Face Recognition Based Attendance System

[1]S Bazith, *[2]Dr. Preety Savant

[1]PG Student, [2]Assistant Professor
Department of MCA, School of CS and IT
JAIN (Deemed-to-be-University), Bangalore, India

***Abstract*: Traditional ways are becoming less popular as the globe moves toward a digital future. People are increasingly reliant on digital technologies. In this scenario, facial recognition has emerged as a digital pioneer. Every day, our faces are examined and processed for a variety of reasons, including security, authentication, and identification, among others. Facial Recognition can also be used to track attendance at schools, colleges, and other places.**

This allows us to manage daily student attendance without having to interact with the model, making it more efficient and convenient. Objects will be detected using a camera, but faces will be targeted using a different algorithm. We can verify that faces are identified and matched as precisely as possible using a distance method. And because each student's information will be saved in the database, the camera will compare the dimensions of the student's faces to the photos stored in the database whenever the student's face is taken by the camera. It will store the student as a present until the face is more exact to the dimension of the face in the picture. This paper helps to understand how face recognition works and the different methodologies proposed by different researchers.

Index Terms: Imperfect facial data; face recognition; PYTHON, C++, OOP; OpenCV, AdaBoost, HAAR Classifier, DCNN.

INTRODUCTION

Face recognition is commonly recognized as one of the most effective biometric authentication systems. In 1965, Woody Bledsoe invented facial recognition. His main objective was to provide a database of 10 different people's face images to the computer and see if it could recognize new shots of each of them. He taught his system to partition a face into features and then compare their distances. This will aid in a more accurate identification of the face. Today, we may utilize OpenCV, an open-source library, to execute automatic facial recognition systems using computer vision applications such as facial identification.
Face recognition is becoming increasingly popular among today's generation. Every day, gadgets ranging from cellphones to cameras capture our faces. These can also be used to identify criminals and in other crime-related situations. To find the criminals, photographs of individuals are taken and matched against a database. Facial recognition is used in a variety of situations.

Facial recognition software maps, analyses, and validates the identification of a face in a photograph or video. One of the most successful surveillance techniques ever invented, it is widely considered as one of the most effective surveillance strategies ever devised.

The first step is to locate the item in the image, which in this case is the face [1][2]. Everything that is seen by a camera is captured. As a result, we must train our system to recognize only the face and not other items. The system will draw a box around our face once it learns to recognize faces[3], signifying that it has detected our face. We need to separate them since each face has a unique biometric[3]. Some people have little faces, while others have large ones. As a result, the system calculates the distance between the faces as well as their proportions in order to distinguish and precisely identify them. Snapchat, for example, employs this filter to assess the face structure and apply the appropriate filter to the face. The terms "detection" and "recognition" are not interchangeable.

THE ILLUMINATION PROBLEM

Automated face recognition systems have been enabled thanks to recent advances in automated face analysis, pattern recognition, and machine learning. Face recognition is a natural process that humans engage in on a daily basis. However, because people have distinct faces, utilizing a system to recognize a face remains a difficult task. Physiological and behavioral patterns could be used to create diverse biometric features. Due to lighting and viewing direction, when photos of a human face are taken, there is a variation in the images of the faces. It's due to a variety of circumstances, including pose, attitude, lighting, and others. Even though the facial recognition system has come a long way, it still has issues to address in order to produce a more accurate system. This will aid the system's ability to function effectively in the face of a variety of issues (Figure 1).

Figure 1: A picture taken of a student with multiple lightning resulting to various illuminations. Thus difficult to identify.

Changes in the direction of the light source can make the same person with the same facial expression look drastically different, as shown in figure 1. Face recognition problems can be classified as either a classifier pattern recognition or a machine learning challenge. Face recognition systems are mostly built on three key steps. Before the steps are carried out, a set of labeled and unlabeled data, in this case photographs of people, is created. The system is given these faces in order to recognize them based on their attributes. Face detection is the first step in identifying each person in the data set by locating the dace in the image. The feature vector is then used to measure the features. Finally, a classifier is trained to assign a label to each face vector that corresponds to a person's identification.

