WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Study of Implementing Automated Attendance System Using Face Recognition Technique

Mr. AKSHAT GAUR[1], Ms. TANISHKA[2], Ms. AASTHA[3], Mr. HIMANSHU PANDEY[4], Dr. SONAM KAUSHIK[5]

[1]Student, BCA, Maharaja Surajmal Institute, Delhi- 110058
[2] Student, BCA, Maharaja Surajmal Institute, Delhi- 110058
[3] Student, BCA, Maharaja Surajmal Institute, Delhi-110058
[4] Student, BCA, Maharaja Surajmal Institute, Delhi-110058
[5]Assistant Professor, BCA, Maharaja Surajmal Institute, Delhi- 110058.

ABSTRACT:

System control in computer-based communication faces a major challenge with authentication. A significant area of biometric verification is human face recognition, which finds extensive use in door control systems, video monitors, HCI, and network security, among other areas. This article outlines a technique for the Student Attendance System that will use the Personal Component Analysis (PCA) algorithm to interact with facial recognition technology. By keeping a log of the students' clock-in and clock-out times, the system will automatically record the students' attendance in a classroom setting and give faculty members easy access to student data.

Index Terms—face recognition system, automatic attendance, authentication, bio-metric, PCA.

INTRODUCTION

Due to the topic's practical significance as well as cognitive scientists' theoretical interest, face recognition is as ancient as computer vision. Face recognition is the primary means by which people identify themselves, and as such, it has always been a major focus of research, even when alternative methods (such fingerprints or iris scans) may be more accurate. Additionally, face recognition is non-invasive. Since face recognition technology needs almost no effort on the part of the user compared to other biometric solutions, it is gradually becoming a universal biometric solution. In essence, there are three main areas in which biometric face recognition is used: visitor management systems, personnel management systems, and, last but not least, authorization and access control systems.

In the past, attendance was manually recorded by students using a form that the teachers handed out during class. This process takes a lot of time. Furthermore, in a large classroom setting with dispersed branches, it is exceedingly challenging to confirm with each student individually whether or not the verified pupils are truly answering.

In this study, the authors show how facial recognition can be used to automatically record a registered person's presence within the designated location as part of an efficient attendance system. The proposed system additionally keeps track of each person's entrance in relation to a universal system time by maintaining a log file.

A. Background and Related Work

During the 1960s, the first attempts at face recognition were made using a semi-automatic system. Marks were created on photos to identify various features, such as the eyes, nose, and mouth. They were then compared to the reference data and used to calculate ratios and distances. Goldstein, Harmon, and Lesk developed a system of 21 subjective markers, including lip thickness and color, in the early 1970s. Due to the subjective nature of many of the measurements, which are still conducted entirely by hand, this proved to be much more difficult to automate.

By measuring several facial traits and mapping them all onto a global template, Fisher and Elschlagerb [3] discovered that these features lacked sufficient unique data to accurately depict an adult face.

The Connectionist approach [4] is an alternative method that aims to categorize the human face by combining a set of distinguishing markers with a variety of movements. Neural net concepts and 2-dimensional pattern recognition are typically used to do this. This approach hasn't been widely used yet because it often needs a large number of training faces to reach reasonable accuracy.
Very general pattern recognition was used in the development of the first completely automated system [5]. It generated a set of patterns for an image in relation to this model by comparing faces to a generic face model of predicted traits. This method is primarily statistical and uses the grey scale value and histograms.

SYSTEM OVERVIEW

The Eigen face technique, first presented by Kirby and Sirovich at Brown University in 1988, was employed by the current authors for face recognition. The process computes Eigen faces [8], or faces made up of eigenvectors, by examining face photos. The presence and identity of a face are determined by comparing Eigen faces. The system that Turk and Pentland created involves a five-step procedure [1]. The system must first be initialized by being fed a set of face training photos. This defines the face space, which is a collection of like-face images. It then computes an Eigen face for each face it encounters. Whether the image is a face at all can be ascertained via statistical analysis and comparison with known faces. After an image is identified as a face, the algorithm will decide if it recognizes the face or not. The algorithm can learn to recognize an unknown face if it is shown repeatedly. This is the optional final stage.

