As per UGC guidelines an electronic bar code is provided to secure your paper

International Journal for Modern Trends in Science and Technology
Volume 10, Issue 06, pages 01-04.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol10issue06.html
DOI: https://doi.org/10.46501/IJMTST1006001

Facial Recognition Based Attendance System with Cloud-Integrated Data Management

Anita Joshi[1] | Pallav Vaniya[2] | Vansh Bhatt[2] | Ved Thorat[2] | Vedant Patil[2] | Vedika Mali[2] | Abhishek Veer[2]

[1]Professor, Vishwakarma Institute of Technology, Upper Indiranagar, Bibwewadi, Pune, Maharashtra, INDIA
[2]Vishwakarma Institute of Technology, Upper Indiranagar, Bibwewadi, Pune, Maharashtra, INDIA

To Cite this Article
Anita Joshi, Pallav Vaniya, Vansh Bhatt, Ved Thorat, Vedant Patil, Vedika Mali and Abhishek Veer, Facial Recognition Based Attendance System with Cloud-Integrated Data Management, International Journal for Modern Trends in Science and Technology, 2024, 10(06), pages. 01-04. https://doi.org/10.46501/IJMTST1006001

Article Info
Received: 11 May 2024; Accepted: 01 June 2024; Published: 06 June 2024.

ABSTRACT

Managing attendance at a place where number of people are more is a challenging task, and likely there are few percent chances of error always there. To solve this problem in this paper we have proposed our Face-recognition based attendance system with cloud integrated data management. Another important thing that is having the record of attendance we have used cloud database to store and record attendance. The database is automatically updated by system developed by PyMongo, so that later that can be also displayed to user via their login onto a dashboard. In this project we have built face recognizer with OpenCV python and Eigenfaces algorithm which uses Principal Component Analysis (PCA) and used MongoDB Atlas, a service of Mongo for hosting the cloud database. We will be using MERN stack for building dashboard on which one can monitor their attendance. Using OpenCV against MATLAB also helps in saving memory and gives higher speed of execution.

KEYWORDS: Face-Recognition, OpenCV python, Eigenfaces, MERN stack, PCA, MongoDB Atlas, Rest-API.

1.INTRODUCTION

Attendance is important aspect when it comes to any type of institution. The traditional way of manually noting down attendance is very time consuming, and it is an extra burden on facultie and there is no chance that the record will be maintained for several years, and if maintained it will be very hard to find a particular person's attendance between the bundles of papers. A facial-recognition based attendance system is a computerised biometric system which Identifies facial feature of a person on the given training dataset and when an image or real-time video frame is passed it first detect the face, then tries to recognises the person based on the facial features extracted from the training dataset. When a face is recognized it searches for the name in database by passing it as a query, to record the attendance of person with date and time. Now the person can login to the dashboard with their credentials (e.g. email, registration Number, etc) and their attendance can be fetched from database which will return an object having the date and time of when the face was recognised by the system, and it can be displayed to user with interface. This system can also be used in places such as educational institutes to monitor

attendance of students, it can be used in corporate offices, at places where only authorised persons are allowed or at such places where person gets paid based on their presence, such as labour work, etc.
In paper[1], authors have compared different tools like OpenCv and Matlab, and different algorithms such as Local Binary Histogram Pattern(LBPH), Eigenfaces, Fisherfaces. In paper[6],the author describes an algorithm for detecting human faces and subsequently locating the eyes, nose, and mouth. Symmetry based functions to locate the center of the eyes, tip of nose, and center of mouth within the facial segmentation mask.
In this paper we have researched on scalability and expandability of this type of attendance system, we have tried to make it scalable by using cloud Integrations and storing data on cloud. Since MongoDB is scalable horizontally it will not have problem if very large amount of data is present on system.

2. TOOLS AND TECHNOLOGY

1. OpenCV with Python: Open Cv is an open sources widely used library for computer vision, having many inbilit functions, such as RGB image, Grayscale image, Edge detection, and many more. It also provides some built-in facerecognizers, such as Eigenfaces which we are going to use in this project.