PROBLEM DEFINITION

Taking manual attendance is a time-consuming task. Instead of squandering time, it may be put to better use. There's a chance the teacher will miss a few students because they have to take attendance for more than 50 students. Another option is for a student to take attendance for other students who are absent. We can employ deep learning and facial recognition ideas to avoid such a scenario. We can recognize faces and things using facial recognition. However, facial recognition is insufficient because the system is unable to distinguish between faces and other items. As a result, we require deep learning. We can educate our system to solve issues using Deep Learning, in this case, detecting faces using biometrics and facial measurements. Each face has its own characteristics and measurements. A machine can detect these faces, but we need to teach our system to match faces accurately in order to find the specific ones. As a result, Deep Learning can be used. We can achieve automatic attendance by employing this real-time facial recognition technology.

The main purpose of this paper is to learn how to use facial recognition to take attendance. The machine will learn to recognise students' faces based on their facial structure and unique features, give them attendance, and save their names in an excel sheet with time stamps when the system has been loaded with various students' photographs. easing the work of the faculties in terms of attendance.

LITERATURE SURVEY

Face recognition is a technique for detecting and identifying persons. The difficulty stems from the fact that most people's faces have a similar shape or structure, making it difficult to distinguish between them. As a result, a standard recognition technique would be ineffective in distinguishing between them.

According to a survey of all biometric-based techniques, face recognition is the most accurate at detecting a person's identification when compared to other biometric-based approaches such as PIN or even password. While remembering passwords or pins can be difficult at times, human faces rely on physiological or behavioral characteristics to identify an individual. It's also possible that the pin or password will be copied. Biological characteristics, on the other hand, cannot be changed, lost, stolen, or invented.

However, like with any advantages, there are disadvantages. Face recognition systems often recognize faces based on the frontal view of an individual because the distinctions between them appear to be minor. It's not uncommon for people from all over the world to have the same facial shape. As a result, face recognition algorithms assess a person's look based on a range of factors. These elements are divided into two categories: a) Extrinsic variables are solely determined by facial physical characteristics. It's further broken down into intrapersonal and interpersonal components. Intrapersonal is based on a single person's characteristics (facial hair, hair style, etc.), whereas interpersonal is based on the characteristics of numerous persons (gender, race, ethnicity, etc.). b) Extrinsic variables are influenced by the environment (lighting, illumination, poses, resolution, focus, noise, and so on), which alter the appearance of the face in various situations.

The author of [2] utilizes OpenCV to collect and identify aspects of a face in order to recognize a human's individual structure. The Haar Cascade Classifier was a famous technique used by OpenCV. It's an object-finding algorithm. It's a machine learning algorithm that trains a cascade function using tones of photos (Figure 2). The photographs are divided into two categories: target and non-target. This will aid in the system's recognition of target and non-target images. These characteristics are made up of various black and white rectangular combinations. Each feature computation finds the total number of pixels under white and black rectangles.

