

SMART ATTENDANCE

**A Project Report submitted in partial fulfilment of the requirements for the
award of the degree of**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

JYOTHSNA LADE **121710303026**

PERI SRIRAM **121710303038**

ANJANI PUTRA **121710303044**

K.SAI TEJA **121710303063**

Under the esteemed guidance of

Dr. SRINIVAS GORLA

ASSOCIATE PROFESSOR, GIT



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)



DECLARATION

We, hereby declare that the Project review entitled “SMART ATTENDANCE” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 09-05-2021.

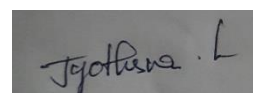
Registration No:

Name:

Signature:

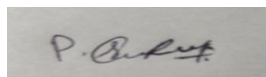
121710303026

Jyothsna Lade



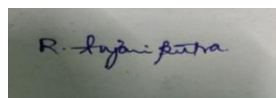
121710303038

Peri Sriram



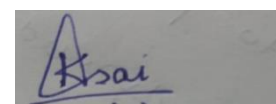
121710303044

R.Anjani Putra



121710303063

K.Sai teja Sreenivas



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM (Deemed to be University)



BONAFIDE CERTIFICATE

This is to certify that the project report entitled “SMART ATTENDANCE” is a Bonafide record of work carried out by **Jyothsna Lade (121710303026)**, **Peri Sriram (121710303038)**, **Anjani Putra (121710303044)**, **K.Sai teja Sreenivas (121710303063)** submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

PROJECT GUIDE

HEAD OF DEPARTMENT

Dr. SRINIVAS GORLA

Dr.R.Sireesha

(ASSOCIATE PROFESSOR)

(PROFESSOR)

CSE, GIT

CSE, GIT

GITAM

GITAM

ACKNOWLEDGEMENT

We would like to thank our project guide **Dr. SRINIVAS GORLA**, Associate Professor, Department of CSE for his stimulating guidance and profuse assistance. We shall always cherish our association for his guidance, encouragement and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under his guidance and constant support

We also express our thanks to the project reviewers **Dr. SRINIVAS GORLA**, Department of CSE, GITAM (Deemed to be University) for his valuable suggestions and guidance for doing our project.

We consider it is a privilege to express our deepest gratitude to **Dr. Sireesha**, Head of the Department, Computer Science and Engineering for his valuable suggestions and constant motivation that greatly helped us to successfully complete this project.

Our sincere thanks to **Dr. C. Dharma Raj**, Principal, GITAM Institute of Technology, GITAM (Deemed to be University) for inspiring us to learn new technologies and tools.

Finally, we deem it a great pleasure to thank one and all that helped us directly and indirectly throughout this project.

JYOTHSNA LADE, 121710303026

PERI SRIRAM, 121710303038

ANJANI PUTRA, 121710303044

KORIKANA. SAI TEJA SREENIVAS, 121710303063

TABLE OF CONTENTS

CONTENTS	PAGE NO.
Declaration	i
Certificate	ii
Acknowledgment	iii
1. Abstract	1
2. Introduction	2
3. Literature Survey	5
4. Problem Identification & Objectives	7
4.1 The Problem	7
4.2 The Proposed Solution	7
5. System Design	8
5.1 Proposed System Architecture	8
5.2 Proposed system Flowchart	8
6. Methodology	10
6.1 Requirements	10
6.2 Face Detection	11
6.3 Face Positioning	14
6.4 Face Encoding	15
6.5 Face Recognition	16
7. Implementation	21
7.1 codes	30
7.2 outputs	32
8. Conclusion	34
9. References	35

LIST OF FIGURES

Figure No.	Description	Page No.
1	Proposed System Architecture	6
2	Proposed system Flowchart	7
3	Face detection	11
4	Face Positioning	13
5	Face Encoding	14
7	Face recognition	15
8	Codes	30
9	Results	32

1. ABSTRACT:

When doing daily tasks, it can be difficult to keep track of attendance. The traditional method of calling each student's name takes a long time, and there's always the possibility of a proxy attendance. To keep track of students' attendance records, the following method relies on face recognition. The administration keeps track of student attendance on a regular basis. As the time for the corresponding subject approaches, the machine begins taking photos, then applies the face detection and thus recognition technique to the given image, resulting in the recognised students being entered into the excel sheet and their attendance being updated with the corresponding time and name of the student. To compute and compare the features, we used a face recognition module.

2. INTRODUCTION:

Attendance is necessary and essential in all institutions in order to determine student success. There are some traditional methods for recording student participation in a class. One method is to call the roll number, but this is a time-consuming procedure that often increases the likelihood of proxy attendance. Finally, a smart attendance system is needed that reduces the manual labour, time, and probability of proxy attendance for professors. This automated system can be created with the help of a facial recognition system and some suitable hardware and software. Attendance is the most significant factor in maintaining student discipline in the classroom, since it is directly related to their academic success.

There are a few popular methods for tracking student participation in class. The first is to call the roll number, and the second is to have students sign a sheet of paper next to their roll number; however, this is a lengthy procedure that often requires proxy participation. To monitor attendance, some companies have implemented biometrics systems such as fingerprint scanners, RFID card readers, and iris scanners. The traditional method of manually calling students' names is a time-consuming event. Each student is assigned a card with their corresponding identification under the RFID card scheme. However, there is a chance of card malfunction or illegal card usage for phoney attendance. Biometrics such as fingerprint, iris, and voice recognition have flaws and are not perfect. Finally, a smart attendance system is needed that reduces the manual labour, time, and probability of proxy attendance for professors.

A facial recognition system, as well as some appropriate hardware and software, can be used to create this automated system. Face recognition is one of image processing's many applications, picture Just a few of the applications include sharpening and reconstruction, medical imaging, remote sensing, transmitting and encoding, machine/robot vision, colour processing, pattern recognition, and video processing.

Image processing is a method of enhancing or extracting essential information from a photograph by adding operations to it. It's a form of signal processing in which a picture is the input and the output is either the image or its attributes/features.