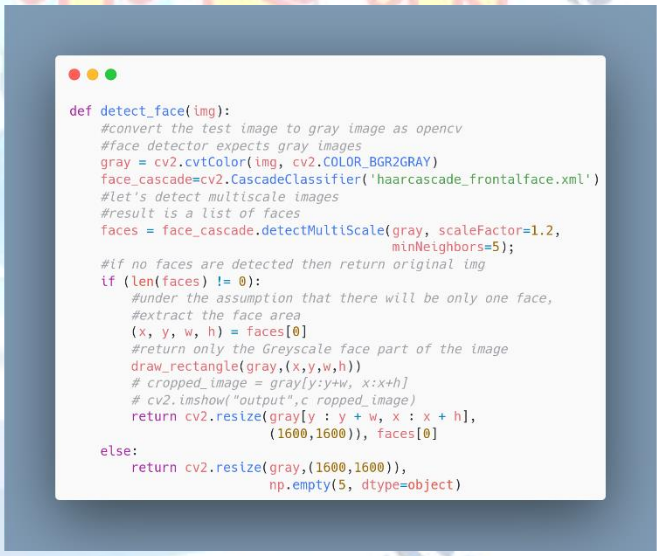
2.EigenFaces Algorithm: There are many algorithms present in openCv such as Local Binary Histogram Pattern(LBPH), Fisherfaces, Haar Cascade, but the reason for choosing Eigenfaces was because mainly it uses Principal Component Analysys(PCA),also it is consume less resources, and overall gives an decent accuracy.
How it works is it reduces the unnecessary factors or these so called “Dimensions” of the image. This will provide us with an image which has a lower “dimension”. The advantage of lowering the dimension is that it just reduces the unnecessary information and just focuses on the face and makes it easier to classify the image that is, to find who the face belongs to. So, in simpler terms we're simplifying the image to make it easier to recognize faces.

2. Mongo DB Atlas
MongoDB Atlas is a service ran by Mongo, which is for hosting mongodb databases on cloud infrastructure. It is a non-relational document database that provides support for JSON-like storge.
In this we have integrated with the free tier of MongoDB atlas which comes with a shared host.

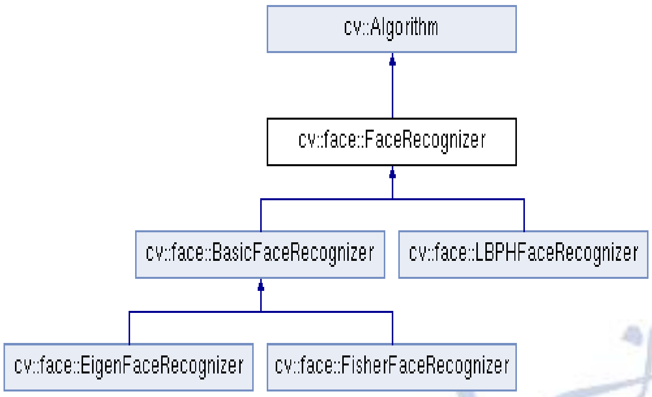
3. METHODOLOGY/EXPERIMENTAL

The first step would be collecting the data for training the algorithm, this can be done by clicking multiple photos of a person using a python script or can be done manually. We have set a minimum of 30 photos per person. The pictures then need to be passed by a pre-processing python script to which will detect the face in image with some pre trained algorithm, then crop accordingly, and make it grey scale to save some computational power when training the face recognition model.

The Fig. 1 code snippet shows the script for pre-processing, which takes image as an argument and returns another cropped greyscale image with detected face.

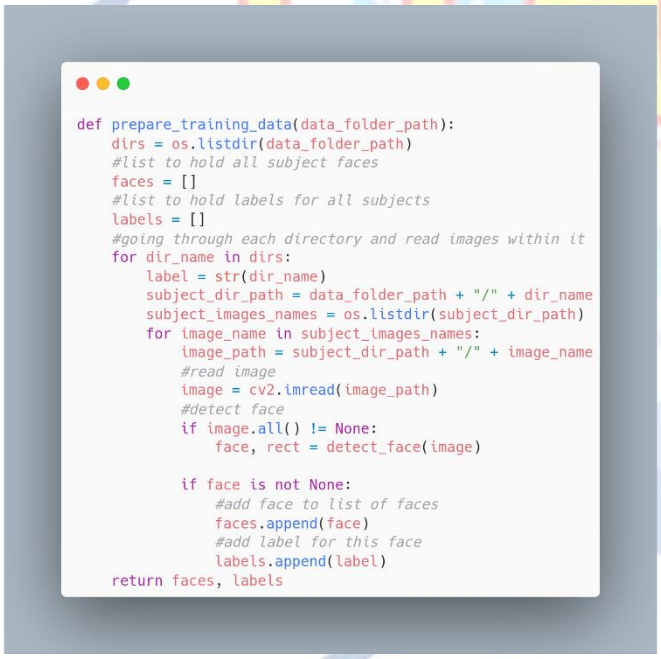

```
def detect_face(img):
    #convert the test image to gray image as opencv
    #face detector expects gray images
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade=cv2.CascadeClassifier('haarcascade_frontalface.xml')
    #let's detect multiscale images
    #result is a list of faces
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2,
                                          minNeighbors=5);
    #if no faces are detected then return original img
    if (len(faces) != 0):
        #under the assumption that there will be only one face,
        #extract the face area
        (x, y, w, h) = faces[0]
        #return only the Greyscale face part of the image
        draw_rectangle(gray,(x,y,w,h))
        # cropped_image = gray[y:y+w, x:x+h]
        # cv2.imshow("output",c ropped_image)
        return cv2.resize(gray[y : y + w, x : x + h],
                          (1600,1600)), faces[0]
    else:
        return cv2.resize(gray,(1600,1600)),
                          np.empty(5, dtype=object)
```
FIG. 1