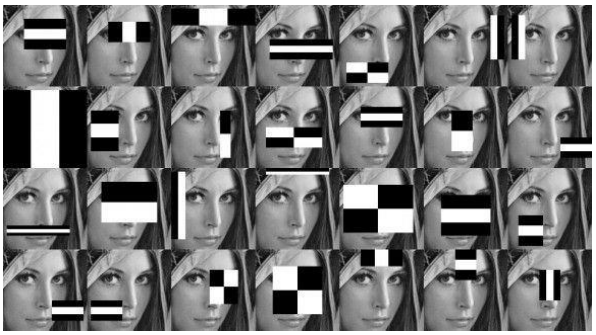

Fig 2: A general HAAR Classifier

Identifying the face in an image, extracting features, determining template compatibility, and declaring matching templates are all aspects of the model. The author uses an OV5647 CMOS image sensor to take a real-time image of the face, which is connected to a Raspberry Pi. The face is captured by scanning the image and recognizing different qualities to distinguish it from other faces. Once a face has been discovered, the HAAR classifier algorithm analyses it and outputs a rectangular box for each detected face. Even if there is just one object in the frame, the classifier may detect a large number of them. As a result, the author employs the Haar Classifier packages' OpenCV and SimpleCV for post-image processing in order to retrieve the exact coordinates of the face.
The author of [5] use Adaboost, an ensemble learning technique. It functions in a similar way to the thumb of the rule, which learns through trial and error. It requires a group of classifiers known as "weak learners" or "basic learners." It combines them into a powerful classifier. AdaBoost is used to train the system. The error rate for weak learners must be fewer than 50%. As a result, the guessing must be less than random.
To recognize faces in [3], the author employs a local binary pattern (LBP)-AdaBoost architecture. It is considered quick and insensitive to variations in illumination. It can recognize faces and eyes of various sizes from afar. Accuracy is one of the key issues that arise when performing automatic facial recognition. Because of lightning, illumination, and other circumstances, the system's accuracy may vary, causing the system to fail to distinguish faces. As a result, we can use the distance technique to solve this problem (Figure 3). The lower the distance, the higher the accuracy. Accuracy is one of the key issues that arise when performing automatic facial recognition. Because of lightning, illumination, and other circumstances, the system's accuracy may vary, causing the system to fail to distinguish faces. As a result, we can use the distance technique to solve this problem. The greater the accuracy, the shorter the distance.

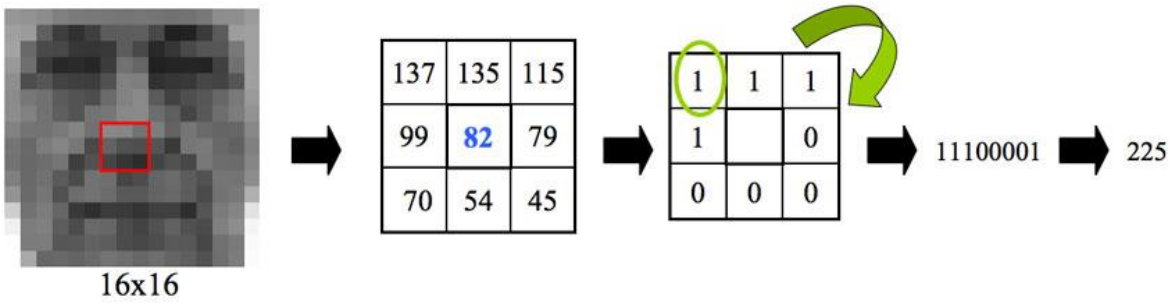

Figure 3: LBPH converts the image into 3x3 window and compare the pixel at center with its surrounding pixels.

The author of [1] analyses facial recognition with a DCNN (Deep Convolutional Neural Network). Face Recognition may encounter obstacles and challenges when dealing with input photographs, such as facial expressions or persons wearing hats or spectacles, among other things. As a result, the DCNN technique has three steps: face detection, analysis, and feature extraction. It improves the solution by adjusting the parameters to achieve the best possible outcome. The author also makes use of machine learning methods including DT, KNN, and SVM. The face of a person is a complex structure that evolves over time. The intricacy and dimensions of the face, as well as how to capture it. Each face has a unique biometric that is utilized to distinguish it from the others, allowing for more precise identification.

IMPLEMENTATION

Preparation

Before implementation we need to make sure certain application are installed in our system to make sure that system meets the requirement to run face detection. First is to identify any algorithm which allows the system to run face detection. Harr Cascade is the algorithm that was employed to carry out this job. It's an Object Detection Algorithm that's used to find faces in photos or videos.

Then you'll need to install python and opencv, among other things. Go to the python directory and check for opencv libraries once the installation procedure is complete. A file named **'haarcascade frontalface default.xml'** will be saved in the folder (Figure 4). It should be copied and pasted into the same working directory.

