_attendance(student_id, date, time):
35     """
36     Mark the attendance of a student.
37
38     Args:
39     student_id (str): The ID of the student.
40     date (str): The current date.
41     time (str): The current time.
42
43     Returns:
44     str: Status message indicating whether the attendance was marked or already marked.
45     """
46     attendance_csv = os.path.join(attendance_dir, f'attendance_{date}.csv') # Path to the attendance CSV file
47     headers = ["ID", "Student Name", "Department", "Attendance", "Date", "Time"]
48
49     # Create the attendance CSV file if it doesn't exist and write the header
50     if not os.path.exists(attendance_csv):
51         with open(attendance_csv, mode='w', newline='') as file:
52             writer = csv.writer(file)
53             writer.writerow(headers)
54
55     # Check if attendance has already been marked for the student
56     with open(attendance_csv, mode='r') as file:
57         reader = csv.reader(file)
58         next(reader) # Skip the header row
59         for row in reader:
```

```

60         if row[0] == student_id:
61             return "Attendance Marked Already"
62
63     # Append the attendance record to the CSV file
64     with open(attendance_csv, mode='a', newline='') as file:
65         writer = csv.writer(file)
66         writer.writerow([student_id, student_data[student_id]["name"], student_data[student_id]["department"], 'Present', date, time])
67     return "Attendance Marked"
68
69 def recognize_faces():
70     """
71     Recognize faces in real-time using a webcam and mark attendance.
72     """
73     cap = cv2.VideoCapture(0) # Open the default camera
74     face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml') # Load the face detection model
75
76     while True:
77         ret, frame = cap.read() # Capture frame-by-frame
78         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # Convert the frame to grayscale
79         faces = face_classifier.detectMultiScale(gray, 1.3, 5) # Detect faces in the frame
80
81         for (x, y, w, h) in faces:
82             face_roi = gray[y:y+h, x:x+w] # Extract the region of interest (face)
83             label, confidence = face_recognizer.predict(face_roi) # Recognize the face
84
85             if confidence < 50: # If confidence is less than 50, consider it a recognized face
86                 student_id = str(label)
87                 if student_id in student_data:
88                     student_name = student_data[student_id]["name"]
89                     department = student_data[student_id]["department"]
90                     date = datetime.now().strftime("%Y-%m-%d")
91                     time = datetime.now().strftime("%H:%M:%S")
92
93                     # Mark attendance
94                     attendance_status = mark_attendance(student_id, date, time)
95
96                     if attendance_status == "Attendance Marked ":
97                         status_text = attendance_status
98                         status_color = (0, 0, 255) # Red color for already marked
99                     else:
100                         status_text = "Attendance Marked Already"
101                         status_color = (0, 255, 0) # Green color for newly marked
102
103                     # Draw a rectangle around the face and display the confidence level
104                     cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
105                     cv2.putText(frame, f"Confidence: {int(confidence)}%", (x, y+h+20), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1)
106                 else:
107                     # Debugging print to check for unknown IDs
108                     print(f"Unknown student ID: {student_id}")
109                     cv2.putText(frame, "Unknown ID", (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 1)
110                     cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
111             else:
112                 # If face is not recognized, mark it as unknown
113                 cv2.putText(frame, "Unknown", (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 1)
114                 cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
115
116     # Create a white info frame to display student details
117     height, width, _ = frame.shape
118     info_frame = 255 * np.ones(shape=[height, 600, 3], dtype=np.uint8)
119
120     # Display student details on the info frame
121     if 'student_id' in locals() and student_id in student_data:
122         cv2.putText(info_frame, "Student Details", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 2)
123         cv2.putText(info_frame, f"ID: {student_id}", (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 1)
124         cv2.putText(info_frame, f"Name: {student_name}", (10, 110), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 1)
125         cv2.putText(info_frame, f"Department: {department}", (10, 150), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 1)
126         cv2.putText(info_frame, f>Status: {status_text}", (10, 190), cv2.FONT_HERSHEY_SIMPLEX, 0.8, status_color, 1)
127
128     # Combine the main frame with the info frame
129     combined_frame = np.hstack((frame, info_frame))
130     cv2.imshow('Attendance System', combined_frame) # Display the combined frame
131
132     if cv2.waitKey(1) == 27: # Press 'Esc' to exit
133         break

```

```

134
135     cap.release() # Release the capture
136     cv2.destroyAllWindows() # Close all OpenCV windows
137
138 if __name__ == "__main__":
139     recognize_faces() # Run the face recognition and attendance marking system
140

```

Console Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

Code + - X