The next step is to build face recognizer using OpenCV with the help of FaceRecognizer class present in it, and feed the recognizer the pre-processed image to train the model.


cv::Algorithm
cv::face::FaceRecognizer
cv::face::BasicFaceRecognizer
cv::face::LBPHFaceRecognizer
cv::face::EigenFaceRecognizer
cv::face::FisherFaceRecognizer
Fig.2

The Fig. 2 shows Inheritance diagram for OpenCV FaceRecognizer class.

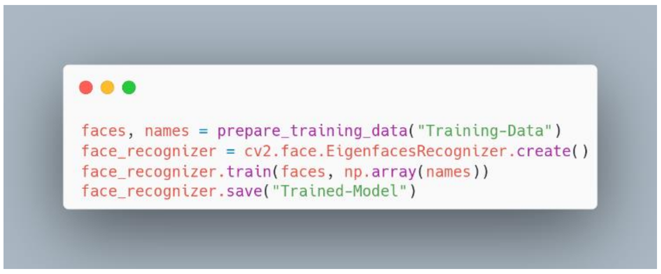
The Fig. 3 code snippet, shows how to clean data for finally feeding it to model, the above function takes the directory of images (which is not pre-processed), pre-processes it and return 2 arrays one is faces which contain face vector and another one is names which contain names of each person.



```
def prepare_training_data(data_folder_path):
    dirs = os.listdir(data_folder_path)
    #list to hold all subject faces
    faces = []
    #list to hold labels for all subjects
    labels = []
    #going through each directory and read images within it
    for dir_name in dirs:
        label = str(dir_name)
        subject_dir_path = data_folder_path + "/" + dir_name
        subject_images_names = os.listdir(subject_dir_path)
        for image_name in subject_images_names:
            image_path = subject_dir_path + "/" + image_name
            #read image
            image = cv2.imread(image_path)
            #detect face
            if image.all() != None:
                face, rect = detect_face(image)

            if face is not None:
                #add face to list of faces
                faces.append(face)
                #add label for this face
                labels.append(label)
    return faces, labels
```

Fig. 3

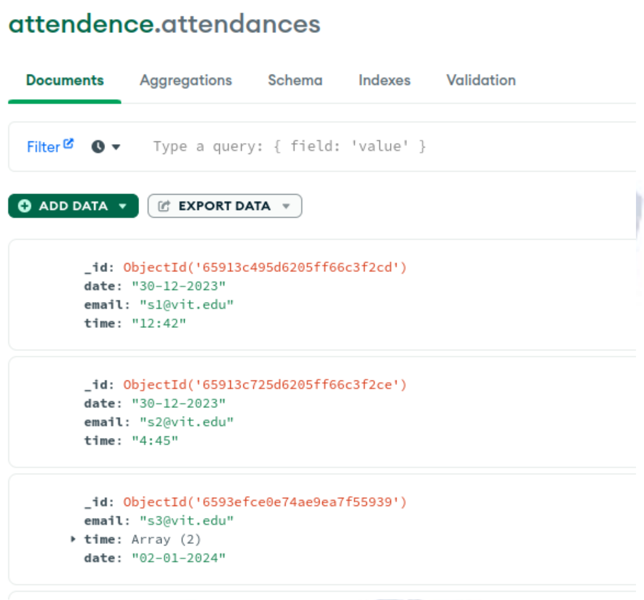


```
faces, names = prepare_training_data("Training-Data")
face_recognizer = cv2.face.EigenfacesRecognizer.create()
face_recognizer.train(faces, np.array(names))
face_recognizer.save("Trained-Model")
```

Fig. 4

In Fig. 4 code snippet, we have trained the face recognizer with the face vectors and names array.

