Facial Recognition Attendance Using LBPH Algorithm

A PROJECT REPORT

Submitted by:

20BCS6140 - Gajula Ajay

20BCS4467 – P. Vinay Kumar

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE WITH SPECIALIZATION IN

BIG DATA ANALYTICS

Under the Supervision of:

Ms. Mansi Kajal (E14871)

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

January 2024

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report **“Facial Recognition Attendance Using LBPH Algorithm”** is the bonafide work of “**Gajula Ajay, P. Vinay Kumar”** who carried out the project work under my/our supervision.

SIGNATURE	**SIGNATURE**
	Ms. Mansi Kajal
HEAD OF THE DEPARTMENT	**SUPERVISOR**

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

List of Figures

List of Tables

Table of Contents

Abstract

The implementation of a Facial Recognition System can aid in identifying or verifying a person's identity from a digital image. Accurate attendance records are vital to classroom evaluation. However, manual attendance tracking can result in errors, missed students, or duplicate entries. The adoption of the Face Recognition-based attendance system could help eliminate these shortcomings. This innovative approach involves utilizing a camera to capture input images, detecting faces using algorithms such as Haar cascade, Eigen values, support vector machines, or the Fisher face algorithm, verifying the faces against a database of student profiles, and marking attendance in an Excel sheet. The use of OpenCV, an open-source computer vision library, ensures the efficient functioning of the system. The proposed model involves training the system with the authorized students' faces to create a database. The system crops and stores the images in a database with corresponding labels and extracts features using algorithms such as LBPH, Eigen values, support vector machines and Fisher face algorithm. The Face Recognition-based attendance system could help automate attendance records with high accuracy and reduce the burden of manual attendance tracking.

1. INTRODUCTION

In today's competitive world, with increasing working hours and less classroom time, teachers need EdTech tools which help them manage precious class time efficiently. Instead of focusing on teaching, faculty members are often stuck with completing formal duties, for e.g., taking daily student attendance. Manually taking attendance and registering it in files & musters makes the daily attendance a mundane task for the faculty and unnecessarily consumes classroom time. To overcome such inefficient work processes, there are various school software systems available to speed up the attendance process and reduce manual work. An online attendance management system or digital attendance platform is one of them, which is developed to automate the daily attendance in schools. Additionally, it helps to maintain accurate records and generate summarized student attendance reports. Attendance management system keeps track of daily attendance, working hours, breaks, login, and logout time. It prevents staff's time theft. An attendance management system integrates all attendance devices such as smart cards, biometric, and facial recognition devices in real-time. Master Soft’s Student attendance management software allows schools of all sizes to manage various attendance requirements. This system makes it easy to create daily attendance reports, absentee lists, letters and other documents almost effortlessly. Student attendance system helps teachers to mark online attendance of students during class & reduce manual work. It is used to track student's attendance, absentee

record, attendance history & other related documents. Student attendance software allows you to record & manage daily student attendance to speed up the daily attendance process. Online attendance management system enables school administrators to record, manage & compile daily student attendance data. Along with student attendance, this software also allows teachers to generate 100% accurate student attendance reports.

Purpose: - The purpose of creating an attendance management system is to automate the traditional method of taking attendance. Another reason for designing this programme is to automatically generate the report at the end of the session or in the middle of it.

1.1 Problem Definition:

In fact, facial recognition-based attendance systems provide a cutting-edge way to simplify the classroom attendance procedure. These systems use sophisticated information technology and high-definition video to identify and effectively record the presence of pupils through facial recognition algorithms.

The basic idea behind face recognition is to give a computer system the capacity to quickly and correctly recognize faces in pictures or videos. Numerous algorithms and methods have been developed over time by researchers to improve facial recognition system performance. Deep learning has become one of these most effective tools, especially for computer vision applications. Convolutional neural networks (CNNs), one type of deep learning technique, have shown impressive results in facial recognition challenges. CNNs are able to identify distinct people with a high degree of accuracy by learning complex patterns and features inside facial photos through training on enormous volumes of labelled data.

Compared to conventional techniques, these face recognition systems based on deep learning provide a number of benefits. They are resilient in real-life situations because they can adjust to different lighting conditions, face emotions, and stances. Furthermore, with more training data, deep learning models can perform better and better over time, increasing their accuracy.

Putting in place a facial recognition-based attendance system in the classroom can help to improve efficiency, lower the risk of mistakes or oversights, and offer insightful data on attendance trends. Additionally, it can save time for both educators and learners, enabling more effective use of the resources in the classroom.

When implementing such systems, it's crucial to take things like data security and privacy concerns into account. Preserving trust and faith in the technology necessitates adhering to pertinent legislation and putting in place suitable measures for data protection.

1.2 Problem Overview

The Attendance Management System you described sounds like a valuable tool for educational institutions, offering streamlined attendance tracking and reporting capabilities. By automating the attendance process and leveraging face recognition technology, it addresses the limitations of manual methods, such as human errors and time consumption.

With face recognition integrated into the system, the process becomes even more efficient and accurate. Instead of relying solely on manual entry or card swiping, students' attendance can be verified automatically through facial recognition, reducing the chances of errors and ensuring the integrity of the data.

Deep learning, particularly convolutional neural networks (CNNs), has revolutionized the field of computer vision, including face recognition. By training on large datasets, deep learning models can learn intricate facial features and patterns, enabling them to identify individuals with high accuracy even in varying conditions such as lighting, facial expressions, and angles.

Integrating deep learning-based face recognition into the Attendance Management System enhances its capabilities, allowing for fast and precise identification of students. With just a click, teachers or administrators can access comprehensive attendance reports, enabling them to monitor attendance trends, evaluate eligibility criteria, and make informed decisions.

Overall, the combination of Attendance Management System and face recognition technology powered by deep learning offers a powerful solution for educational institutions to efficiently manage attendance processes, minimize manual effort, and improve data accuracy.

Hardware Specifications:

Laptop: A portable computer device capable of running software applications.

Operating System (OS): Windows

An operating system is software that manages computer hardware and provides common services for computer programs. Windows is a widely used operating system developed by Microsoft Corporation, known for its user-friendly interface and support for various applications and devices.

System Type: 64-bit Operating System, x64-based Processor

OS Build: 22000.675 (The OS build number represents the specific version and updates applied to the Windows operating system.)

Software Specifications:

Anaconda Navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included with the Anaconda distribution, which is a popular platform for scientific computing and data science tasks in Python. Anaconda Navigator provides an interface for managing environments, packages, and applications, making it easier to work with Python and its scientific libraries.

Jupyter Notebook:

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is widely used for interactive computing and data analysis in various fields, including data science, machine learning, and research

Python 3:

Python is a high-level programming language known for its simplicity and readability. Python 3 is the latest major version of Python, offering improvements and new features over previous versions.

It is widely used for various applications, including web development, scripting, automation, and scientific computing.

OpenCV (Open-Source Computer Vision Library):

OpenCV is an open-source computer vision and machine learning software library designed for real-time image processing and computer vision tasks. It provides a wide range of functions and algorithms for tasks such as object detection, face recognition, image segmentation, and feature extraction.

Microsoft Excel:

Microsoft Excel is a spreadsheet application developed by Microsoft Corporation, widely used for data analysis, calculation, and visualization. It provides a user-friendly interface for creating, editing, and analysing data in tabular format, making it suitable for various business, academic, and personal tasks.

Tkinter:

Tkinter is the standard GUI (Graphical User Interface) toolkit for Python, providing a set of built-in modules for creating and managing graphical user interfaces in Python applications. It allows developers to create windows, buttons, labels, entry fields, and other GUI components to build interactive applications with ease.

2. LITERATURE SURVEY

2.1 Existing System

Face recognition is one of the few biometric methods that possess the merits of both high accuracy and low intrusiveness. It has the accuracy of a physiological approach without being intrusive. Over past 30 years, many researchers have proposed different face recognition techniques, motivated by the increased number of real-world applications requiring the recognition of human faces. There are several problems that make automatic face recognition a very difficult task. However, the face image of a person inputs to the database that is usually acquired under different conditions. The important of automatic face recognition is much be cope with numerous variations of images of the same face due to changes in the following parameters such as 1. Pose, 2. Illumination, 3. Expression, 4. Motion, 5. Facial hair, 6. Glasses, 7. Background of image. Face recognition technology is well advance that can be applied for many commercial applications such as personal identification, security system, image- film processing, psychology, computer interaction, entertainment system, smart card, law enforcement, surveillance and so on. Face recognition can be done in both a still image and video sequence which has its origin in still-image face recognition. Different approaches of face recognition for still images can be categorized into three main groups such as 1. Holistic approach 2. Feature-based approach 3. Hybrid approach product.

One of the rare methods of bio metric identification that has the advantages of high precision and moderate intrusiveness is face recognition. It is simpler than a physiological method while maintaining the accuracy of one. The increasing number of real-world applications requiring the recognition of human faces have inspired numerous researchers to develop various face recognition systems during the past thirty years. A number of problems combine to make automatic facial recognition a highly complex commitment. On the other hand, the facial image that is entered into the database can frequently be obtained in a variety of ways. Automatic face identification is essential because it must be able to handle a large number of different kinds of the same face image due to variations in parameters such as 1. Present, 2. Light, 3. Communication, 4. Motion, 5. Hair on the face, 6. Eye wear 7. The image background. Advanced face recognition technology has many commercial uses, including electronic identification, monitoring, psychology, computer user interface image-film processing, security systems, smart cards, and personal recognition. Face recognition, which originated in still images, can now be performed in both still images and video segments. Three primary categories can be used to group numerous methods of facial recognition for still photos: 1. A holistic method 2. A feature-based approach 3. Product with hybrid approach. The

individual application's demands, the amount of computing power at hand, and the planned compromise between accuracy and efficiency all impact the choice of approach.

1. Holistic method: A face recognition system utilizing a holistic approach, also known as a global function, considers the entire face region as input data. The most popular face recognition method, itself, is an example of a holistic approach. Other holistic methods include fisher faces, the use of support vector machines, nearest feature lines (NFL), statistical itself, and independent-component analysis methods. They are all dependent on principal component analysis, or PCA, methods, which are useful for minimizing the size of a dataset while compromising its quality.

2. Characteristic In a feature-based approach, the eyes are segmented and used as input data for a framework classifier. Based approaches, or local features, involve features on the face, such as the nose. This group includes electromagnetically geometry, dynamic link design, and hidden Markov algorithm techniques. Based on DLA, one of the most effective of these systems is the elastic bunch graph match (EBGM) system [40],[41]. In these graph matching methods, wavelets—particularly Gabor wavelets—serve as the essential building blocks for facial representation. In a local feature representation, wavelet coefficients with values for many scales and rotation based on fixed wave bases are typically used. The wavelet coefficients that are calculated locally exhibit resilience against variations in clarity, translation, distortion, rotation, and scaling. To convey the pattern classes, the grid is positioned correctly over the image and saved along with the locally generated jet of each grid point. When a new image emerges, it is converted into a grid of jets and all previously saved model graphs are compared to the image. By creating and shifting linkages between vertices in the model domain, the DLA is conformed.

3. Hybrid approach: This strategy is based on how the human visual system processes both local and holistic features. The important elements that impact the hybrid approach's efficiency are how to decide which features to include and how to combine them in a way that preserves their benefits while avoiding their drawbacks as well. In the field of machine learning, these kinds of problems are closely related to ensembles learning and the multiple classifier system (MCS). Tragically,

these issues are still unsolved even in these fields. Even this, a lot of work in these areas has given us some insights into how to solve these problems, and these lessons can be applied when creating a hybrid face recognition system. A hybrid strategy that utilizes both local and holistic information for recognition could be a useful tool for streamlining classifier complexity while improving the accuracy of generalization. (2015) Shirodkar M, et al. [41] suggested method for monitoring attendance involves comparing photographic faces to the student face database and marking punctuality using the face focus technique. Edelweiss Practical Science and Technology, Sakshi et al., 2021 Sakshi, Singh P, Sharma C, Sharma S, and Khan AI are cited. Applied sciences and applications for advanced attendance administration systems (2021) In light of this, Edelweiss Applied Sci Tech.For the length of the admissions procedure, student photos are taken in unique movements and saved in the college database. The device already has cameras attached to take a student's front-only picture. If the image matches one in the database, the device will automatically label the present student as that particular one. Reports are accessible for downloading from the front end at any time by authorities. The creator's finishing efficiency with the suggested system was 83.2Due of variances in the students' poses, the author of this study installed an exceptionally high-definition camera above the blackboard and captured every student three times over the whole class hour. The writer regards the Viola-Jones algorithm for face detection in their

knowledge because of its rapid feature determination process The device will capture and process data in three frames, with the frame presenting the most precise detection charge that the gadget will take into account when tracking attendance in an Excel file. A model of an automatic attendance system was presented by the authors in [4]. The model focuses on how approved student are identified and registered as they enter and exit the classroom using face recognition and Radio Frequency Identification (RFID). Every registered student's real record is tracked by the system. In addition, the system saves every pupil's data in the attendance record for a particular class and gives out the necessary data based on demand. the authors of this study [5] developed and put into operation an attendance system that makes use of iris identification. The first thing that the visitors were compelled to do was register their information and retinal template. The technology

automatically recorded attendance in class by taking a picture of every pupil's eye, identifying their iris, and looking for an appropriate match in the database that was built. The web was used as a prototype. The authors of [6] developed an attendance system that used facial recognition technologies. The system was developed using algorithms that included Viola-Jones and Histogram of Oriented Gradients (HOG)\ characteristics together with a Support Vector Machine (SVM) classification. The authors drew into account a number of real-time conditions, involving expanding, clarity, obstacles, and position. Using the MATLAB GUI, quantitative analysis based on peak signal to Noise Ratio (PSNR) values has been carried out. By analyzing the Receiver Operating Characteristics (ROC) arc, the authors in [7] carried out research

to figure out which facial recognition algorithm—Eigenface and Fisher face—provided by the Open CV 2.4.8 were the best. They then incorporated the method into the system for tracking attendance. The ROC curve suggested that Eigenface outperformed Fisherface in the studies performed for this paper. The precision of the system that used the Eigenface algorithm was between 70 In [8], authors used Discrete Wavelet Transforms (DWT) and Discrete Cosine Transform (DCT) to demonstrate a face recognition the solution for an instructional student attendance system. The aforementioned methods were employed to extract characteristics from the students' faces, and then the Radial Basis Function (RBF) was employed to classify the objects on the features. The correctness rate achieved by this method was 82.

