e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:01/January-2021 Impact Factor- 5.354 www.irjmets.com

FACIAL RECOGNITION AND ATTENDANCE SYSTEM USING DLIB AND FACE RECOGNITION LIBRARIES

Shashank Reddy Boyapally*1, Supreethi K.P*2

*1,2Computer Science Engineering, Jawaharlal Nehru Technological University, Hyderabad, India.

ABSTRACT

An Overview to the computer vision algorithms used for facial recognition. The main article idea is to explore an algorithm which can be used in biometric attendance systems with suitable methods and available inputs. The algorithm mainly uses histogram-oriented gradients for finding the faces, estimation of face landmarks, support vector machine to recognize the face and Deep convolutional networks to compare faces. The basis and scientific procedure for facial recognition is described in the article. A basic application is also developed in order to mark the time of the face appears is .csv format and attendance is marked. The article mainly uses dlib and face_recognition libraries in order to provide the functionality.

Keywords: Dlib, face_recognition, DCNN, SVM, HOG, machine learning, face landmark estimation, Attendance system, Computer Vision.

I. INTRODUCTION

In the age of growing automation and mechanization where technology is developing at light speed many opportunities and pathways are created in front of humanity. Analysis of faces, face recognition is one of such pathways which has a promising and a beneficial effect on the society. In many years these technologies have developed thoroughly through artificial intelligence and machine learning technologies. Computer vision is also such field where the tech is very beneficial. The use of such technologies is in demand in our day to day lives these days.

One of the most prominent problems of computer vision system is the face recognition. Many developments which took place in recent years by many of the tech giants is one of most supporting statements. In 2017 September Apple.inc which is one the most break through companies in the world, during the WWDC event announced the Face ID technology. In the similar fashion Facebook also introduced such technology in its social media platform both in Instagram and Facebook where faces of friends are automatically tagged while posting a picture of them. Facebook reports that the efficiency of its algorithm is about 93%.

Such research and development of such problems has resulted in development of many various libraries and APIs for face recognition. These libraries can be used as per our requirements in order to solve our required problem.

Development of many libraries and many other APIs using different technologies and in many supported programming languages is done. Many technologies also find one or more face in the given image as per the given model. Development of so many pathways for a single problem may cause confusion while choosing the solution for the given problem.

This Article will mainly look into the libraries such as face_recogniton and dlib in order to solve our problem of an attendance system. The advantages and the limitations of the libraries will be discussed in the following paper. The reason and the feasibility of the best required solution will also be discussed. Before we discuss the above-mentioned points, Firstly, we discuss the basics of face recognition, we'll discuss the hurdles we face while solving the problem using these technologies, and their solutions.

The research objective is to create a biometric attendance system. The system would be designed such as the algorithm saves the time when the face appears in the video for the first time in a .csv format. To come to such an outcome, we should firstly look into the following tasks:

- Consider the methods and basic principles of face recognition.
- Analyse the methods followed by the libraries to solve the problem.

These points are to be considered while constructing an algorithm, this is done using the libraries and the HOG method in python.

II. RELATED WORK

In these modern times where AI and machine learning have gained so much importance and usage in our day-to-day life, technologies of computer vision are developing actively, with these developments we are able to solve our problems more effectively and precisely. The result of active development has actually led to creation and updates of large number of libraries and solve the problems associated with computer vision. Particularly, in this article we discuss the task of implementing an attendance System by recognizing faces by using the libraries such as face_recognition and dlib.

The best way to get acquainted with such libraries is to read the documentation that will in turn be useful in solving the task and the issues we face. The article [1,2] has actually helped with the documentation of the libraries as mentioned above. Articles [3,4] have focused on theoretical aspects of the issues and building of the face recognition system.

The researchers [7,8] describe the actual methods and technologies for all stages of the development of the recognition system, since in the field of recognition, a huge number of unique solutions have been developed.

The researchers introduce the method of face recognition using the method of Support Vector Machines (SVM), which significantly improve the speed and efficiency of the process of comparison of faces. Scholars have introduced the facial landmark estimation algorithms and techniques which are used to position the faces in the frame. It in turn helps the model in identifying the faces more efficiently and increase the overall quality of the system we are trying to achieve.

Based on the study of documentation and analysis we have come to a conclusion that there no single pathway to conduct a facial recognition rather there are many methods and ways one could achieve this goal. But, pertaining to this article we find it easier to use face_recognition library. We discuss the advantages and hurdles we face during the development of this system.