```

PS E:\Workspace\Clg_Project\last_face_recog\last_face_recog> python -u "e:\Workspace\Clg_Project\last_face_recog\last_face_recog\main_attendance.py"
Loaded student data: {'1': {'name': 'Sandip Karmakar', 'department': 'CSE'}, '2': {'name': 'Soumyadeep Karmakar', 'department': 'CSE'}, '3': {'name': 'Pragya Jalan', 'department': 'CSE'}}
PS E:\Workspace\Clg_Project\last_face_recog\last_face_recog>

```

Face Recognize:

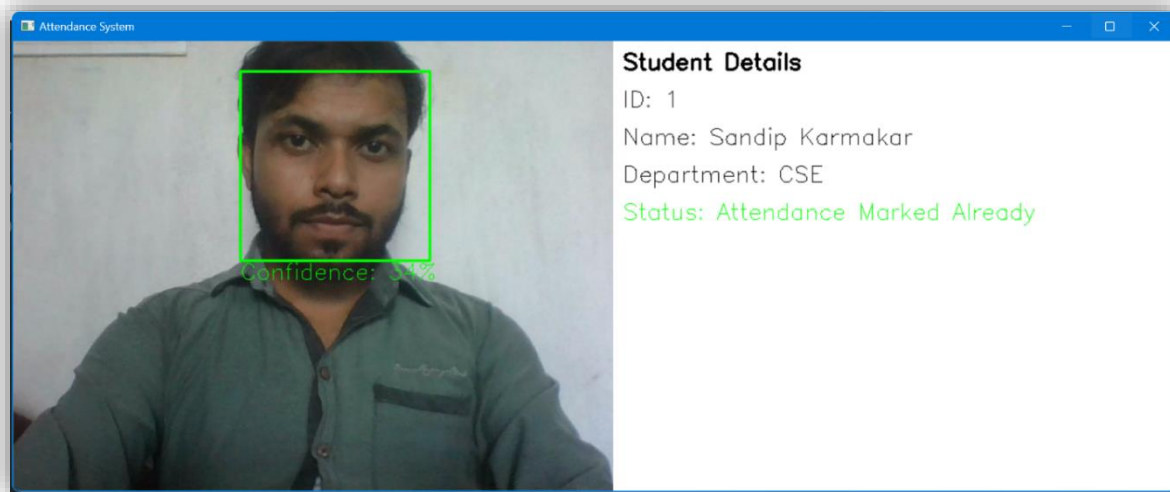


Figure 17: Face Recognize

- And attendance of the student is saved as .CSV format file into the folder name- [CSV_files_dir].

ID	Student Name	Department	Attendance	Date	Time
1	Sandip Karmakar	CSE	Present	27-05-2024	15:28:35

Figure 18: Generated Attendance Datasheet

Project GitHub Link - https://github.com/Sandip-10/Face_Recognition_Attendance_System-.git

9. Conclusion:

In order to maintain the attendance this system has been proposed. It replaces the manual system with an automated system which is fast, efficient, cost and time saving as replaces the stationary material and the paper work. Hence this system is expected to give desired results and in future could be implemented for logout. Also, the efficiency could be improved by integrating other techniques with it in near future.

9.1 Project Benefits:

This project proposes a facial recognition system for attendance tracking in schools and organizations. It uses OpenCV and Face Recognition libraries to identify people through webcams. The system offers a user-friendly interface and utilizes Firebase for secure data storage. Key features include:

- Real-time face detection and recognition
- Contactless attendance marking
- Multiple attendance prevention
- Alerts for unrecognized faces
- Machine learning for improved accuracy over time
- Integration with potential mobile applications
- The project prioritizes user privacy with secure data transmission and encryption. It aims to be adaptable and integrate with future biometric authentication methods. Overall, this project offers a comprehensive and future-proof solution for attendance management.

9.2 Future Scope for improvements:

The system can be made more flexible and scalable using these recommendations. Please note that the system implemented here is just a prototype of idea presented via this project. The recommendations are as follows:

- The system can be extended to a greater number of students with freedom to change list of students according to class changes.
- The system can be made more flexible to allow updating of templates in case student incurs significant amount of change in his facial features.
- The system can also be extended to allow better face recognition algorithm in which even rotational features of face can be detected efficiently.

10. References / Bibliography:

[1]. Prado, K. S. D. (2018, June 19). Face Recognition: Understanding LBPH Algorithm -

Towards Data Science. *Medium*.

<https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>

[2]. Gillis, A. S. (2024, May 3). *facial recognition*. Enterprise AI.

<https://www.techtarget.com/searchenterpriseai/definition/facial-recognition>

[3]. Reference YouTube Channels:

- Murtaza's Workshop - Robotics and AI. (2022, December 13). *Face Recognition with Real Time Database / 2 Hour Course / Computer Vision* [Video]. YouTube.

<https://www.youtube.com/watch?v=iBomaK2ARyI>

- Computer Vision Engineer. (2022, December 21). *Face recognition + liveness detection: Face attendance system* [Video]. YouTube.

https://www.youtube.com/watch?v=_KvtVk8Gk1A

[4]. Reference Websites Links:

Face Recognition with Real-Time Database - Computer Vision Zone. (2022, December 13).

Computer Vision Zone.

<https://www.computervision.zone/courses/face-recognition-with-real-time-database/>

[5]. K, M. (2023, January 9). *SMART ATTENDANCE SYSTEM USING FACE RECOGNITION*

TECHNIQUES.

<https://www.linkedin.com/pulse/smart-attendance-system-using-face-recognition-mr-statistician/>

[6]. Reference GitHub Link:

https://github.com/debashischatterjee1/Automatic_attendence_system_using_facial_recognition_python_openCV