2.2 Proposed System

The Software Requirement Specification (SRS) is aimed at defining the necessary functionalities and Uniform Resource Locator (URL) for the Intelligent Network Backup Tool. It intends to establish a clear understanding of the final product's features and specifications as envisioned by both the development team and the client. The requirement statements are prioritized and detailed in this document. It targets project developers, managers, users, testers, and documentation writers, providing them with information on design and implementation constraints, external interface requirements, system features, nonfunctional requirements, and dependencies. Identifying needs is crucial for businesses and organizations to evaluate their market performance and maintain a competitive edge. The proposed system seeks to automate the existing manual attendance system by utilizing face recognition technology. Its main objective is to capture and store each student's face for attendance purposes. Accurate detection of all facial features during the image capture process is vital. With facial recognition steps applied to the captured image, teachers no longer have to take attendance manually during class. This paper tackles the challenges commonly associated with manual attendance systems. To detect faces, Haar Cascade classifiers are utilized, while the Local Binary Pattern Histogram (LBPH) algorithm is used to recognize student faces. The proposed system for Face Recognition based Classroom attendance system. The system requires a camera installed in the classroom at a position where it could capture all the students in the classroom and thus capture their images effectively. This image is processed to get the desired results.

2.3 Literature Review Summary **(Minimum 7 articles should refer)**

Year and Citation	**Article/ Author**	**Tools/ Software**	**Technique**	**Source**	**Evaluation Parameter**
2012	Naveed Khan Baloch	Face Recognition based Attendance Management System	Image acquisition • Histogram normalization • Noise removal • Skin classification • Face detection • Face recognition • Attendance	Journal of Emerging Technologies and Innovative Research	
2020	Smitha, Pavithra S Hegde, Afshin	Face Recognition based Attendance Management System	• Dataset Creation • Face Detection • Haar-Cascade Classifier with OpenCV • Detect • MultiScale module	International Journal of Engineering Research and Technology (IJERT)	
2018	Md. Sajid Akbar,Pronob Sarker,Ahmad Tamim Mansoor	Face Recognition and RFID Verified Attendance System	• Image Processing • OpenCV • Facial Recognition • RFID Tags • RFID Readers • Arduino Project • IR Module • Class Attendance • Smart Classroom	2018 International Conference on Computing, Electronics & Communications Engineering (ICCECE)	

			• Radio Waves		
2017	Kennedy Okokpujie, Etinosa Noma-Osaghae, Olatunji Okesola, Samuel Ndueso John	Implementation of a Student Attendance System Using Iris Biometric Recognition	• Biometrics • web application • attendance • Iris Recognition • Canny Edge detector algorithm,	2017 International Conference on Computational Science and Computational Intelligence (CSCI)	
2017	Hemantkumar Rathod, Snehal Sane, Suresh Raulo, Yudhisthir Ware	Automated attendance system using machine learning approach	• Viola-Jones and Histogram of Oriented Gradients (HOG) • Support Vector Machine (SVM) classifier • Peak Signal to Noise Ratio (PSNR) values • MATLAB GUI	ICNTE 2017, 10.1109/ICNTE. 2017.7947889	
2016	Lukas, Samuel	Student attendance system in classroom using face recognition technique	• Face recognition technique • Discrete Wavelet Transforms (DWT) • Discrete Cosine Transform (DCT) • Radial Basis Function (RBF)	2016 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2016	
2014	Siswanto, Adrian Rhesa Septian, Anto Satriyo	Implementation of face recognition algorithm for	• Facial recognition algorithm	2014 International Conference on ICT For Smart	

	Nugroho, and Maulahikmah Galinium	biometrics based time attendance system	(Eigenface and Fisherface) • Receiver Operating Characteristics (ROC) • Eigenface algorithm	Society (ICISS). IEEE, 2014	

Research on Based project[1.1]

3. PROBLEM FORMULATION

You're absolutely right. While the human brain effortlessly performs tasks like face recognition, replicating this ability in computers has been a significant challenge, although tremendous progress has been made in recent years.

Face recognition plays a crucial role in biometrics, which involves identifying individuals based on unique biological traits. In the case of face recognition, the distinct facial features of individuals are extracted and analysed using algorithms. These algorithms have evolved over time, becoming more efficient and accurate through continuous refinement and modification.

The applications of face recognition technology are diverse and far-reaching. In criminal identification, it can aid law enforcement agencies in identifying suspects from surveillance footage or images. Security systems can utilize face recognition for access control, allowing only authorized individuals to enter restricted areas. Additionally, face recognition can be used for identity verification in various contexts, such as unlocking smartphones or accessing secure online accounts.

Despite its potential benefits, the widespread adoption of face recognition technology has raised concerns about privacy, security, and potential misuse. Ethical considerations must be carefully weighed, and appropriate safeguards should be implemented to protect individuals' rights and ensure responsible use of this technology.

4. OBJECTIVES

The Attendance Management System you described seems to be a comprehensive solution for tracking student attendance in colleges. Let's break down some of its key features:

User Authentication: Each staff member is given a unique username and password, providing secure access to the system. This ensures accountability and allows staff members to access only the attendance data relevant to the subjects they are responsible for.

Attendance Tracking: Staff members are responsible for recording student attendance for the subjects they teach. The system likely provides an interface for quickly and accurately recording attendance, possibly using dropdown menus or checkboxes to mark attendance.

Attendance Reports: The system generates various types of reports to provide insights into attendance patterns. These reports may include student-by-student attendance, day-by-day attendance, class-by-class attendance, and month-by-month attendance. This comprehensive reporting allows staff members and administrators to monitor attendance trends and identify any issues that may need to be addressed.

User-Friendly Interface: The system prioritizes ease of use with a user-friendly interface. This makes it easy for staff members to navigate the system, record attendance, and generate reports. Additionally, the graphical representation of data enhances interpretation and analysis, making it easier for users to understand attendance trends at a glance.

Fast and Secure Data Storage: The system ensures fast and secure storage of attendance data. This is crucial for maintaining the integrity of the data and ensuring that it is readily accessible when needed.

5. METHODOLOGY

The purpose of the Software Requirement Definition (SRD)is to specify the universal resource location (URL) and effectiveness that the Automated Network Recovery Application must implement. It attempts at establishing an understanding of the features and requirements of the end product as imagined by the client and the creation team. The attached file contains a categorized complete list of the needed requirements. It provides details about design and implementation prohibitions, exterior connection requirements, system characteristics, malfunctioned requires, and dependencies to project builders executives, users, technicians, and document

writers. For businesses and groups to evaluate their market

performance and establish a competitive edge, needs awareness is fundamental. Using recognition of facial features technology, the suggested approach seeks to automate the current individuals timekeeping system. Storing and recording every pupil's face for attendance records is the main objective. It is important that every facial feature be correctly identified while taking an image. Using methods for recognizing faces on the captured image, teachers can take attendance in class without

requiring to actually do it. This essay covers the issues that are frequently associated with keeping written records. The Haar

Cascade algorithms are applied to identify faces, while the Local Binary Pattern Histogram (LBPH) methodology is used to identify faces of pupils. The Face Recognition-based Class attendance method that is being developed. For the system to work, a camera must be placed in the classroom in an orientation that allows it to record each pupil in. Both the detection tool and the simulator are provided by Open-CV. Using Open-CV, we can train a classification algorithm for any object, including automobiles, aircraft, and structures. For the cascade image classification algorithm,

there are two main states: recognition and learning. Open-CV offers two cascading classifier training applications such as OpenCV train cascade and OpenCV retraining. The classification algorithm is maintained in an independent file format in these two programs. We require a collection of instances for learning. Two categories of samples exist: • Negative sample: It has to do with images of non-objects. • Positive samples: This is a significant picture that has recognizable things. While

the collection of positive data is generated by the Open CV create samples tool; a set of negative data requires to be systematically provided. Negative Example Random pictures are used to generate negative examples. In a text file, samples with negative values are inserted. A picture filename for the

sample that is negative is included in each line of the file, corresponding to the location of the content of the file. This

file needs to be made by hand. Multiple dimensions can be seen in specified photos. Positive Example The OpenCV create samples tool produces samples that are positive. These instances can be generated from an earlier the collection or from a single image containing an object. It's essential to keep in mind that, as

the utility suggested above simply applies to the viewpoint modification, we need a sizable dataset of positive examples

before you submit it. Step 1: First, input pictures (or video) in Gray scale format must be installed, together with the required XML classification systems. Step 2: Once the image has been transformed to Gray scale, we can modify it by cropping, resizing, sharpening, and blurring it as required. The next stage is segmentation of the image, which involves locating many objects inside a single picture so that the machine learning algorithm is able to quickly identify faces and objects in the picture.

proposed system for Face Recognition based Classroom attendance system.

Generally, there are three steps to this process:

1**) Dataset Creation:** A web-based footage serves to take pictures of the pupils. Each pupil will be captured in

multiple pictures from various perspectives and movements. These photos are processed beforehand. To capture the Region of Interests (ROI), which will be eventually used in the process of identification, the photos are reduced. The trimmed pictures must then be resized to a specific pixel location. After that, these RGB pictures

will be converted to grayscale version. Once that, a folder holding the names of every pupil will have these

pictures stored in it.

2) Face Detection: Face detection here is performed using Haar Cascade Classifier with OpenCV. Haar Cascade algorithm needs to be trained to detect human faces

before it can be used for face detection. This is called feature extraction. The haar cascade training data used

is an xml file haar cascade frontal face default. The haar features shown in Fig.2. will be used for feature extraction Here, OpenCV's Haar Cascade Classification is used to identify faces. While the Haar Cascade technique can be implemented for face identification, it must be taught to identify human faces. We refer to this as feature extraction. An xml file named haar cascade frontal face default contains the haar cascade training data. For feature extraction, the haar features displayed in will be utilized.

3) Corrections on Attendance: Throughout the face recognition process, the faces that were identified will be noted as present in the spreadsheet, while the remaining faces will be recorded as absent. A list of the absences will then be forwarded to the relevant departments. At the end of every month, the corresponding attendance

sheet will be modified for the departments. Faces will be recognized from the live broadcast classroom footage urging each session. The photos in the dataset will be

contrasted with the faces that were recognized. Should a match be discovered, the appropriate student's attendance will be registered. A list of absentees will be sent

to the appropriate instructor monitoring the session at the conclusion of all of them.

The proposed system face recognition-based attendance system can be divided into five main modules. The modules and their functions are defined as follows.

a. **Image Capture**: The high-resolution camera which is used for capturing video is used to take frontal images of the students.

b. **Pre-processing**: The images are converted from RGB to Grayscale and are scaled down by a factor of 1.2.

c. **Face Detection**: A proper and efficient face detection algorithm always increases the performance of face recognition systems. Various algorithms are proposed for face detection such as face knowledge-based methods, feature invariant methods, machine learning based methods. In this project, I implemented a system for locating faces in digital images. These are in JPEG format only. Before we continue, we must differentiate between face recognition and face detection. They are not the same, but one depends on the other. In this case face recognition needs face detection for making an identification to "recognize" a face. I will only cover face detection. Face detection uses classifiers, which are algorithms that detects what is either a face (1) or not a face (0) in an image.

d. Developing a dataset The faces detected in images are stored in the database after pre-processing and detection. A minimum of 20 images are captured per individual student along with a unique ID.

The dimensions of these stored images are 212×212 pixels. These images are later used to train the recognizer.

e. **Face Recognition**: Local Binary Pattern (LBP) is a smooth & adequate operator, which operates by setting the pixels of an image by thresholding the neighbourhood of each pixel and examines the outcome as a binary number. Histogram of Oriented Gradients (HOG) descriptor increases the detection performance when combined with LBP. Therefore, a combination of LBP & HOG which gives LBPH algorithm is used for face recognition.

Python

Python is a high-level, general-purpose programming language known for its simplicity, versatility, and readability. Here's an expansion on its key features:

- **High-Level Language**: Python is a high-level programming language, meaning it provides abstracted functionalities that allow developers to focus on solving problems rather than worrying about low-level details.
- **General-Purpose:** Python is suitable for a wide range of applications, including web development, data analysis, scientific computing, artificial intelligence, automation, and more. Its versatility makes it a popular choice for both beginner and experienced programmers.
- **Code Readability:** Python's design philosophy emphasizes code readability, making it easier for programmers to write clear and understandable code. One of the defining features of Python is the use of significant indentation (whitespace), which replaces traditional braces or keywords to denote code blocks.
- **Dynamic Typing:** Python is dynamically typed, meaning variables do not have predefined types. The type of a variable is determined at runtime based on the value assigned to it. This allows for flexible and concise code but may require careful attention to type compatibility.
- **Garbage Collection**: Python utilizes automatic garbage collection to manage memory allocation and deallocation. This helps to simplify memory management for developers by automatically reclaiming memory occupied by objects that are no longer in use

Pandas

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively.