OpenCV library can perform a variety of tasks, including:

1. Images are read and written.

2. Identification of faces and their features.

3. Detection of shapes in an image, such as circles, rectangles, and so on.

Detection of coins in pictures, for example.

4. Recognition of text in pictures.

Reading Number Plates, for example

5. Adjusting the image quality and color palette for example, Instagram and Cam Scanner

Face recognition have two steps:

The 1st is detection of faces, and the 2nd is the identification of those identified face images against a database. Face detection and recognition techniques have been introduced in a variety of ways.

Face identification can be done in two ways: feature-based, which covers geometric features such as the eyes, nose, eye brows, and cheeks, or appearance-based, which covers the entire face.

Face recognition is used in our system to improve upon the existing system's flaws using machine learning, and it requires a high-quality camera to capture the faces of students.

HOG:

The (HOG) is a function descriptor for object detection in computer vision and image processing.

The method counts how many times a gradient orientation appears in a certain region of an image.

This method is similar to edge orientation histograms, scale-invariant aspect transformations descriptors, and shape contexts, but it is more computable.

The HOG descriptor has a few main advantages over other descriptors. Since it acts on local cells, it is invariant to geometric and photometric transformations except for object orientation.

Improvements like these would only be evident in larger geographic areas. Furthermore, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalisation all help to improve local photometric normalisation.

3. LITERATURE SURVEY:

A study was carried out on the different methods for putting in place an efficient attendance tracking system. After performing the survey, we discovered that these methods differ in terms of the types of input methods, data processing methods, and controllers used to run the systems. The new developments have a number of advantages and disadvantages.

"Attendance System Using NFC Technology with Embedded Camera on Mobile Device" is the first system. Near field communication (NFC) is a technology that allows two devices, such as a phone and an NFC tag, to communicate with each other.

The embedded camera will capture the student's face, which will be sent to the college server for verification, and the NFC scanner will read the data from the respective NFC tag.

This system, however, is vulnerable to impersonation, which occurs when one person signs in as another. Many colleges, institutions, and schools use time management systems that use biometrics (fingerprint recognition, RFID, and so on) to identify end users. These technologies, on the other hand, raise new privacy issues. These devices may also be damaged physically by their users. As a result, they would incur higher maintenance costs. The concept we suggest denies someone physical access to the automated device.

An abstract for a paper was written by Maulana Dimas Iffandi, Rangga Nata Adiningrat, and Jeremia Rizki Pandapota.

Normally, attendance is taken by hand. One by one, it can be signed or named. In order to be able to accelerate and save time in this digital age, there must be a break from this absence. Face recognition can be used to keep track of anyone's attendance in an organisation. Face recognition employs a variety of algorithms, including Machine Learning, to dissect and capture photographs of a person's face. This algorithm will allow the system to recognise a person's face and record their attendance, making attendance activities more accurate and efficient.

Abstract of the paper written by Sakshi patel,Ravi Kumar

Every company has a mandatory attendance policy. Keeping a daily attendance log is a complex and time-consuming job. Biometric, RFID, eye tracking, voice recognition, and a variety of other automated methods are all available. This paper outlines a simple and effective method for keeping track of attendance. Face recognition provides an effective method that overcomes ambiguities such as fake participation, high cost, and time consumption, as it is understood that every human's primary identity is their face. For facial recognition and attendance storage, this system employs a face recognizer library.

4. PROBLEM IDENTIFICATION AND OBJECTIVES:

The Problem:

The current system primarily consists of a physical ledger into which the supervisor manually enters all of the students' attendance records. Fingerprint, retina scan, speech recognition, and other innovations have been created to replace this manual. The problem with the present method is that it is time consuming and inaccurate due to proxy attendance. The cost of implementing new technology on a wide scale in any company is prohibitive.

The Proposed Solution:

It automates the time-consuming process of manually tracking attendance records. The administrator enters the student's information and photographs into a database at the start of the semester, and the class list is created automatically. The camera then records the images, compares them to the database, and automatically calculates attendance, which is much more reliable than manual attendance.

Objectives:

Learning the Python programming language and associated image processing libraries.

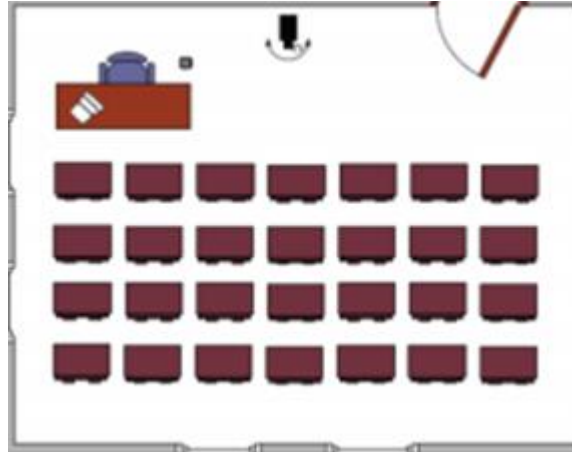
Understanding how the opencv library works.

Creating a database.

The image is compared to the pictures in the database.