Now To predict the face we give it a image, again we pre-process it and give it to our model, and it returns the name of the person and confidence which is a number . If the confidence is higher, then it means that the pictures are less matching, or in other words the lower, the better. Now when we have predicted the face, we connect with our mongodb database, using Pymongo library, which is built to integrate mongo in python easily.



```
Type a query: { field: 'value' }

_id: ObjectId('65913c495d6205ff66c3f2cd')
date: "30-12-2023"
email: "s1@vit.edu"
time: "12:42"

_id: ObjectId('65913c725d6205ff66c3f2ce')
date: "30-12-2023"
email: "s2@vit.edu"
time: "4:45"

_id: ObjectId('6593efce0e74ae9ea7f55939')
email: "s3@vit.edu"
time: Array (2)
date: "02-01-2024"
```

Fig. 5

The Fig. 5 is an example of assuming a student s1, whose attendance has been marked on 29th of October at 10:28 am and 2:32 pm.In this way as the student is recognised by system it can append the date and time here, and later it can be accessed by system admin, and then it can be filtered according to date and time as per requirements. Now for the user, they can go to the dashboard and enter their email, and it will request the server which will request data from database, here we use Mongoose library to interact with MongoDB, and fetch it and show it to user. To develop dashboard, we are using the MERN tech stack, which is MongoDB, Express, React and NodeJs. There can also be an admin login, where the system admin can check attendance of every person, admin can also do this by directly connecting to the database with MongoDB Compas, another service ran my mongo.

4. RESULTS AND DISCUSSIONS

It's a very effective system, giving about 95 to 96% accuracy based on the testing, this type of system should be adapted at every places for monitoring and keeping track of attendance. Also this data can be further analysed and can be used to pull out insights.

5. MATH

The Eigenfaces Algorithm uses the following steps to first convert all training images into a vector and then take a mean of it, and

1. Collecting the face images (in the form of array) I = [I1, I2, I3, ….,Im] for training the algorithm and must be centered with the same size (N, N).
2. Represent the obtained image Ii as a vector Γi , Γ = [Γ1, Γ2, …., Γm]. dim (Ii) = (N, N) and dim (Γi) = (m, N*N)
3. Compute the average face vector: Ψ = (1/m) * Σ_Γi i.e. compute all images and calculate the grayscale value of all images and make it average or mean images.
4. Subtract the mean face from every vector: φ = [Γ1 - Ψ, Γ2 - Ψ, …,Γm - Ψ] dim(φ) = (m, N*N)
5. Form the covariance matrix C: C = (1/m) * $\varphi\varphi$T where φT is the transpose of φ. dim(C) = (m, m)
6. Compute the eigenvectors(eigenfaces) U of C. dim (U) = (K, N*N)
7. Finding the vector of wights Wi for each image vector φi in the database by using this formula:

φi = Σ Wij * Uj (j =1, 2, …, K)

dim (Wi) = (1, K)

WT = [W1, W2, …, Wm]

dim(W) = (K, m)

6. FUTURE SCOPE

This system is very effective and due to the cloud integration it is now become more easier to access data from any where from any device. Since we are using MongoDB it can be easily horizontally scalable and can store a large amount of data. Also improvement can be done on the ML Model by making it more optimized and adding few more detection steps such as fakeness detection for more reliable results.

7. CONCLUSION

It is a very modern, fast, effective way to keep track record of attendance, at the same time this system is easily scalable and expandable due to the cloud Integration and data centralisation concept.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] SudhaNarang, Kriti Jain, MeghaSaxena, AashnaArora, "Comparison of Face Recognition Algorithms Using Opencv for Attendance System" in International Journal of Scientific and Research Publications, Volume 8, Issue 2, February 2018

[2] Almer John E. Cañete "OpenCV Real-time Face Recognition Attendance System to Online-School Attendances" in International Journal of Innovative Science and Research Technology Volume 6, Issue 5, May – 2021

[3] Sudhir Bussa,Shruti Bharuka,Ananya Mani,Sakshi Kaushik "Smart Attendance System using OPENCV based on Facial Recognition" in International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 9 Issue 03, March-2020

[4] Eli Saber, A. Murat Tekalp "Face detection and facial feature extraction using color, shape and symmetry-based cost functions" in Conference Paper · September 1996

[5] Smitha, Pavithra S Hegde, Afshin "Face Recognition based Attendance Management System" in International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 9 Issue 05, May-2020

[6] Dhanush Gowda H.L , K Vishal , Keertiraj B. R , Neha Kumari Dubey , Pooja M. R "Face Recognition based Attendance System" in International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 9 Issue 06, June-2020