It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

dlib

The dlib library provides two functions that can be used for face detection:

1. HOG + Linear SVM: dlib.get_frontal_face_detector()

2. MMOD CNN: dlib.cnn_face_detection_model_v1(modelPath)

get_frontal_face_detector function does not accept any parameters. A call to it returns the pre-trained HOG + Linear SVM face detector included in the dlib library.

The Dlib's HOG + Linear SVM face detector is fast and efficient. By nature of how the Histogram of Oriented Gradients (HOG) descriptor works, it is not invariant to changes in rotation and viewing angle.

For more robust face detection, you can use the MMOD CNN face detector, available via the cnn_face_detection_model_v1 function. This method accepts a single parameter, modelPath, which is the path to the pre-trained mmod_human_face_detector.dat file residing on disk.

pip install click

Click is a Python package for creating beautiful command line interfaces in a composable way with as little code as necessary. It's the "Command Line Interface Creation Kit". It's highly configurable but comes with sensible defaults out of the box. It aims to make the process of writing command line tools quick and fun while also preventing any frustration caused by the inability to implement an intended CLI API.

Click in three points:

• arbitrary nesting of commands

• automatic help page generation

• supports lazy loading of subcommands at runtime

pip install cmake

CMake is a cross-platform free and open-source software tool for managing the build process of software using a compiler-independent method. It supports directory hierarchies and applications that depend on multiple libraries.

pip install Pillow

Pillow is a Python Imaging Library (PIL), which adds support for opening, manipulating, and saving images. The current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.

pip install datetime

A date in Python is not a data type of its own, but we can import a module named ‘datetime’ to work with dates as date objects.

```
import datetime

x = datetime.datetime.now()

print(x.year)

print(x.strftime("%A"))
```

To create a date, we can use the datetime() class (constructor) of the datetime module. The datetime() class requires three parameters to create a date: year, month, day.

pip install pytest-shutil

This library is a goodie-bag of Unix shell and environment management tools for automated tests. A summary of the available functions is below, look at the source for the full listing.

pip install python-csv

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. CSV format was used for many years prior to attempts to describe the format in a standardized way in **RFC 4180**. The lack of a well-defined standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary.

possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer. The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

pip install NumPy

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

Face recognition

The face recognition library, created by Adam Geitgey, wraps around dlib's facial recognition functionality, and this library is super easy to work with and we will be using this in our code. Remember to install dlib library first before you install face recognition.

There are various types of face recognition algorithms, for example:

- Eigenfaces (1991)
- Local Binary Patterns Histograms (LBPH) (1996)
- Fisherfaces (1997)
- Scale Invariant Feature Transform (SIFT) (1999)
- Speed Up Robust Features (SURF) (2006)

1. **Face Detection:**

   Face detection is the initial step in the face recognition process. In this step, the system analyses an image or video stream to identify regions where faces are present. Once a face is detected, the system can then extract the facial region for further analysis and processing.

   Various algorithms and techniques are employed for face detection, with some of the most popular ones being Viola-Jones algorithm, Histogram of Oriented Gradients (HOG), and Convolutional Neural Networks (CNNs). These algorithms analyze the features of the image or video frames to identify patterns indicative of a human face, such as the arrangement of eyes, nose, and mouth.

   Once the face is detected, its exact coordinates or location within the image are determined. This information is crucial for accurately extracting the facial region from the image or video frame. By isolating the face, the system can focus its processing resources specifically on the facial features, which is essential for subsequent steps in the face recognition pipeline.

2. **Feature Extraction:**

   Now see we have cropped out the face from the image, so we extract specific features from it. Here we are going to see how to use face embeddings to extract these features of the face. As we know a neural network takes an image of the face of the person as input and outputs a vector that represents the most important features of a face! In machine learning, this vector is nothing but called embedding and hence we call this vector face embedding. Now how this will help in recognizing the faces of different people?

   When we train the neural network, the network learns to output similar vectors for faces that look similar. Let us consider an example, if I have multiple images of faces within different timelapse, it's obvious that some features may change but not too much. So, in this problem, the vectors associated with the faces are similar or we can say they are very close in the vector space. Up to this point, we came to know how this network works, let us see how to use this network on our own data. Here we pass all the images in our data to this pre-trained network to get the respective embeddings and save these embeddings in a file for the next step.

3. **Comparing Faces:**

   Once we have face embeddings for each face in our dataset saved in a file, the next step in recognizing a new image is to compute the face embedding for the new image using the same network architecture or model we used earlier for generating embeddings. This network is typically a deep learning model, such as a Convolutional Neural Network (CNN), trained specifically for face recognition tasks.

   To compute the face embedding for the new image, we pass the image through the pre-trained network. The network processes the image and outputs a numerical representation of the facial features, known as the face embedding. This embedding captures the essential characteristics of the face in a high-dimensional space.

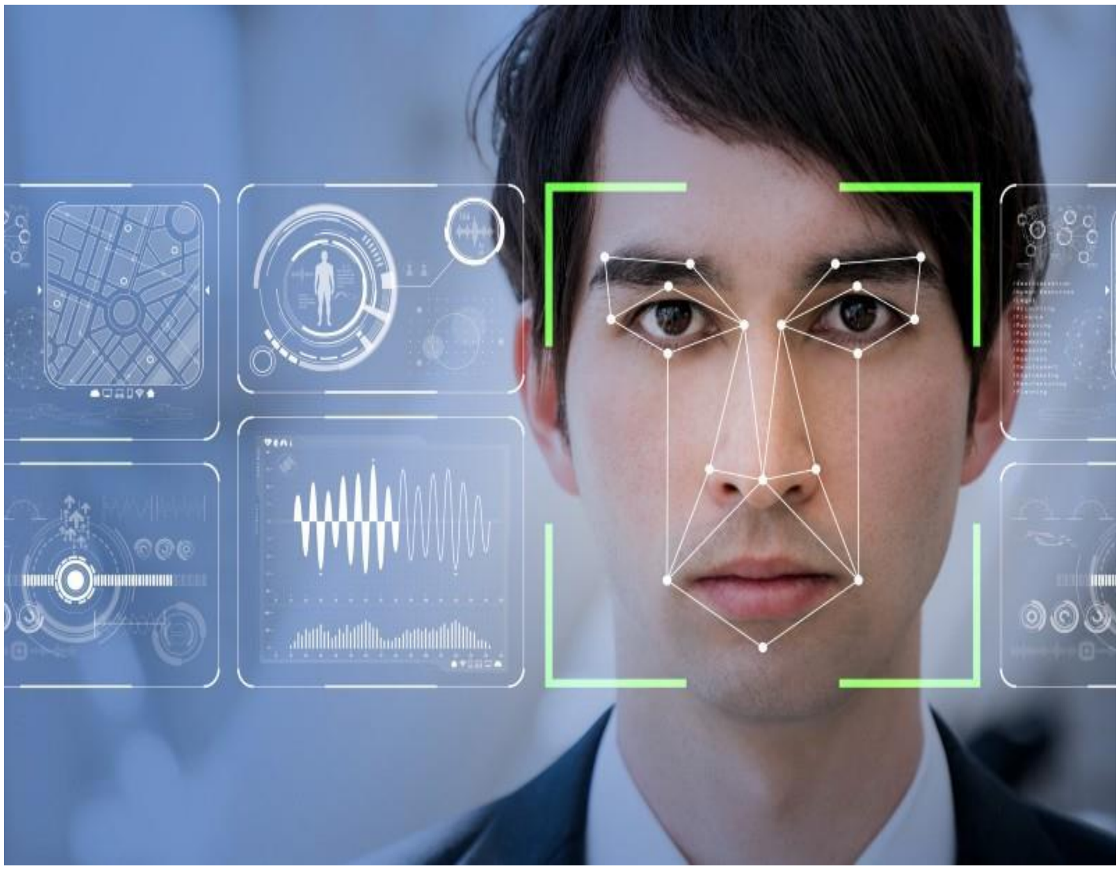
   Once we have the face embedding for the new image, the next step is to compare this embedding with the embeddings of faces in our dataset. This comparison is typically done using a distance metric, such as Euclidean distance or cosine similarity. The goal is to determine how similar or close the embedding of the new image is to the embeddings of faces in our dataset.

If the generated embedding is sufficiently close or similar to any other embedding in our dataset, we can recognize the face as belonging to the individual represented by that embedding. In other words, if the distance between the embeddings falls below a certain threshold or if the similarity score exceeds a predefined threshold, we can conclude that the new image depicts a known face from our dataset.

This process allows us to recognize faces in new images based on their similarity to faces in our dataset. It's worth noting that the accuracy of face recognition depends on various factors, including the quality of the embeddings, the effectiveness of the network architecture, and the size and diversity of the training dataset. Additionally, fine-tuning parameters such as the distance threshold for recognition can impact the performance of the system.

The face embedding for the new image, we pass the image through the pre-trained network. The network processes the image and outputs a numerical representation of the facial features, known as the face embedding. This embedding captures the essential characteristics of the face in a high-dimensional space.

As we know a neural network takes an image of the face of the person as input and outputs a vector that represents the most important features of a face! In machine learning, this vector is nothing but called embedding and hence we call this vector face embedding.



The network processes the image[1.1]

OpenCV is Free and Open-Source:

OpenCV is freely available for download and use, making it accessible to a wide range of developers and organizations without any licensing fees. Being open-source, it allows users to inspect, modify, and contribute to the codebase, fostering collaboration and innovation within the computer vision community.

High Performance and Speed:

OpenCV is written in C/C++, which are highly efficient and fast programming languages. This enables OpenCV to deliver high performance and speed, making it suitable for real-time applications such as video processing, object detection, and image recognition. The optimized codebase of OpenCV ensures efficient utilization of system resources, resulting in faster execution times compared to other libraries.

Low System Resource Usage:

OpenCV is designed to be resource-efficient, making it suitable for deployment on devices with limited hardware capabilities. Even with less system RAM, OpenCV can perform complex computer vision tasks efficiently, making it ideal for embedded systems, mobile devices, and IoT (Internet of Things) applications.

Cross-Platform Compatibility:

OpenCV is compatible with most operating systems, including Windows, Linux, and macOS. This cross-platform compatibility allows developers to write code once and deploy it across different environments without significant modifications. Whether developing applications for desktops, servers, or embedded devices, OpenCV provides a consistent and reliable framework for computer vision tasks.

The combination of being free, open-source, high-performance, low-resource usage, and cross-platform compatibility makes OpenCV a popular choice for developers working on machine learning, deep learning, and computer vision projects. Its versatility and robustness have contributed to its widespread adoption in various industries, including robotics, healthcare, automotive, surveillance, and more.

HAAR-Cascade Detection in OpenCV

Haar-cascade detection in OpenCV is a technique used for object detection based on Haar-like features. Here's a detailed explanation of the process:

Training and Detection:

The process of Haar-cascade detection involves two primary states: training and detection. During training, a cascade classifier is trained on a set of positive and negative samples to recognize a specific object or pattern. Once trained, the cascade classifier can be used for detection to identify instances of the object within images or video streams.

Training Tools:

OpenCV provides two applications for training cascade classifiers:

opencv_haartraining: This is one of the legacy applications for training cascade classifiers. It uses a collection of positive and negative samples to generate a classifier XML file.

opencv_traincascade: This is the more modern and recommended tool for training cascade classifiers in OpenCV. It offers more features and flexibility in the training process and produces a classifier XML file compatible with OpenCV.

Training Process:

To train a cascade classifier, a set of positive and negative samples is required:

Positive samples: Images containing the object of interest (e.g., cars, faces).

Negative samples: Images that do not contain the object of interest.

The training process involves the following steps:

Collect a large dataset of positive and negative samples.

Generate positive samples by cropping and resizing images containing the object of interest.

Create a negative sample file containing the paths to images without the object of interest.

Configure parameters such as the number of stages, minimum hit rate, and maximum false alarm rate.

Run the training process using the opencv_traincascade tool, specifying the positive and negative sample files, as well as other parameters.

Monitor the training process and evaluate the performance of the cascade classifier using metrics such as detection rate and false positive rate.

Once training is complete, the cascade classifier XML file is generated, containing the learned features and parameters for object detection.

Detection Process:

Once the cascade classifier is trained and the XML file is generated, it can be used for object detection. The detection process involves:

Loading the trained cascade classifier XML file using OpenCV.

Reading an input image or video stream.

Applying the cascade classifier to detect instances of the object within the image or video.

Drawing bounding boxes around detected objects and displaying the results.

Usage:

Haar-cascade detection in OpenCV is commonly used for various applications such as face detection, pedestrian detection, vehicle detection, and more. It provides a robust and efficient method for object detection in real-time scenarios.

Overall, Haar-cascade detection in OpenCV offers a powerful tool for training and detecting objects in images and video streams, making it a valuable technique for computer vision applications.

There are two types of samples:

- Negative sample: It is related to non-object images.
- Positive samples: It is a related image with detect objects.

A set of negative samples must be prepared manually, whereas the collection of positive samples is created using the opencv_createsamples utility.