5. SYSTEM DESIGN:

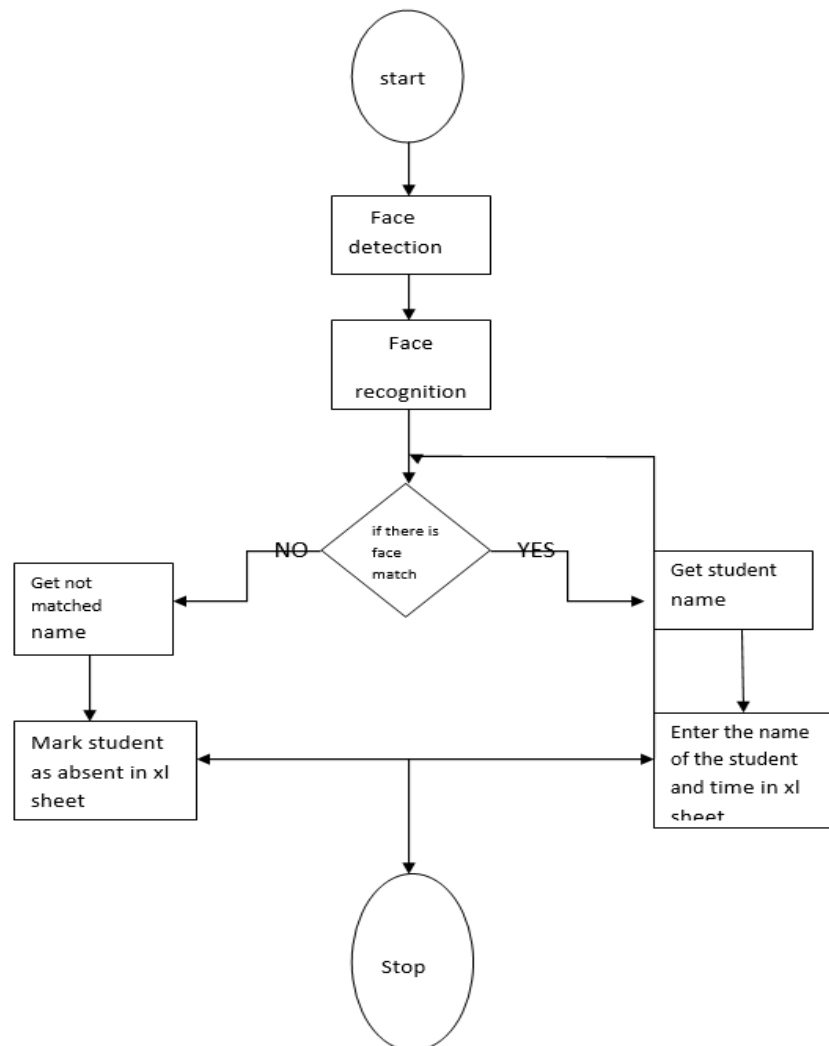
System Architecture Proposed:



This is the blueprint for the attendance management system and how it would be used in a particular class. He would not be considered a student because, as previously said, there is a teacher's desk that faces the students. A camera is mounted in the classroom's centre at a suitable height to catch the entire class up to the last bench. After the students have been seated, the camera will take a picture and start the face detection process using the techniques and methods described in the methodology section. After that, the software would make a folder for the students who needed to be entered into the database. Image recognition is performed using each student's previously placed photographs, which are taken from a database. The photos will be retrieved and connected to any database entries, which will assist us in determining whether or not the student is in class. The software will move on to the next image if no match is found.

Flowchart of the proposed system:

When the professor turns on the machine, the camera captures the image and recognises a face before moving on to the faces that have been identified. It then searches the database for a match, labelling it as present if one exists and as missing if none.



6. METHODOLOGY:

Face recognition is the most critical aspect of smart attendance. Face recognition is being used by a number of tech firms, including Facebook, Amazon, and others. It's mainly used on images that Facebook users post to suggest tagging mates. Amazon has created a device that uses cameras to identify and recognize faces in real time.

SPECIFICATIONS:

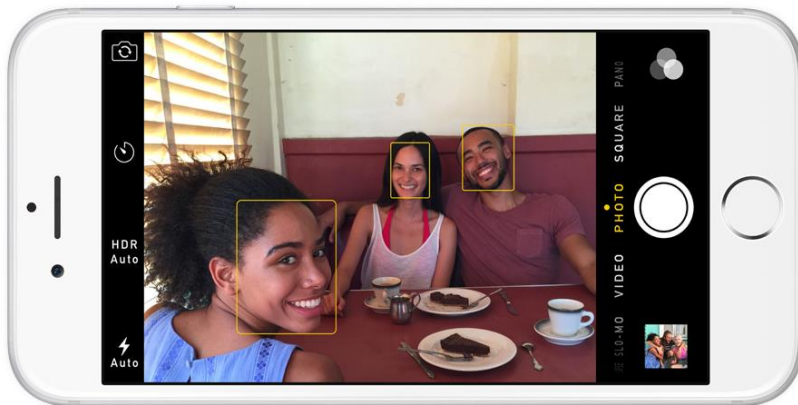
- 1)operating system that is OpenCV and Python compatible like Windows, Linux, MacOS
- 2)Python programming language
- 3)Python-Opencv
- 4)Webcam
- 5)Photo images for testing

We will use OpenCV and Python to apply a face detection algorithm to some images in our project. OpenCV is a free and open-source software library that allows programmers to use routines from computer vision APIs (Application Programming Interfaces). The version we used was called OpenCV-Python, and it was created for Python. The command 'pip install OpenCV-python' can be used to install this on our device.

FACE DETECTION

Face recognition is a form of computer vision that recognises people's faces in digital images. Humans find this easy, but computers need specific instructions. Buildings, vehicles, animals, and other objects that aren't human faces can appear in the photographs. Facial recognition, visualisation, and monitoring are not used in other computer vision applications that include human faces.

We will incorporate a framework for finding faces in digital images in our project. Face recognition and face detection are not the same thing, but they both depend on each other. Face recognition requires face detection in this case in order to "recognise" a face. So we will give the image where the face has to be detected so we will input a picture.

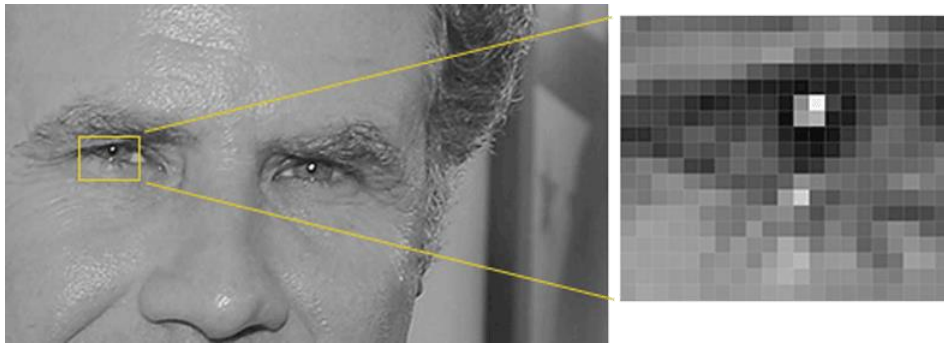


We will take a picture as input and convert it to black and white so we do not need a colour image to find the face in it..