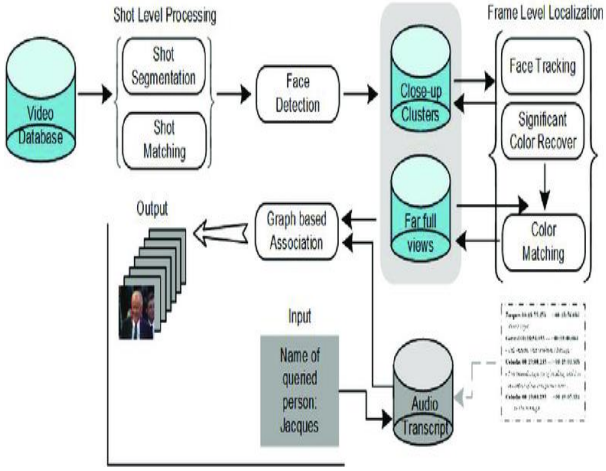

Fig. 1. Architecture of the system

FLTK (Light Tool Kit) and OpenCV (Open Source Computer Vision Library) are the two primary components utilized in the implementation approach. One of the objectives of OpenCV is to offer an easy-to-use computer vision infrastructure that enables individuals to rapidly develop somewhat complex vision applications. The OpenCV library has more than 500 functions covering a wide range of vision-related topics. OpenCV is the main technology used in face recognition, while FLTK is used in interface design. The user positions himself in front of the camera at a minimum distance of 50 cm, and the camera captures his image for input. After being retrieved from the picture, the frontal face is kept and transformed to grayscale. After applying the Principal Component Analysis (PCA) algorithm [7] on the photos, the Eigen values are saved as an xml file. The frontal face of the user is taken out of the recorded video frame by means of the camera upon request for recognition. The closest neighbor's stored data and the recalculated Eigen value for the test face are compared.

PCA (Principal Component Analysis)

1. Applications like face recognition and image reduction have made extensive use of the PCA approach. PCA is a popular method for identifying patterns in data and for highlighting the similarities and differences between various data sets by representing the data as eigenvectors [6]. The PCA process is summed up in the steps that follow. Let *{D1,D2,...DM}* be the training data set. The average *Avg* is defined by:

$$Avg = \frac{1}{M} \sum_{i=1}^{M} Di$$

2. Each element in the training data set differs from *Avg* by the vector *Yi=Di-Avg*. The covariance matrix *Cov* isobtained as:

$$Cov = \frac{1}{M} \sum_{i=1}^{M} Yi.Yi^{T}$$

3. Choose M' significant eigenvectors of *Cov* as *EK's*, andcompute the weight vectors *Wik* for each element in the Training data set, where k varies from 1 to M'.

$$W_{ik} = E_{k}^{T}.(D_{i} - Avg), \forall i, k$$

SYSTEM IMPLEMENTATION

A. Three fundamental steps have been used to implement the suggested system: A. find and extract the face image, then store the extracted face data in an XML file for later use. Acquire knowledge of and practice with the face image, then compute its eigenvalue and eigenvector. Identify and compare face photographs to the database of face images that is stored in an XML file [1].

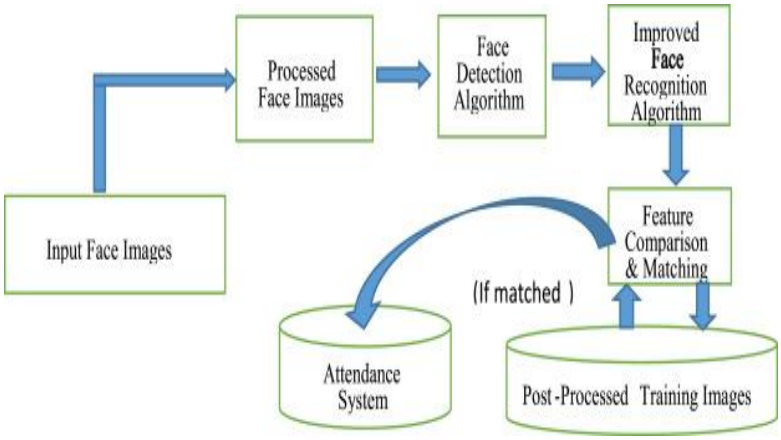

Fig. 2.