Creating training data for object detection or recognition tasks typically involves preparing both negative and positive samples:

Negative Samples:

Negative samples are images that do not contain the object of interest. These images are typically selected from a diverse set of arbitrary images that represent backgrounds where the object is not present.

The negative samples are compiled into a text file, where each line contains the filename (or path) of a negative sample image. These filenames are usually relative to the directory where the description file is located.

It's important to ensure that the negative samples cover a wide range of backgrounds and variations to ensure robustness in the trained model's performance.

Positive Samples:

Positive samples are images that contain the object of interest. These images are used to train the model to recognize the object's features and characteristics.

The process of creating positive samples often involves the use of OpenCV's opencv_createsamples utility. This utility generates positive samples by applying various transformations (e.g., scaling, rotation, perspective transformations) to a single source image containing the object of interest.

The opencv_createsamples utility requires parameters such as the input image containing the object, the number of samples to generate, and the output file format (e.g., XML or YAML).

The generated positive samples, along with the manually prepared negative samples, are used as input for the training process to create a cascade classifier or other object detection/recognition models.

The combination of negative and positive samples is essential for training accurate and reliable object detection or recognition models. The negative samples provide background information, while the positive samples teach the model to identify the object's features and distinguish it from the background. This process helps in building robust and effective models for various computer vision tasks.

Here's a breakdown of the steps involved:

Step involved in creating positive samples using OpenCV's opencv_createsamples utility:

Positive Samples:

Positive samples are images that contain the object of interest, such as faces, cars, or any other target object for detection or recognition. These samples serve as the basis for training the model to recognize the object.

Creating Positive Samples:

OpenCV provides the opencv_createsamples utility, which automates the process of generating positive samples from a single image or a collection of images containing the target object.

The utility applies perspective transformations to these images to augment the dataset. Augmentation helps increase the diversity of the training data and improve the model's robustness to variations in scale, rotation, and viewpoint.

The opencv_createsamples utility requires parameters such as the input image containing the object, the number of samples to generate, and the output file format (e.g., XML or YAML).

Large Dataset Requirement:

It's essential to have a large dataset of positive samples before using the opencv_createsamples utility. A diverse and representative dataset enables the model to learn the variations and characteristics of the object effectively.

The utility applies perspective transformations to these samples to simulate different viewpoints and orientations, enhancing the model's generalization ability.

The diversity of the dataset ensures that the model can accurately detect or recognize the object under various conditions, such as different lighting conditions, backgrounds, and orientations.

Perspective Transformation:

Perspective transformation involves changing the viewpoint of an object in an image. This transformation is crucial for generating variations of the object's appearance, which helps the model learn to detect or recognize the object under different conditions.

Perspective transformations may include:

- **Scaling:** Resizing the object to simulate variations in size.
- **Rotation**: Changing the object's orientation to simulate different viewpoints.
- **Translation**: Shifting the object's position within the image.
- **Shearing:** Skewing the object's shape to simulate perspective distortions.

these steps and generating a diverse set of positive samples, you can create training data that effectively teaches the model to detect or recognize the target object in various scenarios. It's essential to ensure the quality and diversity of the dataset to achieve optimal performance from the trained model. Regularly evaluating and refining the dataset based on the model's performance is also recommended to improve detection accuracy and robustness

Step - 1

Load Necessary XML Classifiers: OpenCV provides pre-trained Haar cascades or other XML classifiers for various objects, including faces. These classifiers define the features of the object you want to detect.

We can load the appropriate XML classifier file(s) using OpenCV's.

Load Input Images or Video: Next, you need to load the input images or video frames. It's often useful to convert the images to grayscale before processing, as this can improve detection performance and reduce computational load.

Alternatively, if you're working with a video stream, you can capture frames from the video using OpenCV's cv2.VideoCapture() function.

After loading the necessary XML classifiers and input images (or video frames), you're ready to perform face detection on the grayscale images using the loaded classifiers.

Step -2

After converting the image to grayscale and performing any necessary manipulation such as resizing, cropping, blurring, or sharpening, the next step typically involves object segmentation, where we identify multiple objects within the image. This segmentation step is crucial for improving the efficiency and accuracy of object detection, including the detection of faces.

There are several approaches to object segmentation, depending on the specific requirements of your application and the characteristics of the images being processed. Here are a few common techniques:

Thresholding: Thresholding is a simple technique where pixels in the image are classified as foreground or background based on their intensity values. This can be useful for segmenting objects with distinct intensity differences from the background.

Edge Detection: Edge detection algorithms such as Canny edge detection can be used to identify boundaries between objects in the image. These edges can then be used as a basis for segmentation.

Contour Detection: Contour detection algorithms identify the contours or outlines of objects in the image. These contours can be used to segment and extract individual objects.

Clustering: Clustering algorithms such as k-means clustering can be used to group similar pixels together, potentially separating different objects in the image.

Semantic Segmentation: Semantic segmentation algorithms assign a class label to each pixel in the image, allowing for fine-grained segmentation of different objects or regions.

Step - 3

The Haar-like feature algorithm is indeed a common technique used in face detection. It relies on defining and detecting specific features that are characteristic of human faces.

Haar-like features are simple rectangular patterns that are used to detect edges, lines, and textures in an image. These features are derived from the observation that certain regions of the face, such as the eyes, nose, and mouth, exhibit consistent patterns of intensity variations.

For example, as you mentioned, the eye region tends to be darker than its surroundings due to the presence of the eyeballs and eye sockets, while the nose region tends to be brighter due to the reflection of light off the nose surface. Haar-like features capture these intensity variations by comparing the sum of pixel intensities within rectangular regions of the image.

The Viola-Jones algorithm, which is a popular face detection algorithm based on Haar-like features, uses a cascade of classifiers trained to detect these features in images. Each classifier in the cascade evaluates a specific Haar-like feature at multiple scales and positions within the image. If the feature is present at a certain location and scale, it contributes to a score indicating the likelihood of a face being present at that location.

By combining multiple classifiers and thresholding the scores, the Viola-Jones algorithm can effectively detect faces in images with high accuracy and efficiency.

Step -4:

After performing any necessary image manipulation and segmentation, the next step is to extract features from the image that will help in identifying the location of faces. Common techniques for feature extraction in face detection include edge detection, line detection, and center detection.

Edge Detection: Edge detection algorithms such as Canny edge detection can help identify sharp changes in intensity within the image, which often correspond to edges of objects, including facial features.

Line Detection: Line detection algorithms such as Hough Transform can be used to detect straight lines within the image. In the context of face detection, lines representing the edges of facial features such as the eyes, nose, and mouth can be detected.

Centre Detection: Centre detection techniques involve identifying key points or landmarks within the image, such as the centre of the face or the eyes. These key points can serve as reference points for locating and aligning the face within the image.

Once features have been extracted from the image, the next step is to use these features to determine the location of faces. This typically involves defining a bounding box around each detected face, represented by the coordinates (x, y, w, h), where (x, y) denotes the top-left corner of the bounding box, and (w, h) denotes the width and height of the box.

OpenCV provides functions for drawing rectangles around detected objects, making it easy to visualize the location of faces within the image.

```
# drawing a rectangle around a detected face

x, y, w, h = face coordinates

  # Extracted coordinates of the detected face

# Draw rectangle around the detected face
```