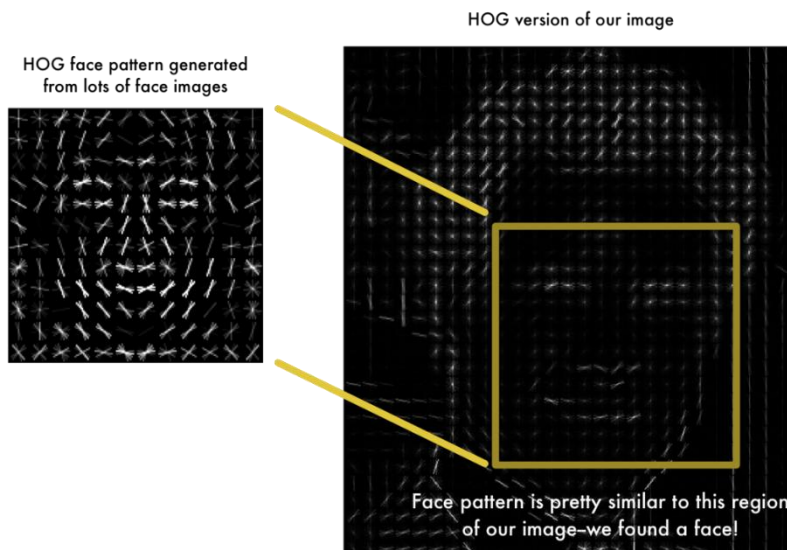
Then we'll go over each and every pixel in our picture one by one.

We're interested in the pixels immediately surrounding each pixel:

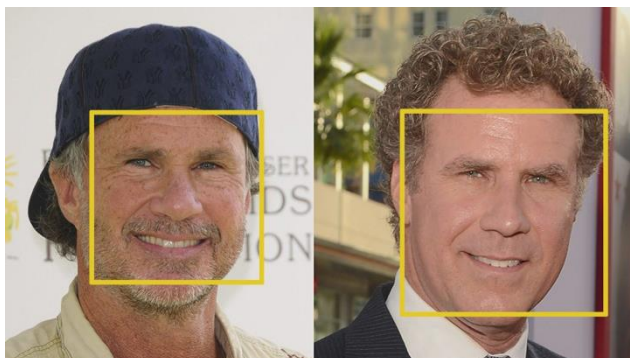


As a result, we've reduced the original image to a very simplistic representation that accurately depicts a face's basic structure.

To locate faces on this HOG photograph, all we should do is search for the a part of the photograph that maximum carefully resembles a recognized HOG sample derived from **a** chain of different education faces:



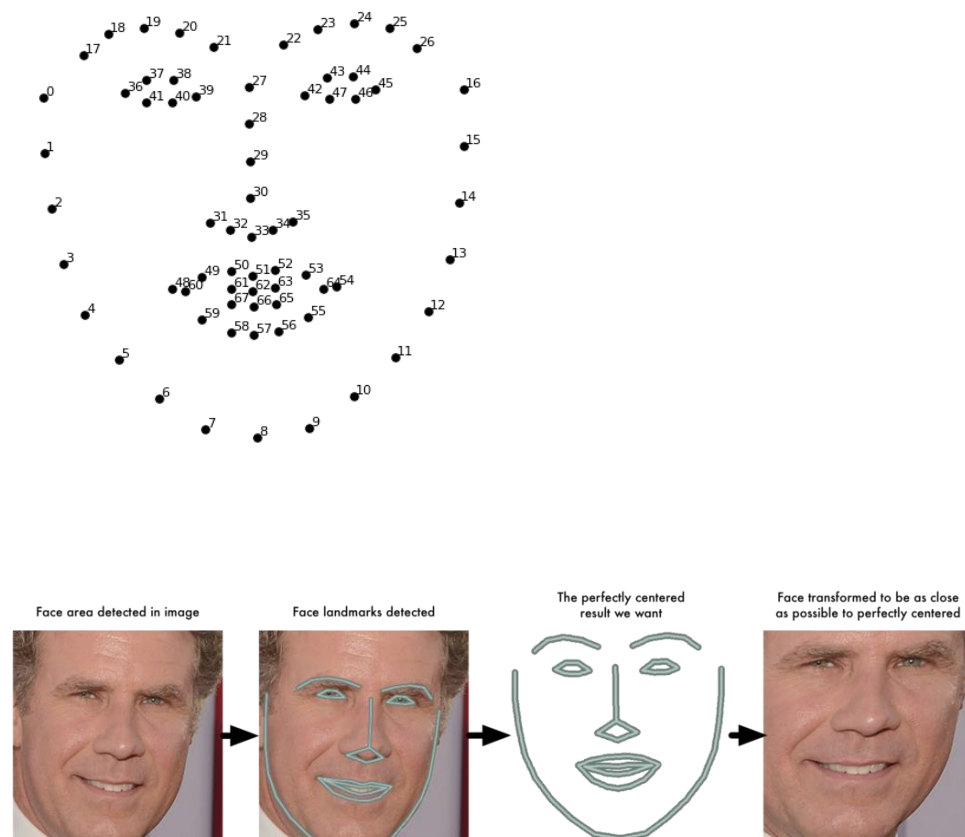
We can now effortlessly discover faces in any photo the usage of this technique:



FACE POSITIONING:

In a human face, there are 68 different points. There are to put it another way, there are 68 face landmarks. This step's primary goal is to locate and position face landmarks. A python script is used to automatically detect face landmarks and position the face as much as possible without distorting the image.