III. MATERIALS AND METHODS

The following problems should be addressed before devising an algorithm for the attendance system.

- Find Faces – the faces should be recognized no matter what the input is a video from a camcorder or a video from a cc tv footage or any video
- Position of faces – In real world test cases we mainly find that the face is often rotated or not in right position i.e., towards the camera. The main objective of this point will be turning the photo such that it was directly taken facing the camera.
- Identifying the unique facial features – this step can be named as the main step in the facial recognition where the unique facial features of the face are acquired and stored in digital valued forms.
- Identifying the person – the received data from the input video is later compared to the data available with us, if both of the data is similar, we will draw a bounding box green in color, around the face of the person along with the names. If the person is not within the known values, we bound it with a red box.

This article will examine in detail all the steps in building a face recognition system and implement with help of the libraries as mentioned above. One should keep in mind that these results are produced under an Intel i5-8250U, other CPUs might be better as their processing power maybe higher than the above-mentioned CPU.

The first step to develop such a kind of attendance system the main step will be identifying a face from the given frame of video. If any face remains undetected or any other object is treated as a face the system, we are developing remains no good and the results which will be produced will be unsatisfactory. So, to overcome this problem we use one of the most popular algorithms for finding faces in an image which was invented in 2005 by Navneet Dalal and Bill Triggs, Histogram oriented gradient [9]. The algorithm works on the following steps.

To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces:

Fig.-1: Image of Will Ferrel

Our goal will be considering each and every pixel and consider their surrounding pixel and the draw arrows towards the darker pixel. The same processes are continued to all the pixel. At last, we arrive to a situation where all the pixels are replaced by arrows. These arrows are called gradients and they show the flow light in the frame.

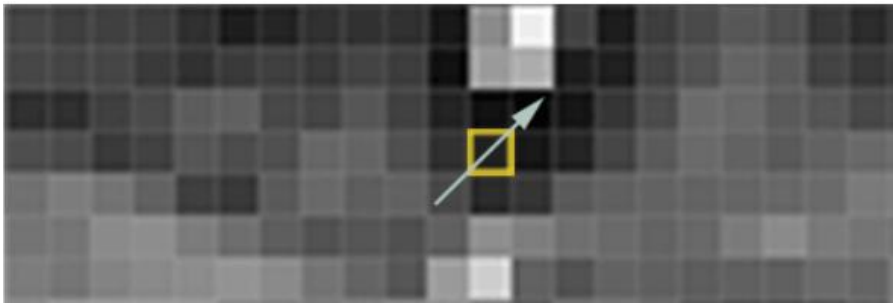

Fig.-2: Pixelated version of fig 1 showing the gradient direction

We'll later break the image into squares of 16 x16 pixels and each block is replaced by the arrow direction which is the strongest in the given block. The end result will be a very simple representation of the face which is captured in the frame.

Later the image is compared to the HOG pattern which was extracted from a bunch of training faces, the most common pattern with the known pattern and marked. This would result us the marking the faces in the obtained frame from the video we received as the input.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:01/January-2021 Impact Factor- 5.354 www.irjmets.com

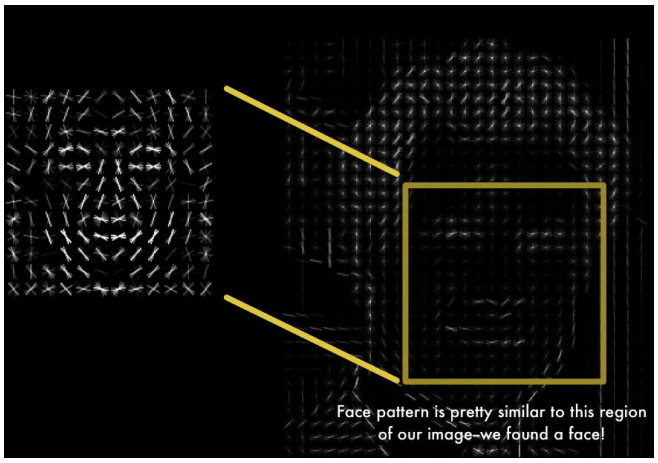

Fig.-3: HOG based image being compared with the HOG which is generated from lots of face images

After finding the face in the image, the next problem we face is the positioning of the face, in most of the images the face not center positioned as it should be required by the algorithm otherwise it'll worsen the performance and accuracy of the algorithm. To solve this situation, we use the face landmark estimation which was invented by Vahid Kazemi and Josephine Sullivan.