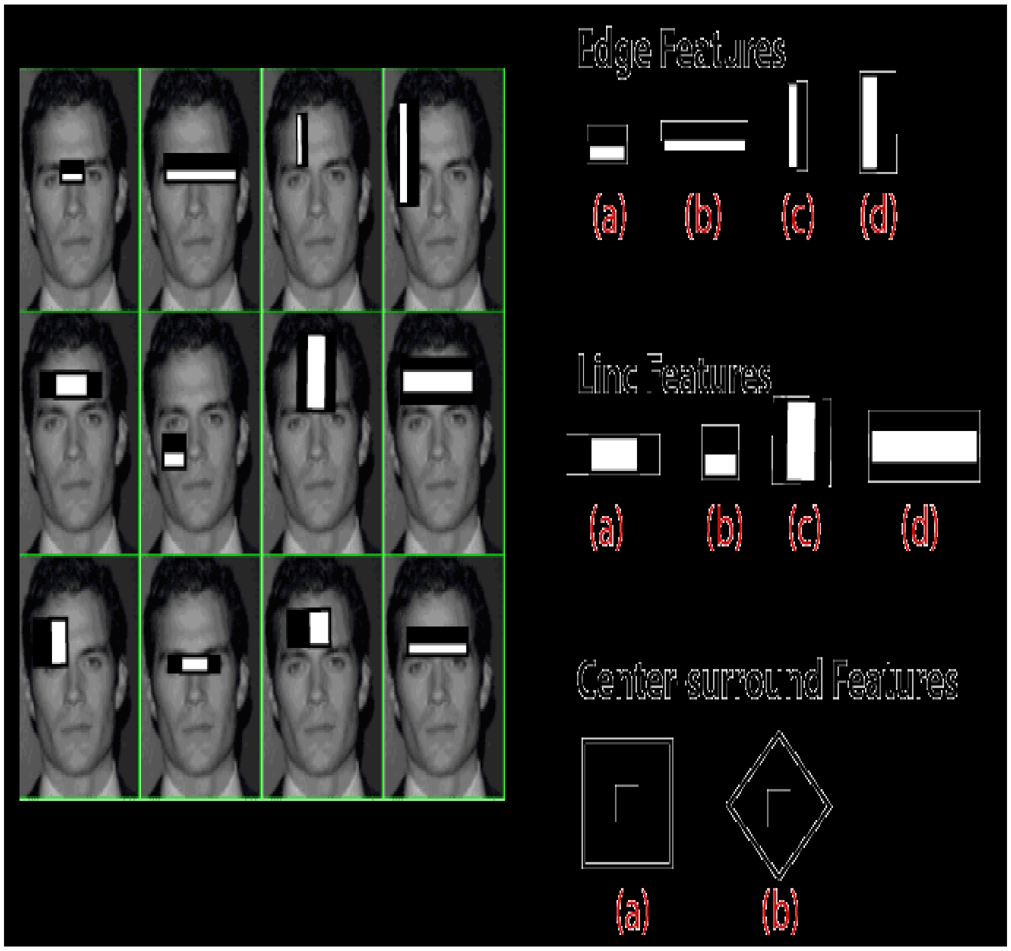
```
cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display the image with the rectangle

cv2.imshow('Detected Faces', image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```



Centre detection techniques involve identifying key points[2.1]

face_coordinates represent the coordinates of the detected face, and a green rectangle is drawn around the face on the original image. This visualization helps to clearly indicate the location of the detected face within the image.

The combination of being free, open-source, high-performance, low-resource usage, and cross-platform compatibility makes OpenCV a popular choice for developers working on machine learning, deep learning, and computer vision projects. Its versatility and robustness have contributed to its widespread adoption in various industries, including robotics, healthcare, automotive, surveillance, and more.

Centre detection techniques are pivotal in identifying significant landmarks or key points within an image, such as the center of the face or the position of the eyes. These landmarks act as crucial reference points for accurately locating and aligning the faces present in the image.

Once the features have been successfully extracted from the image, the subsequent step involves leveraging these extracted features to determine the precise location of faces. This process typically entails defining a bounding box around each detected face. These bounding boxes are represented by a set of coordinates (x, y, w, h), where (x, y) corresponds to the top-left corner of the bounding box, while (w, h) signifies the width and height of the box.

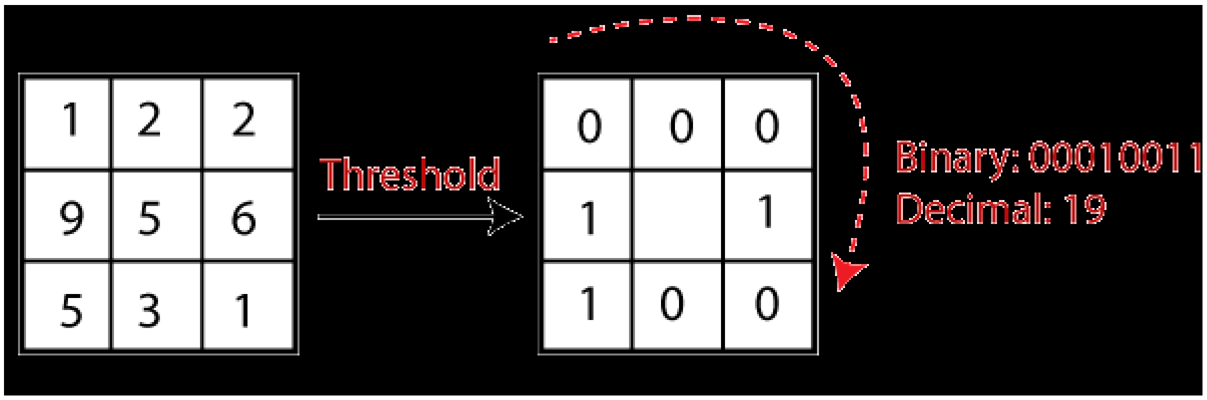
OpenCV offers a variety of functions specifically designed for drawing rectangles around detected objects, simplifying the task of visualizing the exact location of faces within the image. By utilizing these functions, developers can effectively annotate the image with bounding boxes, providing a clear and concise representation of the detected faces. This visualization aids in further analysis and processing of the detected faces, facilitating subsequent tasks such as facial recognition or attribute extraction.

Introduction of LBPH

Local Binary Pattern Histogram algorithm is a simple approach that labels the pixels of the image thresholding the neighbourhood of each pixel. In other words, LBPH summarizes the local structure in an image by comparing each pixel with its neighbours and the result is converted into a binary number. It was first defined in 1994 (LBP) and since that time it has been found to be a powerful algorithm for texture classification.

This algorithm is generally focused on extracting local features from images. The basic idea is not to look at the whole image as a high-dimension vector; it only focuses on the local features of an object.

In the below image, take a pixel as centre and threshold its neighbour against. If the intensity of the centre pixel is greater-equal to its neighbour, then denote it with 1 and if not then denote it with 0.



a pixel as centre and threshold its neighbour against[3.1]

Let's understand the steps of the algorithm:

1. Selecting the Parameters:

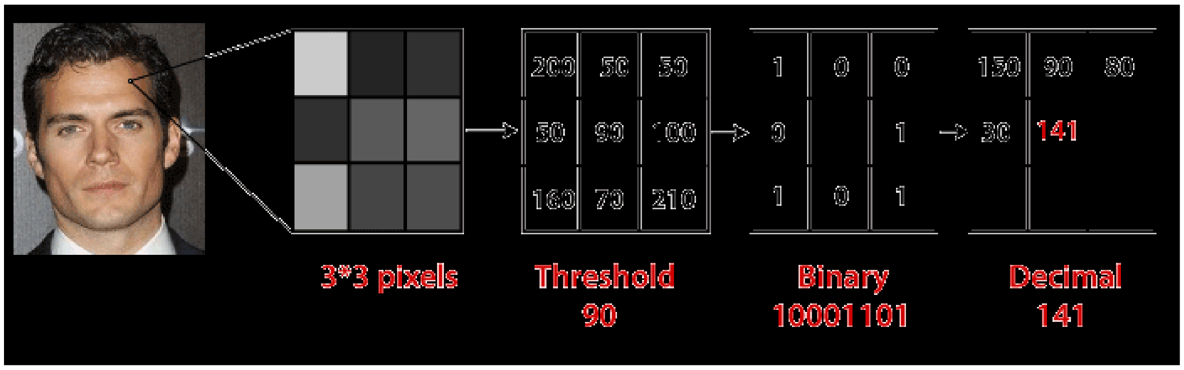
The LBPH accepts the four parameters:

- Radius: It represents the radius around the central pixel. It is usually set to 1. It is used to build the circular local binary pattern.
- Neighbour: The number of sample points to build the circular binary pattern.

- Grid X: The number of cells in the horizontal direction. The more cells and finer grid represent, the higher dimensionality of the resulting feature vector.
- Grid Y: The number of cells in the vertical direction. The more cells and finer grid represent, the higher dimensionality of the resulting feature vector.

2. **Training the Algorithm:** The first step is to train the algorithm. It requires a dataset with the facial images of the person that we want to recognize. A unique ID (it may be a number or name of the person) should provide with each image. Then the algorithm uses this information to recognize an input image and give you the output. An Image of particular person must have the same ID. Let's understand the LBPH computational in the next step.

3. Using the LBP operation: In this step, LBP computation is used to create an intermediate image that describes the original image in a specific way through highlighting the facial characteristic. The parameters radius and neighbour are used in the concept of sliding window.

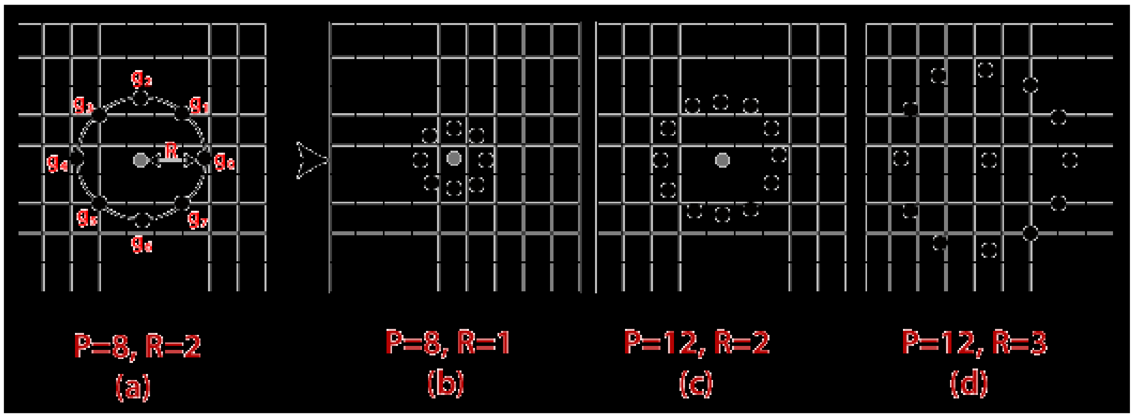


creating an intermediate image[4.1]

To understand in a more specific way, let's break it into several small steps:

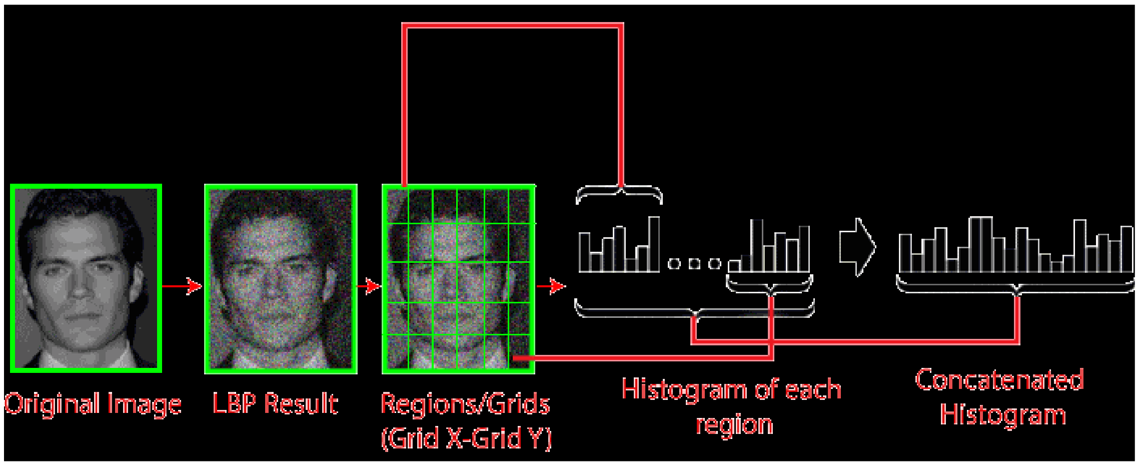
- Suppose the input facial image is grayscale.
- We can get part of this image as a window of 3x3 pixels.
- We can use the 3x3 matrix containing the intensity of each pixel (0-255).
- Then, we need to take the central value of the matrix to be used as a threshold.
- This value will be used to define the new values from the 8 neighbours.

- For every neighbour of the central value (threshold), we set a new binary value. The value 1 is set for equal or higher than the threshold and 0 for values lower than the threshold.
- Now the matrix will consist of only binary values (skip the central value). We need to take care of each binary value from each position from the matrix line by line into new binary values (10001101). There are other approaches to concatenate the binary values (clockwise direction), but the final result will be the same.
- We convert this binary value to decimal value and set it to the central value of the matrix, which is a pixel from the original image.
- After completing the LBP procedure, we get the new image, which represents better characteristics of the original image.



convert binary value to decimal value [5.1]

4. **Extracting the Histograms from the image:** The image is generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, let's consider the following image:



Extracting the Histograms from the image[6.1]

- We have an image in grayscale; each histogram (from each grid) will contain only 256 positions representing the occurrence of each pixel intensity.
- It is required to create a new bigger histogram by concatenating each histogram.

5. **Performing face recognition**: Now, the algorithm is well trained. The extracted histogram is used to represent each image from the training dataset. For the new image, we perform steps again and create a new histogram. To find the image that matches the given image, we just need to match two histograms and return the image with the closest histogram.

- There are various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. We can use the Euclidean distance based on the following formula:
- The algorithm will return ID as an output from the image with the closest histogram. The algorithm should also return the calculated distance that can be called confidence measurement.

$$D = \sqrt{\sum_{i=1}^{n}(\text{hist}\,1_i - \text{hist}2_i)^2}$$

- If the confidence is lower than the threshold value, that means the algorithm has successfully recognized the face.

Initialization:

Open Camera: Open the camera to capture live video frames using cv2.VideoCapture(0).

Import Face Classifier: Import the face classifier, which is often a LBPH (Local Binary Patterns Histograms) Face Recognizer, created using cv2.face. LBPHFaceRecognizer_create().

Read Trained Data: Read the trained data from the file trainer.yml using recognizer.read('trainer.yml'). This file contains the trained model with labeled face data.

GPU Part:

(a) Capture Image: Capture the current frame from the camera feed using img = cap.read().

(b) Convert to Grayscale: Convert the captured frame to grayscale, as face detection and recognition often perform better on grayscale images, using gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY).

(c) Detect Faces: Detect faces in the grayscale frame using a pre-trained face cascade classifier, such as face_cas.detectMultiScale(gray).

Face Recognition Loop:

For Each Detected Face:

Read the coordinates of the detected face.

Extract the region of interest (ROI) from the grayscale frame corresponding to the detected face using roi_gray = gray[y:y + h, x:x + w].

Compare the ROI with stored face data to recognize the face using confidence = recognizer.predict(roi_gray).

If Confidence < 85:

If the confidence level is below a certain threshold (e.g., 85), the face is recognized as one of the stored faces.

Attendance is marked for the recognized face using dataEntry(name, class_count).

Else:

If the confidence level is above the threshold, the face is not recognized.

The face is labelled as 'Unknown'.

Processes Involved:

Face Detection: Detect faces in the frame using a pre-trained face cascade classifier.

Face Recognition: Compare the detected faces with stored face data to recognize known faces.

Attendance Marking: Mark attendance for recognized faces.

Unknown Face Handling: Handle cases where the detected face does not match any stored face data.

This algorithm combines face detection and recognition to mark attendance for recognized faces in live video frames captured from the camera feed. It involves preprocessing the images, detecting faces, comparing them with stored face data, and taking appropriate actions based on the recognition results.

1. Pre-Processing Images:

Pre-processing images is a crucial step in preparing data for face recognition algorithms like Local Binary Patterns Histograms (LBPH). Here's how the pre-processing might be done:

Image Capture:

The system captures around 50 images of each individual's face. This ensures a diverse set of images that cover different poses, expressions, and lighting conditions, improving the robustness of the face recognition system.

These images are typically captured using a camera or webcam.

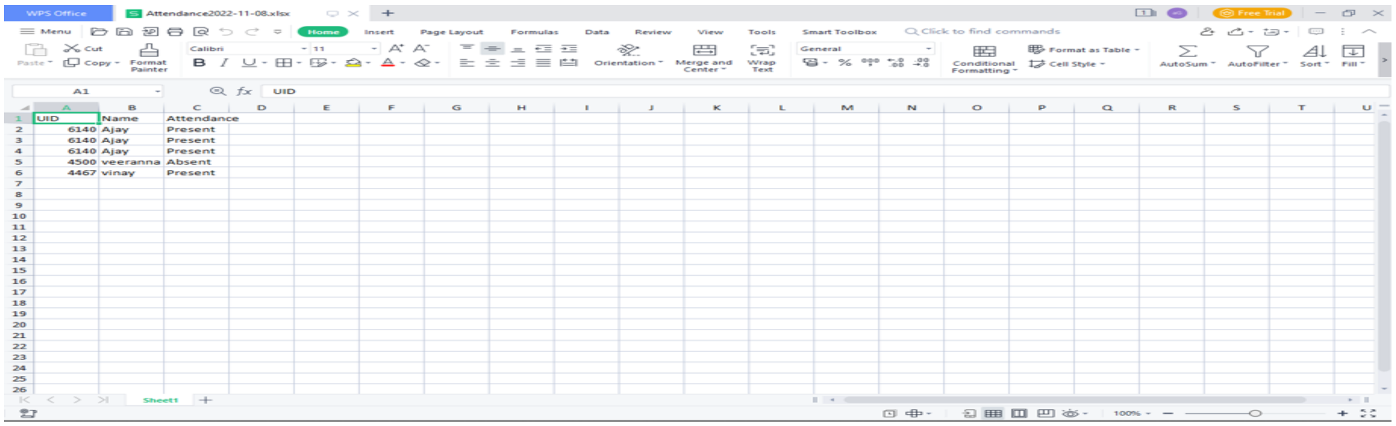
Conversion to Grayscale:

Before storing the images, they are converted to grayscale. Grayscale images contain only intensity values (typically ranging from 0 to 255) representing brightness levels, rather than color information.

Converting images to grayscale simplifies the processing and reduces computational requirements for face recognition algorithms like LBPH.

Storage in a Folder:

The pre-processed grayscale images are stored in a folder on the system's storage. Each person's images are typically stored in a separate subfolder, with the folder name or ID serving as a unique identifier for that person.



UID	Name	Attendance
6140	Ajay	Present
6140	Ajay	Present
6140	Ajay	Present
4500	veeranna	Absent
4467	vinay	Present

Data Storage file [7.1]

The folder structure might look like this:

```
dataset/


├── person1/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
├── person2/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
└── ...
```

Naming Convention:

Each image is saved with a name and ID unique to the person it represents. This ensures that the images can be associated with the correct individual during training and recognition.

The naming convention might include the person's name or a unique identifier followed by a sequential number to distinguish between multiple images of the same person.

Quality Control:

It's important to ensure that the captured images are of sufficient quality for effective face recognition. Factors such as resolution, lighting, and focus should be considered to minimize variability and improve the accuracy of recognition.

2. Face Detection When a person appears in front of the camera, the camera detects that a face is present and a frame appears around the face. The entire frame is converted to greyscale as LBPH works only on greyscale images.

1) Eigen face:

- This algorithm extracts the necessary information from an image and efficiently encodes it.
- To obtain variations, a number of pictures of a single person is taken.
- For the set of images of faces, eigenvectors and its covariance matric is calculated and stored.
- Since every image represents an eigen vector, the data set helps produce variety for the system.
- A representation of these eigen vectors is called eigen faces.

2) Line Edge Map:

- One of the popular methods is using the Line Edge Maps algorithm.
- In this method line matching is done to map the features of the face.
- This algorithm mainly uses the most prominent features of the face; mainly the eyes, nose and mouth having high characteristics.
- The colour images are converted to greyscale to observe and extract the similarities in the faces.

- Sobel edge detection algorithm is made use of to encode the greyscale images into binary edge maps.
- This technique was developed by studying how we human beings remember others people's faces (remembering face's prominent features)

3) Histogram of oriented gradients (HOG):

This technique can be applied to detecting objects as well as faces.All images used are converted to greyscale and every pixel in this image in assigned an integer.

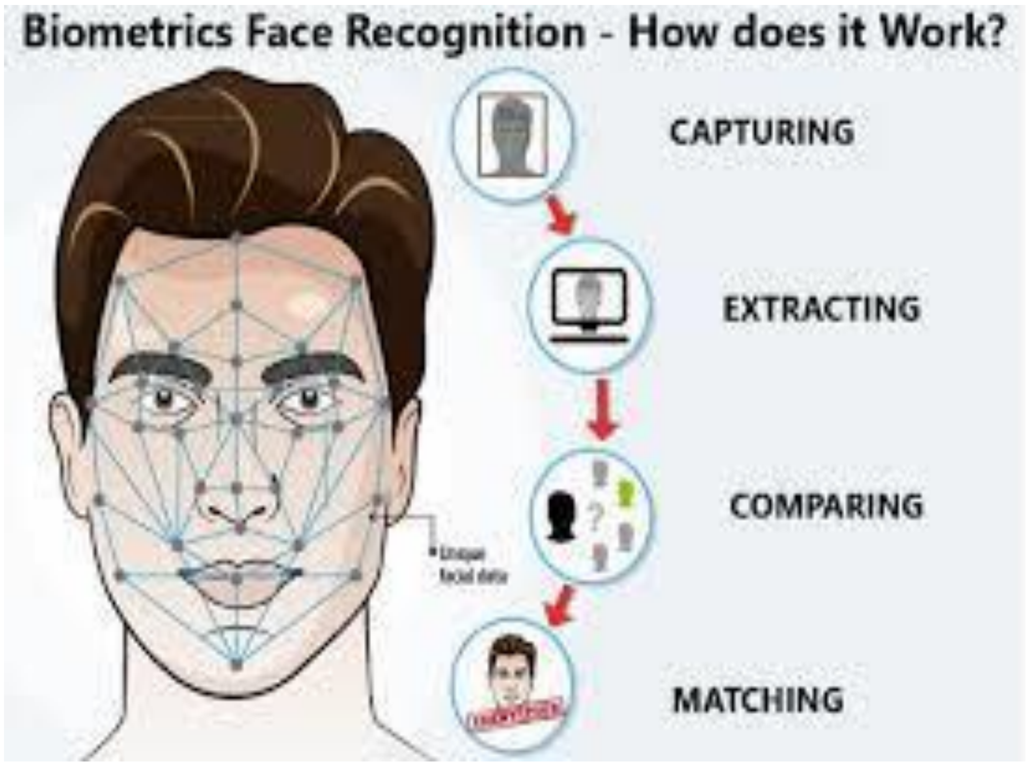
Every pixel compares its value to its neighbouring pixels.The primary motive is to find the dark regions of the face in the image.The direction pointing to that dark region will have a white arrow pointing towards it.This treatment is done for each pixel of the picture.

Gradient Calculation: After converting the image to grayscale, the gradient magnitude and direction are calculated for each pixel. This is typically done using derivative filters such as Sobel or Scharr filters. The gradient magnitude represents the rate of change of intensity at each pixel, while the gradient direction indicates the direction of the steepest change in intensity. These gradients provide information about the local edge orientations within the image.

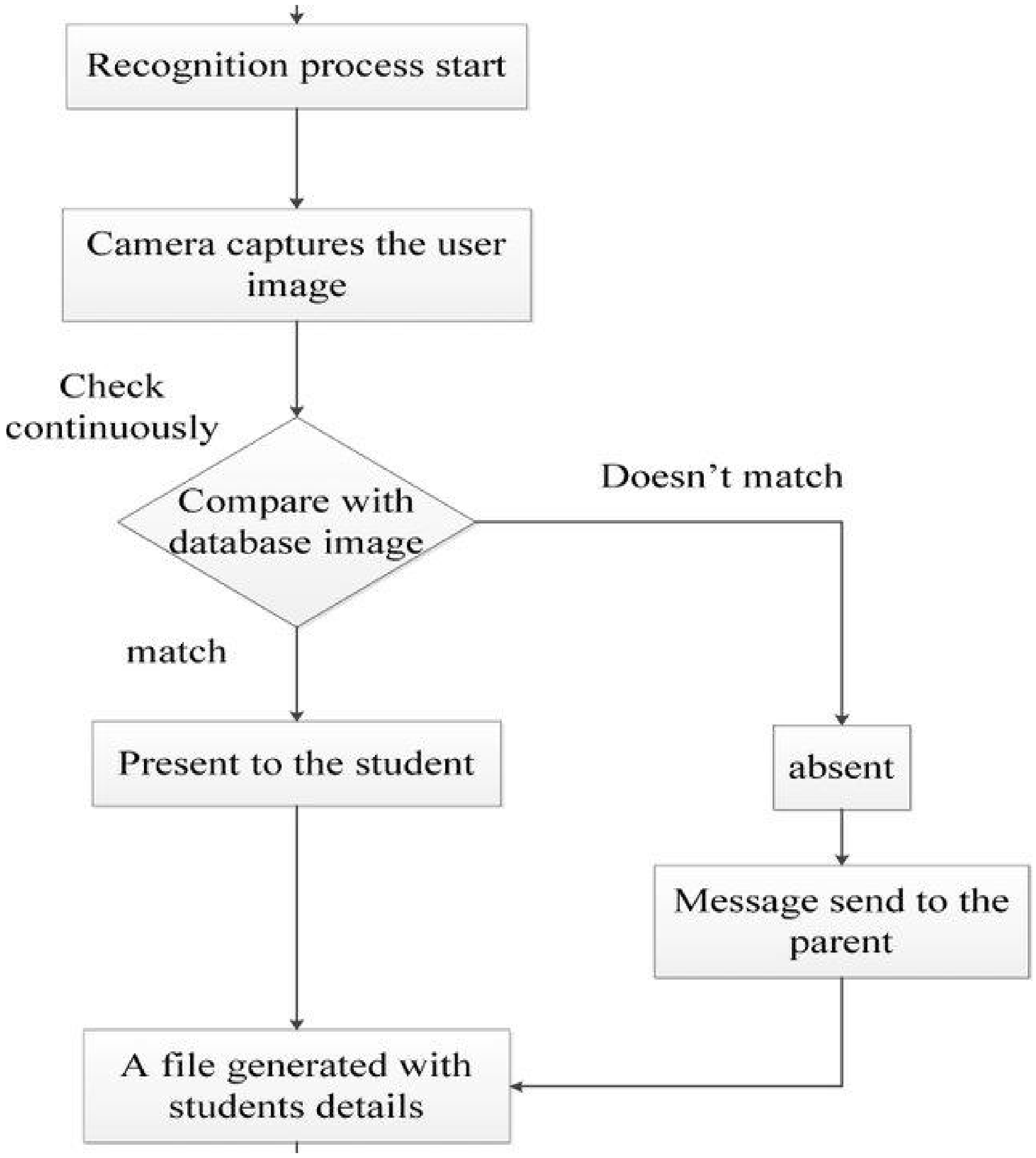
Histogram Calculation: The image is divided into small cells, typically 8x8 pixels in size. For each cell, a histogram of gradient orientations is computed based on the gradient magnitudes and directions within that cell. The histogram bins represent different orientation ranges (e.g., 0-20 degrees, 20-40 degrees, etc.), and the values in each bin correspond to the accumulated gradient magnitudes with orientations falling within that range.

Normalization and Block Processing: To improve robustness to changes in illumination and contrast, the histogram values are normalized within each block of cells. Blocks are defined as small groups of adjacent cells, and normalization is typically performed by dividing the histogram values by the sum of squares of the values within the block. This normalization process helps to reduce the influence of lighting variations and enhances the discriminative power of the features.

The HOG technique provides a powerful method for representing local object appearance and shape information in an image. By analysing the distribution of gradient orientations within small image regions, HOG descriptors capture important edge and texture information, making them effective for object detection tasks, including face detection.



Biometric Face Recognition [8.1]


Recognition process start
Camera captures the user image
Check continuously
Compare with database image
Doesn't match
match
Present to the student
absent
Message send to the parent
A file generated with students details

HOG technique[9.1]

Python GUI – Tkinter

tkinter is indeed one of the most commonly used methods for developing GUI (Graphical User Interface) applications in Python. It provides a simple and easy-to-use interface to the Tk GUI toolkit, which is included with Python installations on most platforms. Here's an expansion on creating GUI applications using tkinter:

1.) **Importing tkinter Module:** The first step is to import the tkinter module, which provides the necessary classes and functions for creating GUI components.

```
import tkinter as tk
```

2.) **Creating a Main Application Window:** Next, you create the main application window using the Tk() constructor. This window serves as the container for all the GUI components.

```
root = tk.Tk()

root.title("My GUI Application")  # Set title for the window
```

3.) **Adding Widgets:** You can add various widgets (GUI components) to the main window, such as buttons, labels, entry fields, etc. Each widget is created as an instance of a tkinter class, such as Label, Button, or Entry, and then placed within the main window using geometry managers like pack(), grid(), or place().

```
label = tk.Label(root, text="Hello, World!")

label.pack()

button = tk.Button(root, text="Click Me!", command=lambda: print("Button Clicked!"))

button.pack()
```

4.) **Configuring Widget Properties:** You can configure various properties of the widgets, such as text, colors, fonts, sizes, etc., using keyword arguments passed to the widget constructors.

```
  entry = tk.Entry(root, width=30)

  entry.pack()

  text = tk.Text(root, height=10, width=50)

 text.pack()
```

5.) **Handling Events**: You can define event handlers to respond to user interactions with the widgets, such as button clicks, mouse movements, keyboard input, etc. Event handlers are typically associated with specific widgets using the command parameter or the bind() method.

6.) **Handling Events:** You can define event handlers to respond to user interactions with the widgets, such as button clicks, mouse movements, keyboard input, etc. Event handlers are typically associated with specific widgets using the command parameter or the bind() method.

```
 def button_click():

 print("Button Clicked!")

button = tk.Button(root, text="Click Me!", command=button_click)

button.pack()
```

7.) **Running the Application:** Finally, you start the GUI application's event loop by calling the mainloop() method on the main window object. This method listens for user events and updates the GUI accordingly until the user closes the application window.

```
root.mainloop()
```

To create a tkinter app:

1). Importing the Module: Import the tkinter module, as you would with any other Python module. Note that in Python 2.x, the module is named 'Tkinter', while in Python 3.x, it's 'tkinter'.

```
import tkinter as tk  # for Python 3.x
```

2). Create the Main Window (Container): Create the main application window using the Tk() constructor.

```
root = tk.Tk()
```

3). Add Widgets to the Main Window: Add any desired widgets, such as buttons, labels, entry fields, etc., to the main window using appropriate tkinter classes (e.g., Button, Label, Entry).

```
label = tk.Label(root, text="Hello, tkinter!")

label.pack(

button = tk.Button(root, text="Click Me!")

button.pack()
```

4). Apply Event Triggers on Widgets: Apply event triggers (e.g., button clicks, mouse movements) on the widgets using event handling mechanisms provided by tkinter, such as the command parameter for buttons or the bind() method for other widgets.

```
def button_click():

  print("Button Clicked!")
```

```
button = tk.Button(root, text="Click Me!", command=button_click)

button.pack(
```

5). Run the Application: Start the GUI application's event loop by calling the mainloop() method on the main window object. This method listens for user events and updates the GUI accordingly until the user closes the application window.

```
root.mainloop()
```

There are two main methods used which the user needs to remember while creating the Python application with GUI.

The creation of a main window using Tkinter in Python and the use of the mainloop() method:

Creating the Main Window: Tkinter provides the Tk() method to create the main window of the application.

The syntax for creating the main window is:

```
m = tkinter.Tk(screenName=None, baseName=None, className='Tk', useTk=1)
```

screenName: Optional parameter specifying the name of the screen where the window will be displayed.

baseName: Optional parameter specifying the base name for the window.

className: Optional parameter specifying the class name for the window. This can be changed to customize the appearance or behavior of the window.

useTk: Optional parameter specifying whether to use Tk (set to 1 by default). mainloop() Method: Once the main window is created, the mainloop() method is called to start the event processing loop. The mainloop() method runs an infinite loop, waiting for events (such as button clicks or mouse movements) to occur and processing them as long as the window is not closed. This method is essential for keeping the application responsive and interactive while waiting for user input. The mainloop() method is typically called after creating the main window and adding all the necessary widgets and event handlers.