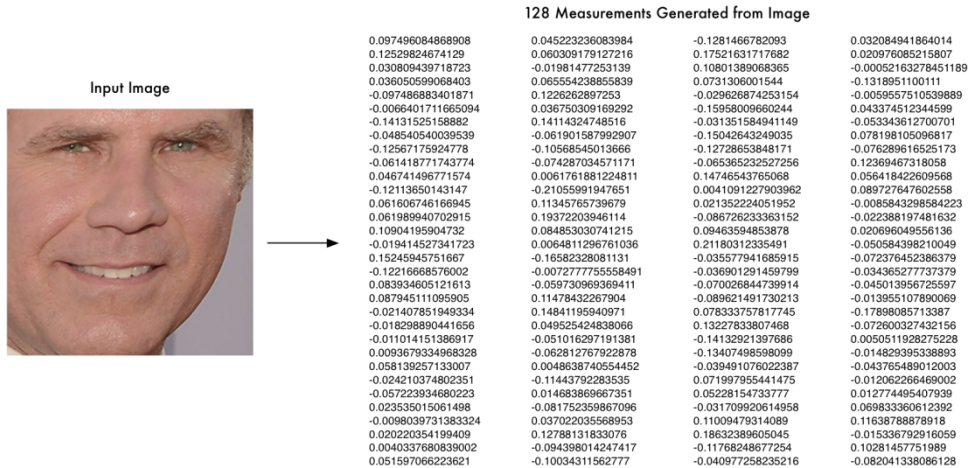
The basic idea is that any face has 68 distinct points (known as landmarks), such as top of chin, outer edge of each eye, inner edge of each forehead, and so on. Then, on any face, we'll use a machine learning algorithm to find these 68 distinct points:



ENCODING OF THE FACE:

The next step after detecting faces in an image is to extract each image's unique identifying facial feature. We extract 128 key facial points for each image of a given input whenever we get a face localization, which are highly accurate.

All we have to do to get the 128 measurements for and face is to put our faces into their pre-trained software.



FACE RECOGNITION:

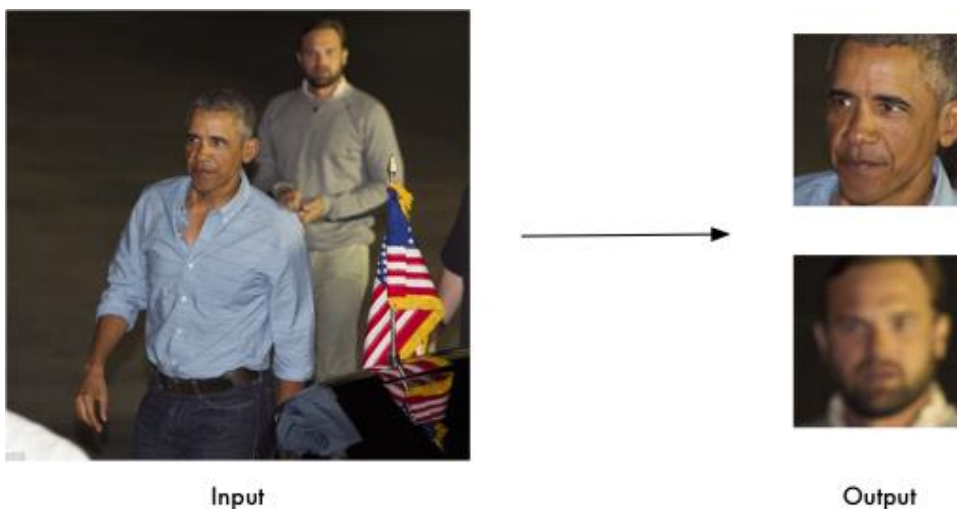
Following the portrayal of faces, the final move is to classify them. We'll need to build a face database for automatic recognition. For each user, multiple photos are taken, and the database is used to retrieve and store their characteristics. Face detection and feature extraction are then performed on an input image, and the features of each face class are compared to the database's stored images.

The compare face's function calculates the distance between a face in a picture and all other faces in the dataset in Euclidean coordinates. In our project, we're using the face recognition module because it's more accurate.

The Face Recognition module includes a simple face recognition command-line tool that allows you to perform face recognition on a folder of images directly from the command line.

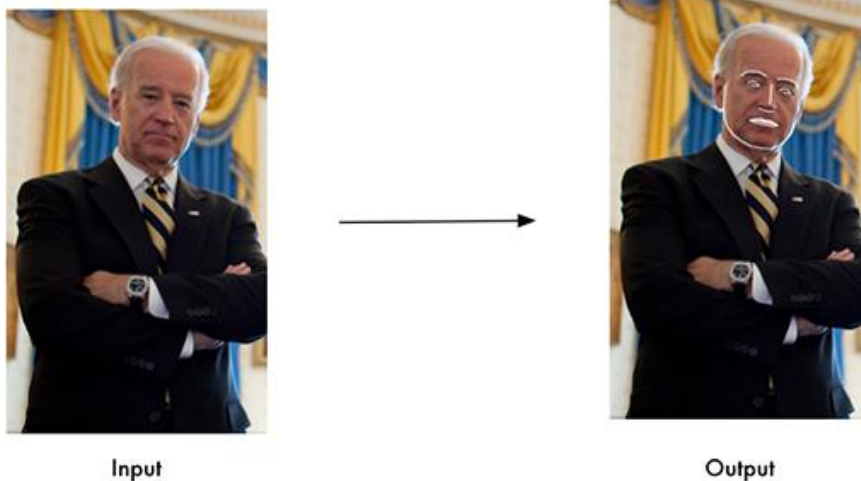
Features:

Find all of the faces in a photograph.



Face features can be found and manipulated in photos.

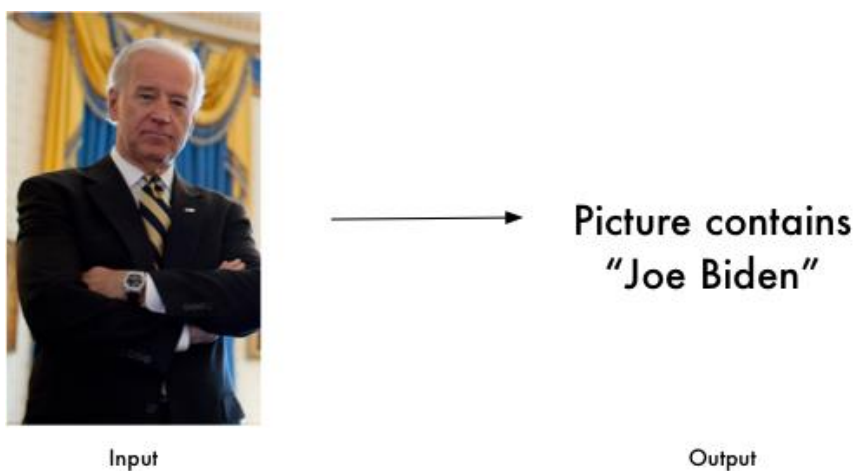
Here in this step, we will get the location of eyes, nose, mouth and chin of the face detected in the image.