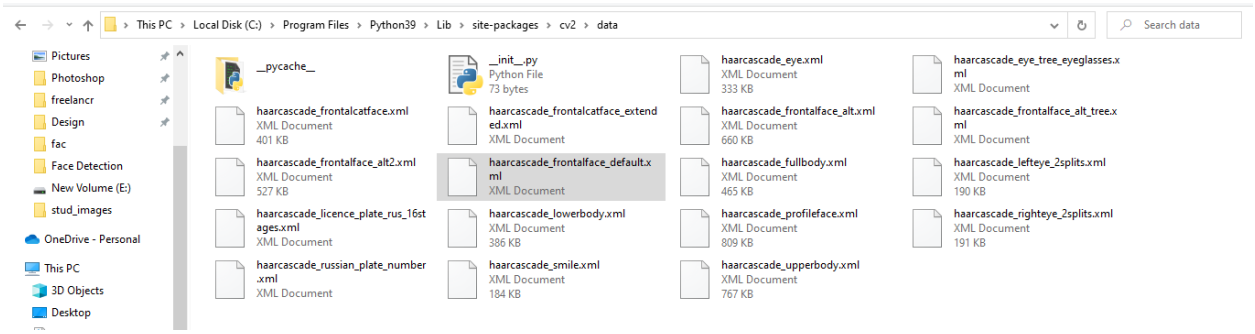

Figure 4: Location of the 'haarcascade frontalface default.xml' which will be used to detect faces.

This project's face detection and identification implementation is broken down into four sections.

A. Collection the face data

Before doing any type of face detection, we must first collect data, in this case images of people, in order to train the system to correctly identify the person in the image. As a result, we'll build a variable that will be used to initialize the haar cascade classifiers. Because the dimensions of an image acquired by a camera are so large, identifying a face among them becomes challenging. Thus we must reduce the image dimension for better output.

To accomplish so, we'll need to use the CascadeClassifier method to import cv2 and numpy. The second step is to load the image and convert it to gray-scale when the appropriate libraries have been imported. Because the system only knows 0s and 1s, it has a hard time distinguishing and identifying colours. As a result of the conversion, the system can quickly identify the image. Because there is only one black-white channel, the explanation is simple to comprehend and computationally less intensive We need to try to pinpoint the exact feature in our face once the image has been transformed from RGB to Gray. DetectMultiScale is an in-built function in the Haar Cascade classifier. This function aids in the extraction of specific features from a gathered image (Figure 5). The detectMultiScale function returns four values: x-coordinate, y-coordinate, width(w), and height(h) of the discovered face feature. We must draw a rectangle around the recognized face based on these values.

Figure 5: Original Image to Gray Image after conversion.

We now need to set up the system camera and read the image from it. The id value of a system camera is usually assumed to be zero. When the camera is turned on, it will begin to detect the face in front of it. The image is then resized and converted from RBG to Gray. Later, a condition will be specified using waitKey(), stating that the classifier will continue until the id exists or 100 photos are not acquired. The system prompts a message and closes immediately once it has finished capturing the face (Figure 6).

Figure 6: Using Haar Classifier capturing data set

B. Training the model

It is not enough for a system to store multiple photos of human faces in order to correctly recognize them. Before we can do that, we need to train the system how to recognize faces and correctly identify them. The basic goal of training a system is to make it

able to recognize faces in a variety of situations, including lighting variations, low resolution, occlusion, and so on. We can use a popular face recognition algorithm called Local Binary Patterns Histograms to train our model (LBPH). Local Binary Pattern (LBP) is an effective texture operator that labels pixels in an image by thresholding each pixel's neighborhood and treating the result as a binary number.