```
import tkinter

# Create the main window

m = tkinter.Tk()

# Customize the window title

m.title("My Application")

# Add widgets, event handlers, and other components to the window

# Start the event processing loop

m.mainloop()
```

This creates a main window with the title "My Application" using Tkinter.

After creating the window and adding any desired widgets or components, the mainloop() method is called to start processing events.

using the Tk() method to create the main window and the mainloop() method to start the event processing loop, you can create interactive GUI applications using Tkinter in Python.

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method**: It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method**: It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place() method**: It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

1. **Button:** To add a button in your application, this widget is used.
   The general syntax is:

   ```
   w=Button(master, option=value)
   ```

   master is the parameter used to represent the parent window.
   There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.
   - **activebackground:** to set the background color when button is under the cursor.
   - **activeforeground:** to set the foreground color when button is under the cursor.
   - **bg:** to set he normal backgroundcolor.
   - **command:** to call a function.
   - **font:** to set the font on the button label.
   - **image:** to set the image on the button.
   - **width:** to set the width of the button.
   - **height:** to set the height of the button.

```
import tkinter as tk
r = tk.Tk()
r.title(           'Counting Seconds')
button = tk.Button(r, text='Stop', width=25, command=r.destroy)
button.pack()
  r.mainloop()
```

2. Canvas: It is used to draw pictures and other complex layout like graphics, text and widgets.

The general syntax is:

```
w = Canvas(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border width in pixels.
- **bg**: to set the normal background color.
- **cursor**: to set the cursor used in the canvas.
- **highlightcolor**: to set the color shown in the focus highlight.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter
import * master = Tk()
w = Canvas(master, width=40, height=60)
w.pack()
canvas_height=20
canvas_width=200
y = int(canvas_height / 2)
w.create_line(0, y, canvas_width, y )
mainloop()
```

3. CheckButton: To select any number of options by displaying a number of options to a user as toggle buttons.

The general syntax is:

```
w = CheckButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **Title**: To set the title of the widget.
- **activebackground**: to set the background color when widget is under the cursor.
- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal background Steganography Break Secret Code: Attach a File:nd color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.

```
from tkinter
import * master = Tk()
var1 = IntVar()
Checkbutton(master,text='male',variable=var1).grid(row=0,sticky=W)
var2=IntVar()
Checkbutton(master,text='female',variable=var2).grid(row=1,sticky=W)
mainloop()
```

4. Entry: It is used to input the single line text entry from the user. For multi-line text input, Text widget is used.

The general syntax is:

```
w=Entry(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd:** to set the border width in pixels.
- **bg:** to set the normal background color.
- **cursor:** to set the cursor used.
- **command:** to call a function.
- **highlightcolor:** to set the color shown in the focus highlight.
- **width**: to set the width of the button.
- **height:** to set the height of the button.

```
from tkinter
import * master = Tk()
Label(master, text='First Name').grid(row=0)
Label(master, text='Last Name').grid(row=1)
e1 = Entry(master)
e2 = Entry(master)e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```

5. Frame: It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.

The general syntax is:

```
w = Frame(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- highlightcolor: To set the color of the focus highlight when widget has to be focused.
- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used.
- width: to set the width of the widget.

- height: to set the height of the widget.

```
from tkinter
import * root = Tk()
frame = Frame(root)
frame.pack()
bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )
redbutton = Button(frame, text = 'Red', fg ='red')
redbutton.pack( side = LEFT)
greenbutton = Button(frame, text = 'Brown', fg='brown')
greenbutton.pack( side = LEFT )
bluebutton = Button(frame, text ='Blue',fg ='blue')
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)
root.mainloop()
```

6. **Label:** It refers to the display box where you can put any text or image which can be updated any time as per the code.

The general syntax is:

```
w=Label(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg**: to set he normal background color.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.

- **image**: to set the image on the button.
- **width**: to set the width of the button.
- **height**: to set the height of the button.

```
from tkinter
import *
root = Tk()
w = Label(root, text=’E-attendance system')
w.pack()
root.mainloop()
```

7. Listbox: It offers a list to the user from which the user can accept any number of options.

The general syntax is:

```
w = Listbox(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- highlightcolor: To set the color of the focus highlight when widget has to be focused.
- bg: to set he normal background color.
- bd: to set the border width in pixels.
- font: to set the font on the button label.
- image: to set the image on the widget.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
        from tkinter import *

top = Tk()
```

```
Lb = Listbox(top)

Lb.insert(1, 'Python')

Lb.insert(2, 'Java')

Lb.insert(3, 'C++')

Lb.insert(4, 'Any other')

Lb.pack()

top.mainloop()
```

8. MenuButton: It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality.

The general syntax is:

```
w = MenuButton(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **active background**: To set the background when mouse is over the widget.
- **active foreground**: To set the foreground when mouse is over the widget.
- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menu button.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.
- **highlights**: To set the color of the focus highlight when widget has to be focused.

```
from tkinter import *
top = Tk()
mb = Menubutton ( top, text = &quot;GfG&quot;)
mb.grid()
mb.menu = Menu ( mb, tearoff = 0 )
mb[&quot;menu&quot;] = mb.menu
cVar = IntVar()
aVar = IntVar()
mb.menu.add_checkbutton ( label ='Contact', variable = cVar ) mb.menu.add_checkbutton ( label =
'About', variable = aVar )
mb.pack()
top.mainloop()
```

9. Menu: It is used to create all kinds of menus used by the application.

The general syntax is:

```
w = Menu(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **title**: To set the title of the widget.
- **activebackground**: to set the background color when widget is under the cursor.
- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.

```
from tkinter import *
root = Tk()
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label='File',menu=filemenu) filemenu.add_command(label='New')
filemenu.add_command(label='Open...')
filemenu.add_separator()
filemenu.add_command(label='Exit', command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label='Help',menu=helpmenu) helpmenu.add_command(label='About')
mainloop()
```

10. Message: It refers to the multi-line and non-editable text. It works same as that of Label.

The general syntax is:

```
w = Message(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border around the indicator.
- **bg**: to set he normal background color.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
main = Tk()
```

```
ourMessage ='This is our Message'
messageVar = Message(main, text = ourMessage) messageVar.config(bg='lightgreen')
messageVar.pack( )
main.mainloop( )
```

11. RadioButton: It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.

The general syntax is:

```
w = RadioButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas.

Some of them are listed below.

- **activebackground**: to set the background color when widget is under the cursor.
- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the label in characters.
- **height**: to set the height of the label in characters.

```
from tkinter import *
root = Tk()
v = IntVar()
Radiobutton(root, text='GfG', variable=v, value=1).pack(anchor=W) Radiobutton(root, text='MIT',
variable=v, value=2).pack(anchor=W) mainloop()
```

12. Scale: It is used to provide a graphical slider that allows to select any value from that scale.

The general syntax is:

```
w = Scale(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **cursor**: To change the cursor pattern when the mouse is over the widget.
- **activebackground**: To set the background of the widget when mouse is over the widget.
- **bg**: to set he normal background color.
- **orient**: Set it to HORIZONTAL or VERTICAL according to the requirement.
- **from_:** To set the value of one end of the scale range.
- **to**: To set the value of the other end of the scale range.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.

```
from tkinter import *
master = Tk()
w = Scale(master, from_=0, to=42)
w.pack()
w = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w.pack()
mainloop()
```

13. Scrollbar: It refers to the slide controller which will be used to implement listed widgets.

The general syntax is:

```
w = Scrollbar(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **width**: to set the width of the widget.
- **activebackground**: To set the background when mouse is over the widget.
- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.

```
from tkinter import *
root = Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill = Y )
mylist = Listbox(root, yscrollcommand = scrollbar.set )

for line in range(100):
        mylist.insert(END, 'This is line number' + str(line))
mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )
mainloop()
```

14. Text: To edit a multi-line text and format the way it has to be displayed.

The general syntax is:

```
w =Text(master, option=value)
```

There are number of options which are used to change the format of the text. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **insert background**: To set the background of the widget.
- **bg**: to set he normal background color.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
root = Tk()
T = Text(root, height=2, width=30)
T.pack()
T.insert(END, 'GeeksforGeeks\nBEST WEBSITE\n')
mainloop()
```

15. TopLevel: This widget is directly controlled by the window manager. It don’t need any parent window to work on.

The general syntax is:

```
w = TopLevel(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

Some of them are listed below.

- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.

- **cursor**: To appear the cursor when the mouse over the menubutton.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
root = Tk()
root.title('GfG')
top = Toplevel()
top.title('Python')
top.mainloop()
```

16. Spin Box: It is an entry of ‘Entry’ widget. Here, value can be input by selecting a fixed value of numbers.

The general syntax is:

```
w = SpinBox(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

The Spin Box widget in Tkinter provides a convenient way to input numerical values within a predefined range. Here's an expansion of its usage and options:

master: The parent widget (e.g., a frame or a window) where the Spin Box will be placed.

option=value: Optional parameters that can be used to customize the behaviour and appearance of the Spin Box widget.

Options:

from_: Specifies the lower bound of the range of values allowed in the Spin Box.

to: Specifies the upper bound of the range of values allowed in the Spin Box.

increment: Specifies the amount by which the value changes when the user clicks the up or down arrow buttons.

text variable: Associates a Tkinter variable (e.g., String Var, IntVar) with the Spin Box, allowing you to get or set its value programmatically.

wrap: If set to True, the Spin Box value wraps around when it reaches the minimum or maximum value.

state: Sets the state of the Spin Box widget (e.g., "normal" for enabled, "disabled" for disabled).

format: Specifies the format of the displayed value, such as the number of decimal places or leading zeros.

command: Specifies a callback function that is called whenever the Spin Box value changes.

font: Sets the font style and size of the Spin Box text.

width: Specifies the width of the Spin Box widget in characters.

background or bg: Sets the background colour of the Spin Box.

foreground or fg: Sets the text colour of the Spin Box.

Usage:

Create a Tkinter window or frame as the parent widget.

Create a Spin Box widget and specify its parent and any desired options.

Pack or grid the Spin Box widget within the parent widget to display it on the GUI.

Optionally, associate a Tkinter variable with the Spin Box using the text variable option to track or manipulate its value dynamically.

```
import tkinter as tk

root = tk.Tk()

spinbox_var = tk.IntVar()

spinbox = tk.Spinbox(root, from_=0, to=100, textvariable=spinbox_var, increment=5, width=10)

spinbox.pack()

def on_change():

    print("SpinBox value:", spinbox_var.get())

spinbox.config(command=on_change)

root.mainloop()
```

reates a SpinBox widget with values ranging from 0 to 100, with an increment of 5, and displays it in a Tkinter window. The SpinBox value is associated with an IntVar variable spinbox_var, and a callback function on_change() is called whenever the SpinBox value changes.

Some of them are listed below.

- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.
- **command**: To call a function.
- **width**: to set the width of the widget.
- **active background**: To set the background when mouse is over the widget.
- **disabled background**: To disable the background when mouse is over the widget.
- **from_:** To set the value of one end of the range.

- **to**: To set the value of the other end of the range.

```
from tkinter import *
master = Tk()
w = Spinbox(master, from_ = 0, to = 10)
w.pack()
mainloop()
```

17. Panned Window It is a container widget which is used to handle number of panes arranged in it.

The general syntax is:

```
w = PannedWindow(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg:** to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menu button.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
m1 = PanedWindow()
m1.pack(fill = BOTH, expand = 1)
 left = Entry(m1, bd = 5)
m1.add(left)
m2 = PanedWindow(m1, orient = VERTICAL)
m1.add(m2)
top = Scale( m2, orient = HORIZONTAL)
m2.add(top)
```