Take note of where each person's eyes, nose, mouth, and chin are located, as well as the outlines of their eyes, nose, mouth, and chin.

Finding facial features is extremely useful for a variety of purposes. However, you can use it for a variety of odd and amusing tasks, such as applying digital make-up.

Recognize person name in photographs



Recognize the people in each photograph.

This library can also be used in conjunction with other Python libraries to perform real-time face recognition.

The encoding to find the person's name:

This final move is probably the simplest of the entire procedure. All we have to do now is look through our database of well-known individuals to find individual with the most similar measurements to our test picture.

All we have to do now is to train a classifier to take measurements from a new test image and determine which known individual is the closest match. This takes milliseconds to run. The classifier's output is the person's name.



These are some images of our fellow friends to recognize and compare the faces.

ADJUSTING TOLERANCE OR SENSITIVITY:

If you have several matches for the same person, the people in your pictures can be very similar, and you'll need a lower tolerance value to make face comparisons more stringent. You can do this with the `—tolerance` parameter. The default tolerance value is 0.6, which allows for more stringent face comparisons.

Grid X:

in the horizontal direction, the count of cells. The more cells in the grid and the finer the grid are, the higher the dimensionality of the resulting feature vector. It is normally set to 8.

Grid Y:

The number of cells in a vertical direction. The more cells there are in the grid and the finer the grid is, the higher the dimensionality of the resulting feature vector. It's usually set to eight.

Create a rectangle around the detected face using the `cv.rectangle` feature.

The identified face will be detected and the text will be written on the rectangle after the face in the image has been detected and the rectangle has been drawn.

Definitely `markattendance(name)` function reads the `attendance.csv` file and writes a new line to the csv files containing the name and the time if the name passed as an argument is not present in the attendance file.

Steps that we followed:

1. To make a simpler version of an image, use the HOG algorithm to encrypt it. Find the section of the image that most closely resembles a generic HOG encoding of a face using this condensed version of the image.
2. You will work out the pose by looking for main landmarks in the face. Warp the image using those landmarks such that the eyes and mouth are in the middle.
3. Create a network that understands how to calculate facial features using a centred face image. Keeping track of the 128 measurements is important.
4. Examine all of the faces we've already weighed to see whose measurements are the closest to ours. That's all there is to it; we've found our soul mate.

7. IMPLEMENTATION:

import cv2

OpenCV is the large open-source computer vision, machine learning, and image processing library

import numpy as np

NumPy, or Numerical Python is a Python library that allows you to manipulate arrays. We have lists in Python that serve as arrays, but they are slow to process. NumPy aims to have a 50-fold faster array object than standard Python lists. NumPy arrays, unlike lists, Processes can access and modify them easily since they are stored in a single continuous location in memory. In computer science, this is addressed to as locality of reference. This is the primary explanation why NumPy outperforms lists. It's also been tweaked to work with the most recent CPU architectures

path = "training_images"

Need to give the path of the folder where the training images were stored **images = []**

image name = []

create two empty lists which will store the read images and name of the image **my list = os.listdir(path)**

The os module's listdir() function returns a list containing the names of the entries in the directory specified by route.

curImg = cv2.imread("path to the image")

The method is used to retrieve an image from a register. If the image can't be read, this method returns an empty matrix (due to a missing file, insufficient permissions, or an unsupported or invalid format).

Syntax: cv2.imread(path, flag)

path: The path of an image to be read is represented as a string.

flag: It defines how an image should be interpreted. `cv2.IMREAD_COLOR` is the default value.

Return Value: The image loaded from the specified file is written by this process.

Note: The image should either be in the working directory or have its full path specified.

All three types of flags are described below:

cv2.IMREAD_COLOR: It tells the computer to load a color image. Any picture accountability will be ignored. It's the default setting. We may also transfer the integer value 1 for this flag.

cv2.IMREAD_GRAYSCALE: It specifies that an image should be loaded in grayscale mode. We may also use the integer value 0 for this flag.

cv2.IMREAD_UNCHANGED: It specifies that an image with an alpha channel be loaded. Alternatively, we can set this flag to the integer value -1.

`images.append(curImg)`

appends the read images to the images list

`imagename.append(os.path.splitext(cl)) [0]`

cl indicates a specific element in the mylist, which contains the names of the elements present in the training images folder

`def findencodingd(images):`

The images are encoded and stored in a list by this function.

images : list of images which are used for training

Returns: returns a list which contains encoded information of training images

`img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`

To change an image from a color space to other, use the `cv2.cvtColor()` process. There are over 150 color-space conversion methods in OpenCV. The first parameter is the image whose color space must be modified, and the second parameter is the color space to which it must be changed.

Example: When using the OpenCV function imread() to read an image file, the color order is BGR (blue, green, red)

cv2.COLOR_BGR2RGB: converts the order of colours from bgr to rgb (red green blue)

cv2.COLOR_BGR2GRAY: converts the coloured image to gray

cv2.COLOR_BGR2HSV: converts the bgr image to hsv image code

encodeList.append(encode)

appends the 128-dimensional face encoded values of every image in the images list to the encodeList list and return encodeList

def markAttendance(name):

The function will read attendance.csv file and if the name which was passed as an argument is not present in the attendance file, then it writes a new line to the csv files which contains the name and the time

with open ('Attendance.csv', 'r+') as f:

r+ indicates we were opening the attendance file with read and write permissions

myDataList = f.readlines()

It reads the csv file and each row of the csv files is stored as a string element in the myDataList

list nameList = []

for line in myDataList:

entry =line.split(',')

As each row of the attendance.csv file is stored a concatenated string in the myDataList list now we were splitting the concatenated string into separates strings considering ',' as the separator, and the entry is a list whose first element is the name and second element is the time at which the person has entered the class

nameList.append(entry [0])

we were appending the first element of the entry list I.e., name to the nameList list.

if name not in nameList:

If the name which is passed as an argument is not present in the nameList then the following lines of code is executed.

now = datetime.now()

The current local date and time is returned by the now () function, which is defined in the datetime module.