As mentioned above, the first stage in training a model is to acquire a dataset with facial photos of the people we wish to recognize. We'll give each collection of photographs a unique identification so that we can readily distinguish and identify them. In this situation, the student's registration ID serves as a unique identifier for each batch of photographs. This will aid the algorithm's ability to recognize, input, and output images. The collected photos are then combined into a more intermediate image that better highlights the human facial structure by highlighting facial traits. This is done using the radius and neighbor's parameters that have been provided. The radius surrounding the central pixel is represented by this circular local binary pattern.

Images, as we all know, are a collection of various pixels. Each of these are made up of little squares. We can build the whole image by putting them side by side. A single pixel in an image is thought to contain the least amount of information possible. The value of pixels in each image ranges from 0 to 255. The model turns the grayscale image into a 3x3 matrix in order to comprehend the images. Each matrix has pixels ranging from 0-255. The model uses the matrix's central value as the threshold. This value will be used to define the 8 neighbors' new values. The matrix will then be transformed to binary values, and each value from each position will be concatenated into a new binary value. The binary value is then converted to decimal and assigned to the matrices' center value. As a consequence, a new image is created that better represents the original image's qualities.

Using the image created in the previous phase, we can now divide the image into numerous grids using the Grid X and Grid Y parameters. The settings entered are subsequently saved in the 'Classifier.xml' xml file. This classifier ensures that the larger the radius, the smoother the image will be, but the more spatial information we will be able to obtain. As a result, the classifier required to train the model is now available. The classifier performs and creates a histogram after it is matched with the image and ID. Each histogram will be compared, and the image with the closest histogram will be returned.

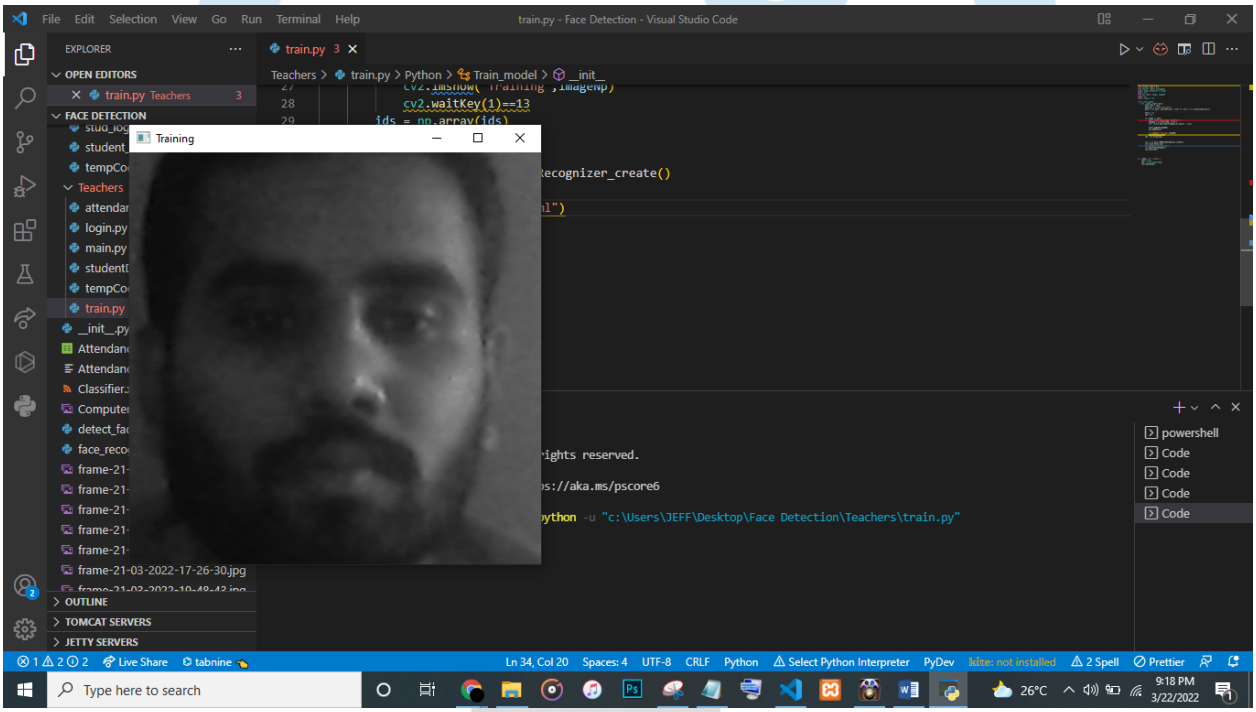

Figure 7: Model is under training using each images stored.