B. *Face Detection and Extract*

At first, openCAM_CB () is called to open the camera for image capture. Next the frontal face [2] is extracted from the Video frame by calling the function Extract Face (). The Extract Face () function uses the OpenCv HaarCascade Method to load the haarcascade_ frontalface_alt_tree.xml as the classifier. The classifier outputs a "1" if the region is likely to show the object (i.e., face), and "0" otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed such a manner That it can be easily "resized" in order to be able to find the Objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure is done Several times at different scales. After the face is detected it Is clipped into a gray scale image of 50x50 pixels.
Is clipped into a gray scale image of 50x50 pixels.

C. *Learn and Train Face Images*

Learn () function which performs the PCA algorithm on the training set. The learn () function implementation is done in four steps:

1. Load the training data.
2. Do PCA on it to find a subspace.
3. Project the training faces onto the PCA subspace.
4. Save all the training information.
    a. Eigenvalues
    b. Eigenvectors

c. The average training face image
d. Projected face image
e. Person ID numbers

Calling the built-in OpenCV PCA method, cvCalcEigen Objects (), yields the PCA subspace. When cvCalcEigenObjects () returns, the remaining code in doPCA () generates the output variables that will store the PCA results [5]. Before performing PCA, the dataset needs to be "centered." This entails locating the average image for each of our face images, which is an image where each pixel has the average value for that pixel across all of the training set's face photographs. The average face's pixel values are subtracted from each training image to center the dataset. Within cvCalcEigenObjects (), it takes place. But we need to hold onto the average image, as it will be needed later to project the data for that purpose it is needed to allocate memory for the average image and the image is a floating-point image. Now we have found a subspace using PCA, we can convert the training images to points in this subspace. This step is called "projecting" the training image. The OpenCV function for this step is called cvEigenDecomposite(). Then all the data for the learned face representation is saved as an XML file using OpenCV's built-in persistence functions.

D. Recognize and Identification

The Eigenface program's recognition phase is implemented by the Recognize () function [5]. There are only three steps in it. You are already familiar with the first two: loading the face images and projecting them onto the subspace. The ground truth for the person ID number is stored in personNumTruthMat, and the call to loadFaceImgArray() loads the face images listed in the train.txt into the faceImgArr. In this case, the local variable n TestFaces contains the number of face images.

Along with loading the majority of the other training data, such as nEigens, EigenVectArr, pAvgTrainImg, and so forth, we also need to load the global variable n Train Faces. That is done for us by the functionloadTrainingData(). OpenCV uses the file's name to find and load each data value. Projecting each test image onto the PCA subspace and locating the nearest projected training image is the last step in the recognition phase, which is completed once all the data have been loaded. Similar to the face-projection code in the learn () function, the call to cvEigenDecomposite() projects the test picture.

We pass it the number of eigen values as previously.
(nEigens), and the array of eigenvectors (eigenVectArr). This time, however, we pass a test image, instead of a training image, as the first parameter. The output from cvEigenDecomposite() is stored in a local variable - projectedTestFace. Because there's no need to store the projected test image, we used a C array for projectedTestFace, rather than an OpenCV matrix.

The Find Nearest Neighbor () function computes distance from the projected test image to each projected training example. The distance basis here is "Squared Euclidean Distance." To calculate Euclidean distance between two points, we need to add up the squared distance in each dimension, and then take the square root of that sum. Here, we take the sum, but skip the square root step. The final result is the same, because the neighbor with the smallest distance also has the smallest squared distance, so we can save some computation time by comparing squared values.

EXPERIMENT AND RESULT

The step of the experiments process are given below:
Face Detection:

Start capturing images through web camera of the client side: Begin:
//Pre-process the captured image and extract face image

//calculate the Eigen value of the captured face image and compared with Eigen values of existing faces in the database.

//If Eigen value does not matched with existing ones, save the new face image information to the face database (xml file).

//If Eigen value matched with existing one then recognition step will done.

End;

Face Recognition:
Using PCA algorithm the following steps would be followed in for face recognition:

Begin:

// Find the face information of matched face image in from the database.

// update the log table with corresponding face image and system time that makes completion of attendance for an individual students.

end;

This section presents the results of the experiments conducted to capture the face into a grey scale image of 50x50 pixels.

TABLE 1: DESCRIBES THE OPENCV FUNCTION USED IN THE PROPOSED SYSTEM AND ITS EXECUTION RESULTS.