```
mainloop()
```

Through a GUI, users have the ability to interact with the system. Users will primarily have access to three choices here: mark registration, faculty registration, and student registration being present. It is expected of the students to fill out the student registration form with all necessary information. Following the registration button’s click, the webcam establishes automatically, presenting the window represented in Figure 3 and beginning to identify faces inside the frame.

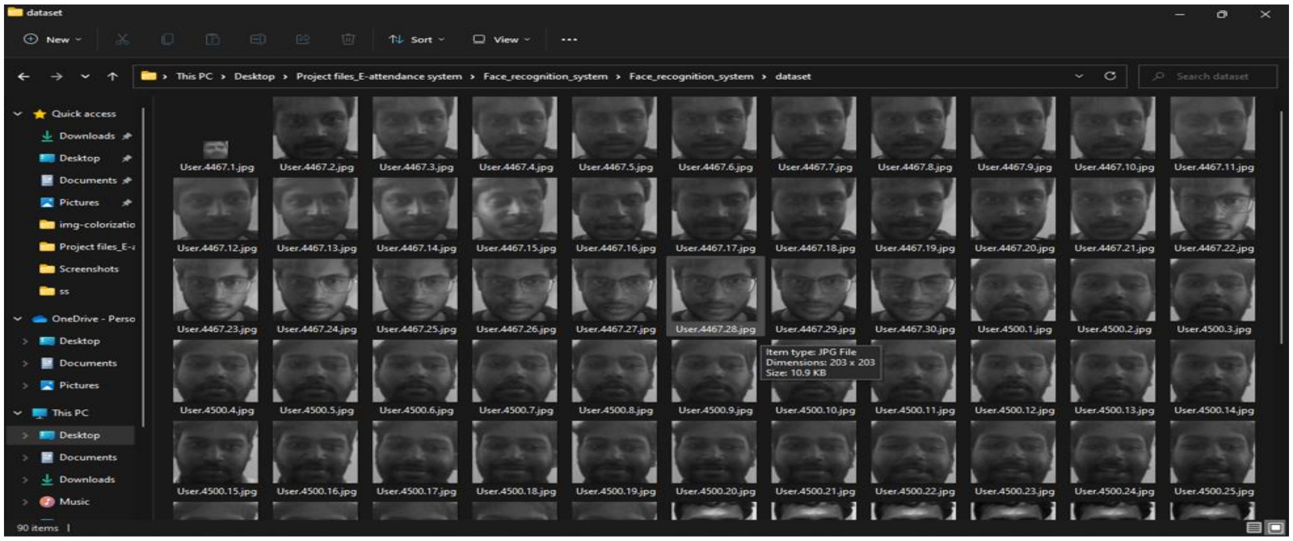
After then, it will begin to automatically take pictures until 60 samples have been collected or CRTL+Q is pressed. During pre-processing, these photos will be kept in the images used for training folder. The faculty members are required to fill out the instructor registration form with their email address

and the appropriate course codes. This is essential since the corresponding to the departments will eventually receive a list of those who are absent.

Each training session, the necessary trainer needs to input their course code. The recording device will then turn on by

itself after the course code has been entered. Figure 4 displays the facial recognition window where two enrolled students are recognized; if the individuals hadn’t registered, the window would have shown” unidentified.” The window will remove when you hit CTRL+Q, and the attendance will be modified

in the spreadsheet. The identities of the absentees will then be sent to the appropriate faculty person.

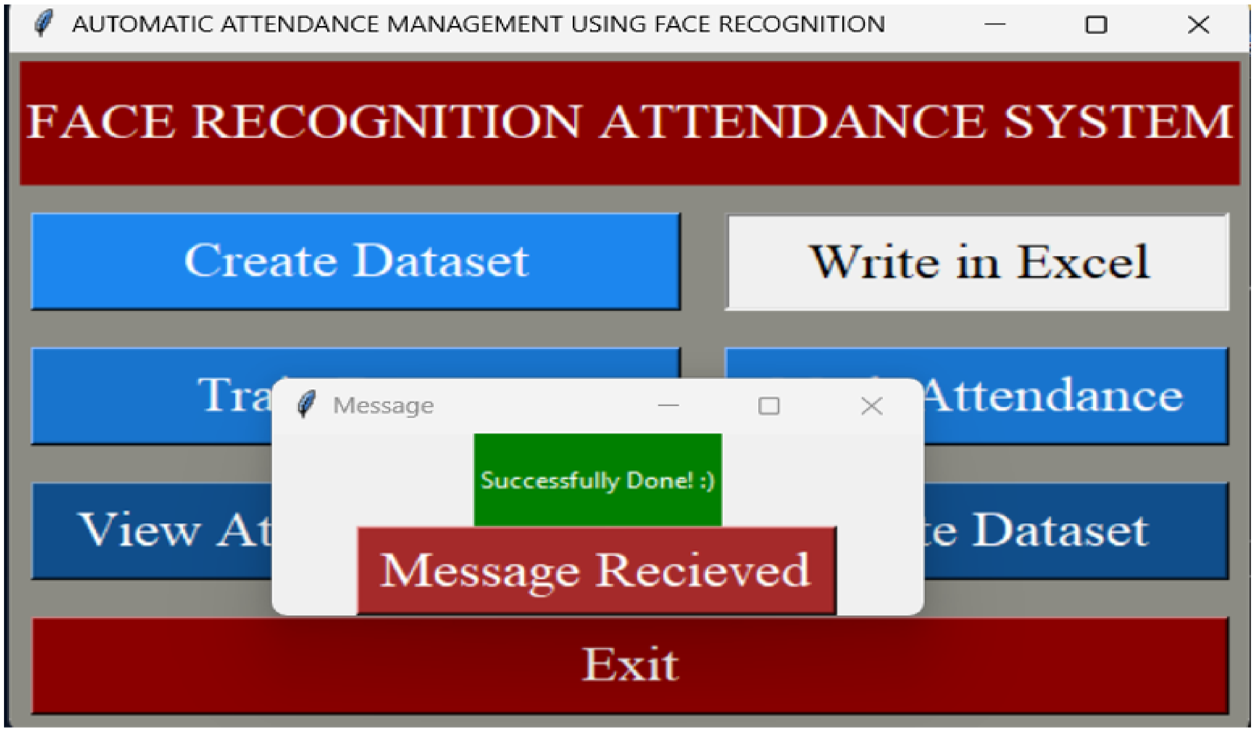


Face Detection by the System.[10.1]

• Evaluation Metrics: Benchmark datasets were used to assess the SSD method's real-time object detection execution. A number of metrics were calculated, including mean average precision (mAP), which gauges how accurately objects are detected. It is also possible to report additional metrics like recall, F1 score, and precision.

• Detection Accuracy: The LBPH algorithm demonstrated high detection accuracy across various object classes. The mAP score indicated the overall performance of the system in terms of correctly identifying and localizing objects. Results showed that the algorithm achieved competitive or state-of-the-art performance compared to other object detection methods.

• Scale and Variability: The LBPH algorithm was tested on objects of different scales and aspect ratios. Results showed that the algorithm effectively handled scale variations, detecting both small and large objects accurately. The use of multi-scale feature maps and default boxes contributed to the robustness of the system.


AUTOMATIC ATTENDANCE MANAGEMENT USING FACE RECOGNITION
FACE RECOGNITION ATTENDANCE SYSTEM
Create Dataset
Write in Excel
Tra
Attendance
View At
e Dataset
Exit
Message

Successfully Done! :)

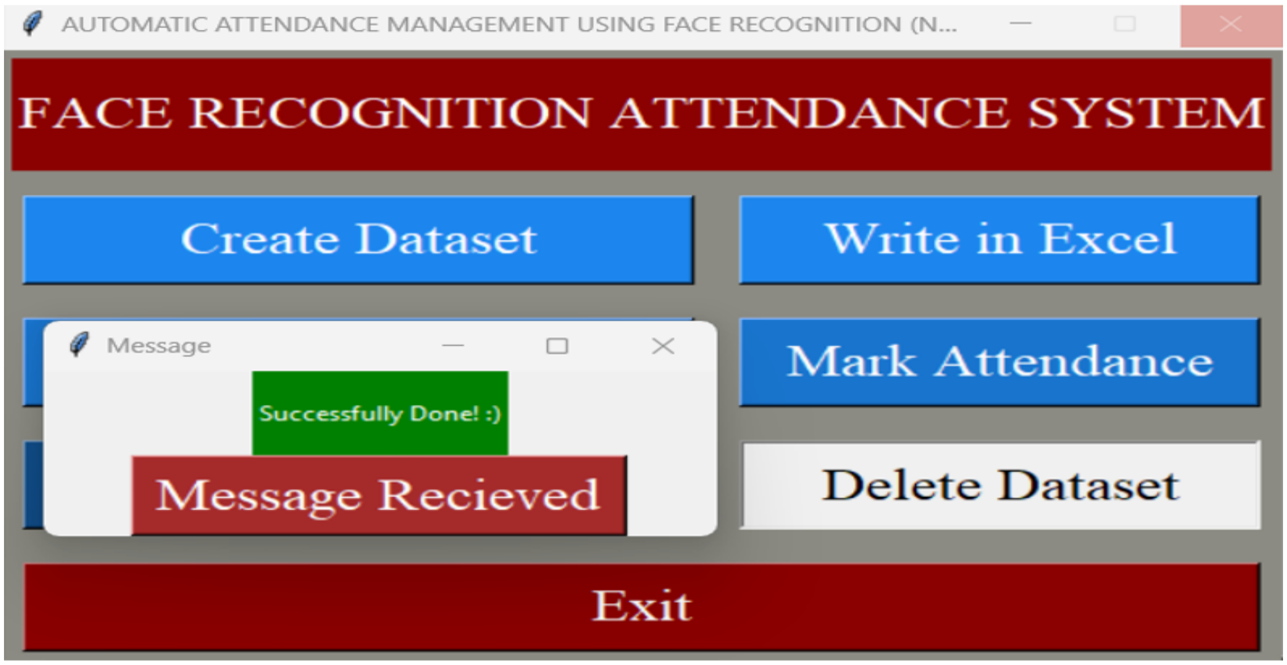
Message Recieved

Writing the attendance by the System[11.1]

• Limitations and Challenges: The discussion also addressed the limitations and challenges encountered during the implementation of the algorithm. For example, the algorithm may face difficulties in accurately detecting objects with extreme aspect ratios or occluded objects. Strategies to mitigate these limitations, such as data

augmentation techniques or architecture modifications, were proposed for future enhancements.

• Practical Applications: The practical applications of the Facial Recognition system based on the algorithm were discussed. These may include autonomous driving, surveillance systems, or robotics, where fast and accurate object detection is crucial. The potential impact and benefits of the system in real-world scenarios were highlighted.



Deleting the registered student from the dataset by the application.[12.1]

• Future Directions: The discussion section also provided insights into future research directions and possible improvements to the algorithm. Areas for further exploration may include incorporating advanced techniques such as attention mechanisms, exploring novel training strategies, or addressing specific challenges in object detection, such as handling crowded scenes or detecting objects i

6.CONCLUSION

The future scope for the Attendance Management System project includes several potential enhancements and expansions. One key improvement is the capability to discontinue particular students from attendance tracking, effectively removing them from future attendance calculations. Additionally, integrating a bar code reader-based attendance system can streamline attendance tracking processes by allowing students or staff to scan unique bar codes for attendance marking, reducing manual effort and errors. Enhancements in attendance administration features, such as tracking working hours, breaks, and login/logout times, can provide deeper insights into staff productivity and performance. Integration of advanced biometric and facial recognition devices enhances security and accuracy in real-time attendance tracking. Real-time integration with various attendance devices enables seamless synchronization of attendance data across different locations. Customizable reporting and analytics features facilitate detailed attendance analysis, aiding in identifying patterns and areas for improvement. Furthermore, mobile accessibility and remote attendance tracking through mobile applications or web interfaces offer flexibility and convenience for managing attendance data from anywhere. Continuous updates and expansions in the Attendance Management System contribute to improved efficiency, accuracy, and transparency in attendance tracking and administration, leading to better workforce management and operational effectiveness.

REFERENCES:

1 . Akbar, Md Sajid, et al. ”Face Recognition and RFID Verified Attendance System.” 2018 International Conference on Computing, Electronics Communications Engineering (iCCECE). IEEE, 2018.

2 5. Okokpujie, Kennedy O., et al. ”Design and implementation of a student attendance system using iris biometric recognition.” 2017 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2017

3 6. Rathod, Hemantkumar, et al. ”Automated attendance system using machine learning approach.” 2017 International Conference on Nascent Technologies in Engineering (ICNTE). IEEE, 2017.

4 7. Siswanto, Adrian Rhesa Septian, Anto Satriyo Nugroho, and Maulahikmah Galinium. ”Implementation of face recognition algorithm for biometrics-based time attendance system.” 2014 International Conference on ICT For Smart Society (ICISS). IEEE, 2014.

5 8. Lukas, Samuel, et al. ”Student attendance system in classroom using face recognition technique.” 2016 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2016

6 9. Salim, Omar Abdul Rhman, Rashidah Funke Olanrewaju, and Wasiu Adebayo Balogun. ”Class attendance management system using face recognition.” 2018 7th International Conference on Computer and Communication Engineering (ICCCE). IEEE, 2018

7 Smitha, Pavithra S Hegde, Afshin, ”Facial Recognition Based Attendance Management System”, june 2020 International Journal of Engineering Research and V9(05).

8 Hapani, Smit, et al. ”Automated Attendance System Using Image Processing.” 2018 Fourth International Conference on Computing).

9 Rathod, Hemantkumar, et al. ”Automated attendance system using machine learning approach.” 2017 International Conference on Nascent Technologies in Engineering (ICNTE). IEEE, 2017.

10 Reena Gunjan, Anannya Mital, Muskan Jain. ”Smart Capture - An Advanced Attendance System using facial Recognition”. Conference: 2023 1st International Conference on Cognitive Computing and Engineering Education (ICCCEE)

11 Shireesha Chintalpati, M.V. Raghunadh. ” Automated attendance based management system based on face recognition algorithms”. Conference: 2013 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).

12 Shubhobrata Bhattacharya, Gowtham Sandeep Nainala, Prosenjit Das, Aurobinda Routray. ”Smart Attendance Monitoring System (SAMS): A Face Recognition Based Attendance System for Classroom Environment”. Conference: 2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT).

13 Shreyak Sawhney, Karan Kacker, Samyak Jain Shailendra Narayan Singh. ”Real Time Smart Attendance System Using Face Recognition Techniques”. Conference: 2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence).

14 Khem puthea, Rudy Hartanto, Risanuri Hidayat. ” A review paper on attendance marking system based on face recognition”. Conference: 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)

15 Ashwin Raj, Aparna Raj, Imteyaz Ahmad. ” Smart Attendance Monitoring System With Computer Vision Using IOT ”. February 2021Journal of Mobile Multimedia.

16 Husam Aizaq. ” Student Attendance System Using iTag’s Beacon Detection”. Conference: 2023 20th International Multi-Conference on Systems, Signals Devices (SSD).