The basic idea is we will come up with 68 specific points (called landmarks) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face.

Fig.-4: The 68 landmarks we will locate on every face. This image was created by Brandon Amos of CMU who works on OpenFace.

Now that we have an idea where the mouth and eyes are, we'll rotate, scale, shear the image such that the eyes and mouth are centered as much as possible. By doing this step i.e., centering the face it'll help us a lot in the further step by making it more accurate and efficient.

The next step we shall focus on is the way to tell different faces apart. We can do this by extracting basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, or the space between our eyes etc. So, which measurements shall we take into account? Another problem we might face is while we have a huge database of faces the comparison process will take a lot of time by which it'll be quite inefficient process. At first glance we may feel that the main characteristics of face such as distance between the eyes or the distance between the ears or the length of the nose might be good for the main measurements of the face but later while we take deeper dive, we see that the computer doesn't perceive the face as a whole but it considers pixels of the image.

To solve this problem, it was suggested that we can use a DCNN, which will be trained to identify 128 unique numerical facial features.

But instead of training the network to recognize pictures objects like we did last time, we generate the 128 measurements:

The training process works by following the below steps:

- Load the training face image of a known person
- Load another picture of the same person
- Load a picture of a different person

The Deep convolutional network will adjust the results obtained from the images 1 and 2 such that the obtained 128 characteristics are as similar as possible and image which loaded in 3rd step is as different as possible.

The next step includes the training the deep convolutional neural networks is to generate unique numerical values of the 128 characteristics from a huge no. of faces in the databases and it takes a lot of computational power to train the Deep convolutional neural network.

Even though we use NVIDIA Tesla graphic card which has very powerful GPU cores from the company NVIDIA along with the support for CUDA the learning process takes about 24hrs of time. However, when the neural network will be trained it can take the input of the face never seen before and generate the unique characteristics instantly. This makes the step the most important among the other steps we have discussed, incomplete training may result in unsatisfactory results which may lead to inefficiency. If you do not want to train a model you can always use a pre-trained model and use it in your algorithm so that it can generate the features. In this article we follow the same technique by using a pre-trained model so that our work and economic costs are reduced by great amount.

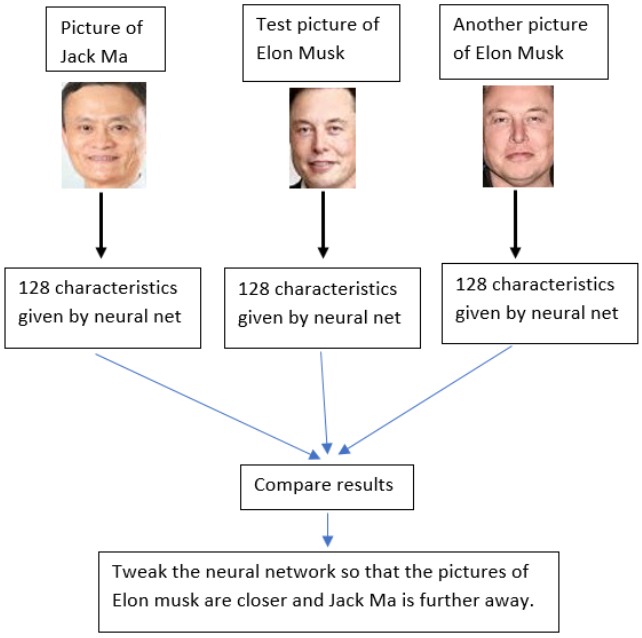

Fig.-5: Representation of the steps discussed above

The last step of the algorithm will be comparing the faces to available faces i.e., the available 128 features which were obtained in the previous step will be compared to the data we have, if we data will match, we enter the time in the file along with the name of the person which will help us create the Attendance System.

The task we face is how do we compare the faces for this algorithm we choose to use the Support vector Machine or SVM.

Using the available features, we train the classifier, the more homogenous the features are the better we can determine the face. The reason for this is that for the classifier we send the value in the form of data set:

$$(\overrightarrow{x_1}, y_1), \ldots., (\overrightarrow{x_n}, y_n)$$

Where y is 1 or -1, and each one indicates the class to which the point belongs. During the training, support vector method, our challenge is to find the maximum separation hyperplane (see Fig.3), simply put, the dataset is divided into classes so that they are difficult to confuse during comparison. The classifier's working time is several milliseconds, ideal for face-to-face identification.