Syntax: datetime.now(tz)

Parameters:

tz : The current time and date must be in the specified time zone.

Returns: The current date and time in time format is returned. The return will be in the below format 2018-03-29 10:26:23.473031

dtstring = now.strftime("%H: %M")

The strftime() method accepts one or more format codes as an argument and returns a formatted string. The letter H stands for the hour, and the letter M stands for the minute.

writelines(f'\n{name}, {dtstring}')

writes a new line to the attendance.csv file with first element as the name and second element as the time string at which the face was recognised

encodeListKnown = findEncodings(images)

It calls the find Encoding's function and converts the images to a 128 dimensional encode value

cap = cv2.VideoCapture(0)

A Video Capture object is required to capture a video. The system index or the name of a video file may be used as an argument. The device index is simply a number that identifies which camera is being used. As a result, I simply move 0 (or -1). By passing 1 and so on, you can pick the second camera. But don't forget to release the catch at the end.

while True: success,

img = cap.read()

cap.read() returns a bool (True/False) and the frame it reads. If frame is correctly read, it will return True.

imgS = cv2.resize(img, (0,0), None, 0.25, 0.25)

The captured image is being resized using the cv2.resize function.

imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

we are converting the captured image color space from the bgr to rgb and the description about the cvtColor function is described above

facesCurFrame=face_recognition.face_locations(imgS)

face_recognition.face_locations(img,number_of_times_to_upsample=1,model='hog')

Returns an array of bounding boxes of human faces in an image

Parameters:

img : An image (as a numpy array)

Number-of-times-to-upsample : The no. of times should the picture be upsampled in search of faces? Smaller faces are seen in higher numbers.

model : Which face recognition model should I use? On CPUs, "hog" is less precise but quicker. "cnn" is a GPU/CUDA accelerated deep-learning model that is more accurate (if available). The default value is "hog."

Returns: A list of tuples of found face locations in css (top, right, bottom, left) order

encodeCurFrame= face_recognition.face_encodings(imgS, facesCurFrame) 1

The above line of code encodes the captured image to a 128-dimensional encode value, with the image as the first argument and the second argument facing the camera. CurFrame is well-known for its facelocations (if you already know the bounding boxes of each face)

for encodeFace, faceLoc in zip(encodeCurFrame, facesCurFrame):

for each set of encode value and face_location in the encodeCurFrame and facesCurFrame the below lines code gets executed.

matches=face_recognition.compare_faces(encodeListKnown,encodeFace)

face_recognition.compare_faces(known_face_encodings,face_encoding_to_check, tolerance=0.6)

Compare a list of face encodings to a candidate encoding to see whether they match.

Parameters:

known_face_encodings:A list of known face encodings.

face encoding to check: A single face encoding to test against the list tolerance – the minimum distance between the faces to consider a match. The lower the amount, the more stringent the policy. The highest score on average is 0.6.

Returns: A list of True/False values indicating which known_face_encodings match the face encoding to check

faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)

face_recognition.face_distance(face_encodings, face_to_compare)

Calculate the euclidean distance between each reference face by comparing a list of face encodings to a known face encoding. The distance between the two faces shows their proximity.

Parameters:

face_encodings : List of face encodings to compare

face_to_compare :A face encoding to compare against

Returns: The distances for each face are stored in the same order as the 'faces' list in a numpy ndarray.

matchIndex = np.argmin(faceDis)

The numpy.argmin() method returns indices of the min element of the array in a particular axis.

if matches[matchIndex]:

The `matchIndex` is the index of training image which has most similarity to the captured frame and `matches` is a list of true or false based on the tolerance value. If the `matches[matchIndex]` is true then below lines of code will be executed **`name = classNames[matchIndex].upper()`**

The `classNames` is a list of names and `classNames[matchIndex]` returns the name at the index and the above code converts it into upper case letters

```
y1, x2, y2, x1=faceLoc
```

```
y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
```

```
cv2.rectangle(img, (x1,y1), (x2,y2), (0,255,0),2)
```

`cv2.rectangle()` method is used to draw a rectangle on any image.

Syntax: `cv2.rectangle(image, start_point, end_point, color, thickness)`

Parameters:

image: It is the image on which rectangle is to be drawn.

start_point: It is the rectangle's starting coordinates. The coordinates are expressed as two-valued tuples, i.e. (X coordinate value, Y coordinate value).

end_point: It's the rectangle's final coordinates. The coordinates are expressed as two-valued tuples, i.e. (X coordinate value, Y coordinate value).

color: It is the color of the rectangle's boundary line to be drawn. We move a tuple to BGR. For example, (255, 0, 0) is a blue color.

thickness: The width of the rectangle boundary line in pixels. The defined color will fill the rectangle form with a thickness of -1 px.

Return Value: It returns an image.

```
cv2.rectangle(img, (x1,y2 - 35), (x2,y2), (0,255,0), cv2.FILLED)
```

Draws another rectangle with just below the above one which is used to put the test(name).

```
cv2.putText(img, (x1+6, y2-6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
```

`cv2.putText()` method is used to draw a text string on any image.

Syntax:

Syntax: `cv2.putText(image, text, org, font, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]])`

Parameters:

image: It is the picture that will be used to write text on.

text: text string is drawn.

org: It's the image's coordinates for the bottom-left corner of the text string. The coordinates are expressed as two-valued tuples.

font: It specifies the type of font. FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN, and others are examples of font styles.

Font Scale: Font scale factor multiplied by the font-specific base size.

color: It's the color of the text string that's going to be drawn. We move a tuple to BGR. For example, (255, 0, 0) is a blue color.

thickness: The line having thickness in px.