Test data	**Expected Result**	**Observed Result**	**Pass/ Fail**
OpenCAM_CB()	**Connects with the installed camera and Starts playing.**	**Camerastarted.**	**pass**
LoadHaar Classifier()	**Loads the HaarClassifier Cascade files for frontal face**	**Gets ready for Extraction.**	**Pass**
ExtractFace()	**Initiates the Paul-Viola Face extracting Frame work.**	**Face extracted**	**Pass**
Learn()	**Start the PCAAlgorithm**	**Updates the facedata. xml**	**Pass**
Recognize()	**It compares the input face with the saved Faces.**	**Nearest face**	**Pass**

Fig. 3. Training Images

TABLE 2: FACE DETECTION AND RECOGNITION RATE

Face Orientations	**Detection Rate**	**Recognition Rate**
O^0 (Frontal face)	**98.7 %**	**95%**
18º	**80.0 %**	**78%**
54º	**59.2 %**	**58%**
72º	**0.00 %**	**0.00%**
90º (Profile face)	**0.00 %**	**0.00%**

We conducted a series of tests to show the effectiveness of the suggested approach. 30 distinct pictures of ten different people make up the training

set. An example binary picture identified by the Paul-Viola Face extraction Frame work detection technique, as demonstrated by the Extract Face() function, is displayed in Figure 3.

Table 2 shows that the face angle-related reduction in the camera face detection and identification rate occurs as the angle increases.

CONCLUSION AND FUTURE WORK

In order to obtain the attendance of individuals and to the authors suggested that institutions and organizations use a facial recognition technology-based attendance management system to track employees' arrival and departure times. The system continuously monitors each student's attendance at the entry and departure points. The early experiment's results show improved performance in attendance estimations when compared to the traditional black and white attendance methods. The present work focuses mostly on face identification algorithms from photo or video frames.

In order to improve facial recognition efficiency, the authors want to take use of the interactions that occur between our system, users, and administrators in further research. On the other hand, mobile-based facial recognition represents a whole new application domain for our technology. This can aid in educating the broader audience about everyone seen on video by a cell phone, regardless of whether they are authorized to access a centralized database.

REFERENCES

[1] M. A. Turk and A. P. Pentland, "Eigenfaces for Face Recognition," in Proc. IEEE Conference on Pattern Recognition and Computer Vision, pages 586–591. 1991

[2] .A. Goldstein, J., Harmon, L. D., and A. "Identification of Human Faces," by B. Lesk, in Proc. IEEE Conference on Pattern Recognition and Computer Vision, vol. 59, May 1971, pp. 748–760

[3] M. Fischler, A., and R. "The Representation and Matching of Pictorial Structures," by A. Elschlager, IEEE Transaction on Computer, vol. C-22, 1973, pp. 67-92.

[4] S. In Proceedings TENCON 2000, vol., S. R. Abibi discusses "Simulating evolution: connectionist metaphors for studying human cognitive behaviour." 1 pp. 167–173, 2000.

[5] Ethical Image, vol. 59, pp. 768–773, May 1992, Y. Cui, J. S. Jin, S. Luo, M. Park, and S. S. L. AuIn the proceedings of the 5th International Conference on Computer Vision and Graphical Image, vol. 59, pp. 768–773, May 1992, Y. Cui, J. S. Jin, S. Luo, M. Park, and S. S. L. Au presented their automated pattern recognition and defect inspection system.

[6] Y. Zhang and C. Liu, "Face recognition through genetic algorithms and kemel principal component analysis," IEEE Workshop on Neural Networks for Signal Processing, September 4-6, 2002.

[7] Journal of IEEE International Conference on Industrial Technology, Dec. 1994, pp. 434–438; J. Zhu and Y. L. Yu, "Face Recognition with Eigenfaces."

[8] M. H. Yang, N. Ahuja, and D. Kriegmao, “Face recognition using kernel eigenfaces,” *IEEE International Conference on Image Processing*, vol. 1, pp. 10-13, Sept. 2000.

[9] "3D Facial Recognition: A Quantitative Analysis," 38th Annual 2004 International Carnahan Conference on Security Technology, T. D. Russ, M. W. Koch, and C. Q. Little, 2004.

[10] "Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About," by P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell, Proceedings of the IEEE, vol. 94, Issue 11, 2006.

[11] "Personal based authentication by face recognition," in proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management, pp. 81–85, 2008, Y.-W. Kao, H.-Z. Gu, and S.-M. Yuan.

[12] Image Processing: Principles and Applications, A. T. Acharya and A. Ray, New York: Wiley, 2005.