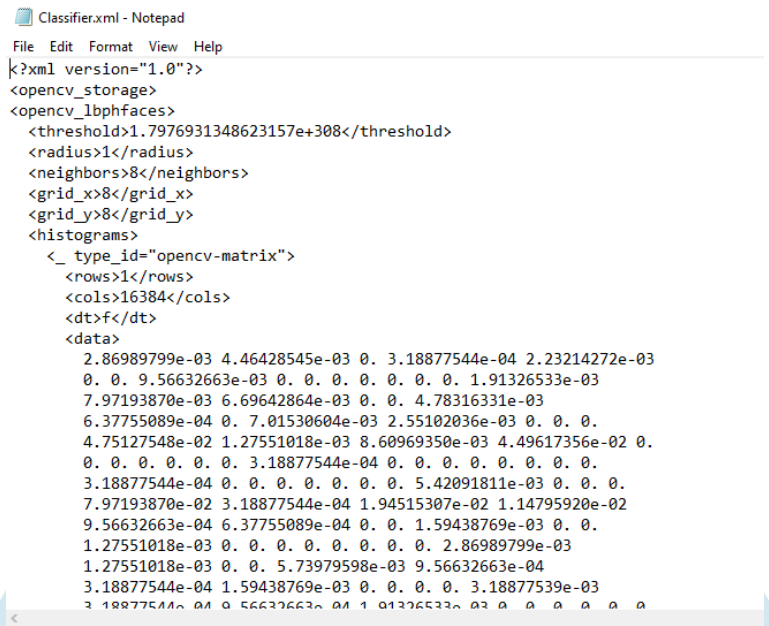

Figure 8: Model is trained and the calculated values are stored in 'Classifier.xml'

C. Identifying the face

Since the model has been trained with the collected data set now the system can easily identify the face by analyzing the unique features of the faces. We must first initialize the Haar Cascade Classifier and read the 'Classifier.xml' file that was created during data training. The system camera must then be initialized using openCV. Once all three have been initialized, we must run it through a function that will recognize specific faces. It's tough to interpret the images captured by the camera because they're so large. As a result, we'll draw a boundary around the face to help the system focus on it. To meet the requirements, photos will first be transformed from RGB to Gray. A square box is drawn around the face using detectMultiScale.

Once the object has been spotted from the image using the classifier, we must make a prediction as to which face this image belongs to. We call this prediction because no system can be certain whether the individual in the image is 'x' 100 %. As a result, making a prediction could assist the mode in learning and training more effectively in order to recognize more accurately. To make a prediction, we'll utilize the prediction function, which takes the grey image as an argument. Once the prediction is complete, we must use a mathematical equation to determine how confident the system is in its prediction (Figure 9)

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

Figure 9: Formula used to calculate histogram to get the closest one

The formula calculates the prediction by comparing the histograms of each image and selecting the one that is the closest to them. It returns the calculated distance, which can be used as a metric of 'confidence.' Lower confidences are better since they suggest the gap between the two histograms is closer, so lower confidences are preferable in this scenario. Then we can use a threshold and the 'confidence' to determine whether the algorithm accurately recognized the image. If the confidence is less than the stated threshold, we can assume that the algorithm has effectively recognized it. We can use this confidence value to create a conditional statement that determines how near the confidence value should be to the face's distance value. In this scenario, my confidence level is set at 80%. Giving a confidence level of 100 percent is a bad idea because there's a chance you'll get nothing in return. If the value is greater than 80%, a letter will appear on top of the face; otherwise, a red rectangle will appear, indicating that no face matches the capture face (Figure 10).