Line Type: It is a parameter that specifies the line type to be used. This is an optional parameter.

bottomLeftOrigin: This is an optional parameter. The picture data origin is at the bottom-left corner when this is accurate. If not, it's in the top-left corner.

Return Value: It returns an image.

markAttendance(name)

Calls the markAttendance function and marks the attendance for the recognised person

cv2.imshow('Webcam', img)

cv2.imshow() method is used to display an image in a window. The window automatically fits to the image size.

Syntax: `cv2.imshow(window_name, image)`

Parameters:

window name: As a string, the name of the window in which the image will be shown.

image: It is the picture that will be seen.

Return Value: It doesn't return anything.

cv2.waitKey(1)

This function is critical; without it, `cv2.imshow()` will not function properly.

`cv2.waitKey` (wait time in milliseconds) If the wait time is set to 6000, the image will be shown for 6 seconds before being closed (assuming the script includes `cv2.destroyAllWindows()`). If the parameter is set to '0,' the picture will be shown indefinitely before the esc key is pressed.

CODE:

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime

# from PIL import ImageGrab

path = (r'C:\Users\jyothisna\Desktop\one_shot_Learning\Training Images')
images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])
print(classNames)

def findEncodings(images):
    encodeList = []

    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList

def markAttendance(name):
    with open('Attendance.csv', 'r+') as f:
        myDataList = f.readlines()

        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
        if name not in nameList:
            now = datetime.now()
            dtString = now.strftime('%H:%M')
            f.writelines(f'\n{name},{dtString}')

##### FOR CAPTURING SCREEN RATHER THAN WEBCAM
# def captureScreen(bbox=(300,300,690+300,530+300)):
#     capScr = np.array(ImageGrab.grab(bbox))
#     capScr = cv2.cvtColor(capScr, cv2.COLOR_RGB2BGR)
#     return capScr

encodeListKnown = findEncodings(images)
print('Encoding Complete')

cap = cv2.VideoCapture(0)

while True:
    success, img = cap.read()
    # img = captureScreen()
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
```

```

for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
    matches = face_recognition.compare_faces(encodelistKnown, encodeFace, tolerance=0.5)
    faceDis = face_recognition.face_distance(encodelistKnown, encodeFace)

# print(faceDis)
    matchIndex = np.argmin(faceDis)

    if matches[matchIndex]:
        name = classNames[matchIndex].upper()

# print(name)
    y1, x2, y2, x1 = faceLoc
    y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
    cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
    cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
    markAttendance(name)

#acc = accuracy_score(encodelistKnown[matchIndex], encodeFace)
# print(acc)

cv2.imshow('Webcam',img)
cv2.waitKey(0)

```

OUTPUTS:

```
Console 1/A X

In [2]:

In [1]: runfile('C:/Users/jyothsna/Desktop/one_shot_Learning/main.py', wdir='C:/Users/
jyothsna/Desktop/one_shot_Learning')
['bhargavi.jpg', 'charith.jpeg', 'Jyothsna.jpg', 'manasa.JPG', 'manojna.jpg',
'nitin.jpeg', 'priyanka.jpeg', 'raasi.jpg', 'rohit.jpeg', 'sahithi.jpg',
'sannihitha.jpeg', 'sathwik.jpeg', 'Sriram.jpg', 'subash.jpg']
['bhargavi', 'charith', 'Jyothsna', 'manasa', 'manojna', 'nitin', 'priyanka', 'raasi',
'rohit', 'sahithi', 'sannihitha', 'sathwik', 'Sriram', 'subash']
Encoding Complete
```



Webcam



File Home Insert Page Layout Formulas Data Review View Help								
Paste		Calibri 11		B I U		Font		Alignment
Clipboard		A1		X ✓ fx		Name		
	A	B	C	D	E	F	G	H
1	Name	Time						
2								
3	JYOTHSNA	17:51						
4	MANASA	17:51						
5	NITHIN	17:52						
6	SANNIHITI	17:52						
7	MANOJNA	17:53						
8	SAHITHI	17:53						
9	BHARGAV	17:54						
10	SUBASH	17:55						
11	CHARITH	17:55						
12	PRIYANKA	17:56						
13	RAASI	17:56						

8. CONCLUSION:

Our team chose the Smart Attendance Management System project because, while the world is moving toward artificial intelligence, but the process of taking attendance is still the same old form., which is boring and allows for proxy attendance. Since a student's attendance is directly related to his or her experience, he or she gasps and fails academically. As a result, students will benefit from using this smart attendance management approach to prevent proxy attendance, and proxy attendance can be avoided by using this smart attendance management approach. because there would be less room for proxy attendance.

9. REFERENCES:

- A Feature dependent approach to face recognitionProc,” by ManjunathB S, ChellappaR, and C vonder Malsburg, was published in 1992. IEEE Conference on Computer Science 373-378 Computer Vision and Pattern Recognition
- In 1981, Baron R J published a paper titled “Mechanism of Human Facial Recognition” in the International Journal of Man-Machine Studies, Volume 15, Numbers 137-178.
- Face recognition using convolutional neural networks and Gabor filters," says Bogdan Kwolek. Artificial Neural Networks: An International Conference 2005, Springer Berlin Heidelberg.
- An Efficient Attendance System Using Local Binary Pattern and Local Directional Pattern," by C. Ashwini and colleagues. www. jncet. org Journal of Network Communications and Emerging Technologies (JNCET) 8.4 (2018).
- Oscar Karnalim, "Face-to-Face in the Classroom: Dataset and Discovery." The 8th International Conference on Image Processing Theory, Tools, and Applications that took place in (IPTA). 2018 IEEE.
- Shubhobrata Bhattacharya "A Face Recognition Based Attendance System for Classroom Environment." "Smart Attendance Monitoring System (SAMS): " The 18th IEEE International Conference on Advanced Learning Technologies will be held in 2018. (ICALT). 2018 IEEE.
- Face recognition-based smartphone automated classroom attendance management system," by Samet, Refik, and Muhammed Tanriverdi. International Conference on Cyberworlds (Cyberworlds 2017) (CW). 2017 IEEE.