The below points can help in better precision and improved learning of the model [12]:

- Between the data there should be clear intervals by which the model can recognize them into clear groups.
- Huge data may not help in the betterment of model rather the training images should be with less noise and any data of any class should not overlap with another.

IV. IMPLEMENTATION

Now let us build the attendance system as mentioned above. The above steps are followed while implementing the above-mentioned algorithm, the attendance is stored in .csv file which can be later read to know the time when the people have entered in the frame.

The given code implements the above-mentioned libraries to produce the intended Attendance System:

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
# from PIL import ImageGrab
path = 'faces'
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:%S')
            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(1)
while True:
    success, img = cap.read()
    # img = captureScreen()
    imgS = cv2.resize(img, (0, 0), None, 1, 1)
    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)
        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 , x2 , y2 , x1
                cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
                markAttendance(name)
            else:
                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1, x2, y2, x1
                cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
        cv2.imshow('Webcam', img)
        if cv2.waitKey(20) & 0xFF == ord('q'):
            break
    # When everything done, release the capture
    cap.release()
cv2.destroyAllWindows()
```

V. RESULTS

The final version of the algorithm is based on the above code using the face_recognition and dlib libraries. The result of the algorithm is shown in the below figure fig.6 which shows the interface in an application when a face is detected. The .csv file format is also shown where the details i.e., name and time of arrival of the detected face in the frame is noted down.

Fig.-6: Identification of faces (Elon Musk and Jack ma) interface.

Attendance.csv

Name, time

JACK MA,11:13:26

ELON MUSK,11:13:58

Based on the analysis of the above steps as said before we see that there is no single well-defined pathway for recognizing faces rather there are many methods which we can be followed to achieve the given task. There are huge number of methods which can be followed for searching, estimating the positions and comparing the faces. One should always be careful while choosing the method which he chooses based on the requirement one has. Therefore, articles with detailed information about such technologies more important than ever to decide what to choose.

VI. CONCLUSIONS

As mentioned in the above article the technologies related to the face recognition and computer vision are developing at a very rapid scale. For more accuracy one should not only focus on the software side which can be fooled by images of people, hardware should also be developed such as IR technologies which in turn scan the depth of the image.

The above article concludes that the Attendance system was made with combination of many different algorithms and is works with good accuracy and efficiency. It should be noted that plenty of other algorithms such as Haar cascade were also explored which can be replaced by HOG but it was found that the HOG method is more efficient while having a little number of faces which is generally with the case of biometric and Attendance systems. One could even improve the performance of the algorithm by employing CUDA GPU cores and more powerful CPU which would be available in a server system rather compared to a weak Personal Computer. However, the design logic along with key points of different algorithms were discussed in the article.

ACKNOWLEDGMENT

The authors thank anonymous reviewers for their attentiveness and valuable comments.

VII. REFERENCES

[1] face_recognition: Face recognition library [Electronic Resource] – Access mode: https://face-recognition.readthedocs.io/en/latest/readme.html
[2] Dlib Python API Tutorials [Electronic resource] – Access mode: http://dlib.net/python/index.html
[3] Face detection and estimation [Electronic Resource] - https://github.com/davisking/dlib/tree/master/python_examples
[4] Finding Face landmarks [Electronic resource] - https://gist.github.com/ageitgey/ae340db3e493530d5e1f9c15292e5c74
[5] Face Detection Algorithms and Techniques [Electronic resource] – Access mode: https://facedetection.com/algorithms/
[6] Support Vector Machines [Electronic resource] – Access mode: http://scikit-learn.org/stable/modules/svm.html
[7] Face recognition with OpenCV, Python, and deep learning [Electronic resource] – Access mode: https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
[8] Z. Rybchak, and O. Basystiuk, “Analysis of computer vision and image analysis technics,” ECONTECHMOD: an international quarterly journal on economics of technology and modelling processes, Lublin: Polish Academy of Sciences, vol. 6, no. 2, pp. 79-84, 2017
[9] Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision & Pattern Recognition (CVPR ’05), Jun 2005, San Diego, United States. pp.886–893, ff10.1109/CVPR.2005.177ff. ffinria-00548512f
[10] Vahid Kazemi; Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees, 2014 IEEE Conference on Computer Vision and Pattern Recognition, 23-28 June 2014.
[11] OpenFace [Electronic resource] – Access mode: https://cmusatyalab.github.io/openface/
[12] R. Raja, Face Recognition Using OpenCV and Python [Electronic resource] - Access mode: https://www.superdatascience.com/opencv-face-recognition/