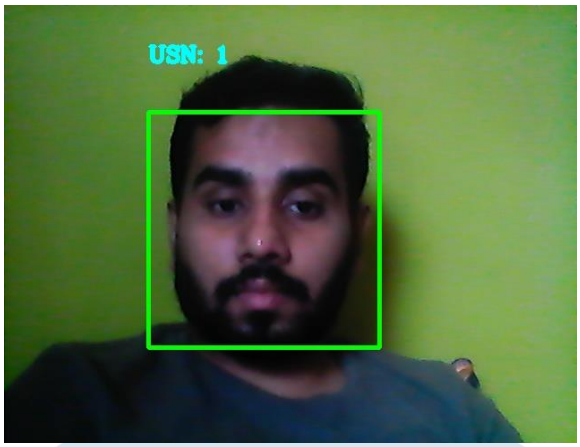

Figure 10: The USN of the student will be highlighted if the face is identified

D. Providing Attendance

The final phase in the process is to give attendance to kids whose faces have been identified. All of the information will be stored in a csv file. We'll extract all of the data from the database and save each column in its own variable. Then, using the append() function, append all variables. Now we'll use the datetime.now() function to collect the date and time of the face that was detected. Once all required details are stored then using writelines() we will insert all details into the csv file (Figure 11).

	A	B	C	D	E	F	G	H
1								
2	1	20MCAR0044	Sheik Bazith	MCA	Third	21/03/202	19:52:55	Present
3								
4								
5								

Figure 11: Details of the student's attendance stored.

RESULT

Face recognition has been used to construct an attendance system. The technology was used to track attendance in a class. If a consistent similar situation (e.g. light, face distance, and facial expression) for image acquisition is maintained, face recognition accuracy is around 95%. The students' survey indicates that the system is functional and efficient, and that it is well-liked by them.

CONCLUSION

At the completion of the project, various faculties will be able to take attendance with the least amount of interaction possible, allowing them to spend more time teaching their students. It has a user-friendly interface for keeping track of all attendance records. This project is really not done yet because it could use a few more features to make it more trustworthy.

REFERENCES

[1] Salama AbdELminaam D, Almansori AM, Taha M, Badr E (2020) A deep facial recognition system using computational intelligent algorithms. PLoS ONE 15(12): e0242269.

[2] Real-Time Face Detection and Tracking Using OpenCVby Prof. P Y Kumbhar, Mohammad Attaullah, ShubhamDhere, Shivkumar, and Hipparagi, 2020.

[3] Face Recognition at a Distance for a Stand-Alone Access Control System by Hansung Lee, So-Hee Park, Jang-Hee Yoo, Se-hoon Jung, and Jun-Ho Huh, 2020.

[4] White D, Dunn JD, Schmid AC, Kemp RI (2015) ErrorRates in Users of Automatic Face Recognition Software.PLoS ONE 10(10): ONE e0139827.

[5] L. Guo & Q.G. Wang (2009), "Research of Face Detection based on Adaboost Algorithm and OpenCV Implementation", J. Harbin University of Sci. and Tech., China, Vol. 14, Pp. 123–126

[6] Sarala A. Dabhade & Mrunal S. Bewoor (2012), "Real-Time Face Detection and Recognition using Haar - based Cascade Classifier and Principal Component Analysis", International Journal of Computer Science and Management Research, Vol. 1, No. 1.

[7] G. Bradski & A. Kaebler (2009), "Learning OpenCV", China: Southeast Univ. Press.

[8] Zhang, Xin & Gonnot, Thomas & Saniie, Jafar. (2017). Real-Time Face Detection and Recognition in Complex Background. Journal of Signal and Information Processing. 8. 99-112. 10.4236/jsip.2017.82007.