Study of Implementing Automated Attendance System Using Face Recognition Technique

Nirmalya Kar, Mrinal Kanti Debbarma, Ashim Saha, and Dwijen Rudra Pal

***Abstract*—Authentication is a significant issue in system control in computer based communication. Human face recognition is an important branch of biometric verification and has been widely used in many applications, such as video monitor system, human-computer interaction, and door control system and network security. This paper describes a method for Student's Attendance System which will integrate with the face recognition technology using Personal Component Analysis (PCA) algorithm. The system will record the attendance of the students in class room environment automatically and it will provide the facilities to the faculty to access the information of the students easily by maintaining a log for clock-in and clock-out time.**

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

I. Introduction

Face recognition is as old as computer vision, both because of the practical importance of the topic and theoretical interest from cognitive scientists. Despite the fact that other methods of identification (such as fingerprints, or iris scans) can be more accurate, face recognition has always remains a major focus of research because of its non-invasive nature and because it is people's primary method of person identification. Face recognition technology is gradually evolving to a universal biometric solution since it requires virtually zero effort from the user end while compared with other biometric options. Biometric face recognition is basically used in three main domains: time attendance systems and employee management; visitor management systems; and last but not the least authorization systems and access control systems.

Traditionally, student's attendances are taken manually by using attendance sheet given by the faculty members in class, which is a time consuming event. Moreover, it is very difficult to verify one by one student in a large classroom environment with distributed branches whether the authenticated students are actually responding or not.

The present authors demonstrate in this paper how face recognition can be used for an effective attendance system to automatically record the presence of an enrolled individual within the respective venue. Proposed system also maintains a log file to keep records of the entry of every individual with respect to a universal system time.

Manuscript received March 8, 2012; revised May 14, 2012.

The authors are with the Computer Science and Engineering Department, National Institute of Technology, Agartala, India (e-mail: nirmalya.kar@gmail.com, mkdb06@gmail.com, ashim.nita@gmail.com, dwijen.rudrapal@gmail.com).

A. Background and Related Work

The first attempts to use face recognition began in the 1960's with a semi-automated system. Marks were made on photographs to locate the major features; it used features such as eyes, ears, noses, and mouths. Then distances and ratios were computed from these marks to a common reference point and compared to reference data. In the early 1970's Goldstein, Harmon and Lesk [2] created a system of 21 subjective markers such as hair colour and lip thickness. This proved even harder to automate due to the subjective nature of many of the measurements still made completely by hand.

Fisher and Elschlagerb [3] approaches to measure different pieces of the face and mapped them all onto a global template, which was found that these features do not contain enough unique data to represent an adult face.

Another approach is the Connectionist approach [4], which seeks to classify the human face using a combination of both range of gestures and a set of identifying markers. This is usually implemented using 2-dimensional pattern recognition and neural net principles. Most of the time this approach requires a huge number of training faces to achieve decent accuracy; for that reason it has yet to be implemented on a large scale.

The first fully automated system [5] to be developed utilized very general pattern recognition. It compared faces to a generic face model of expected features and created a series of patters for an image relative to this model. This approach is mainly statistical and relies on histograms and the gray scale value.

II. System Overview

The present authors used the eigenface approach for face recognition which was introduced by Kirby and Sirovich in 1988 at Brown University. The method works by analyzing face images and computing eigenface [8] which are faces composed of eigenvectors. The comparison of eigenface is used to identify the presence of a face and its identity. There is a five step process involved with the system developed by Turk and Pentland [1]. First, the system needs to be initialized by feeding it a set of training images of faces. This is used to define the face space which is set of images that are face like. Next, when a face is encountered it calculates an eigenface for it. By comparing it with known faces and using some statistical analysis it can be determined whether the image presented is a face at all. Then, if an image is determined to be a face the system will determine whether it knows the identity of it or not. The optional final step is that if an unknown face is seen repeatedly, the system can learn to recognize it.

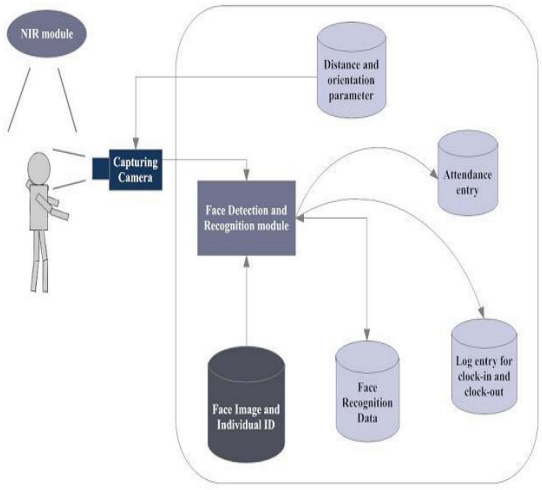

Fig. 1. Architecture of the system

The two main components used in the implementation approach are open source computer vision library (OpenCV) and Light Tool Kit (FLTK). One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. OpenCV library contains over 500 functions that span many areas in vision. The primary technology behind Face recognition is OpenCV; the interface is designed using FLTK. The user stands in front of the camera keeping a minimum distance of 50cm and his image is taken as an input. The frontal face is extracted from the image then converted to gray scale and stored. The Principal component Analysis (PCA) algorithm [7] is performed on the images and the eigen values are stored in an xml file. When a user requests for recognition the frontal face is extracted from the captured video frame through the camera. The eigen value is re-calculated for the test face and it is matched with the stored data for the closest neighbour.

A. PCA (Principal Component Analysis)

PCA method has been widely used in applications such as face recognition and image compression. PCA is a common technique for finding patterns in data, and expressing the data as eigenvector to highlight the similarities and differences between different data [6]. The following steps summarize the PCA process.

1. 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* is obtained as:

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

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

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

III. System Implementation

The proposed system has been implemented with the help of three basic steps: **A**. detect and extract face image and save the face information in an xml file for future references. **B**. Learn and train the face image and calculate eigen value and eigen vector of that image. **C.** Recognise and match face images with existing face images information stored in xml file [1].

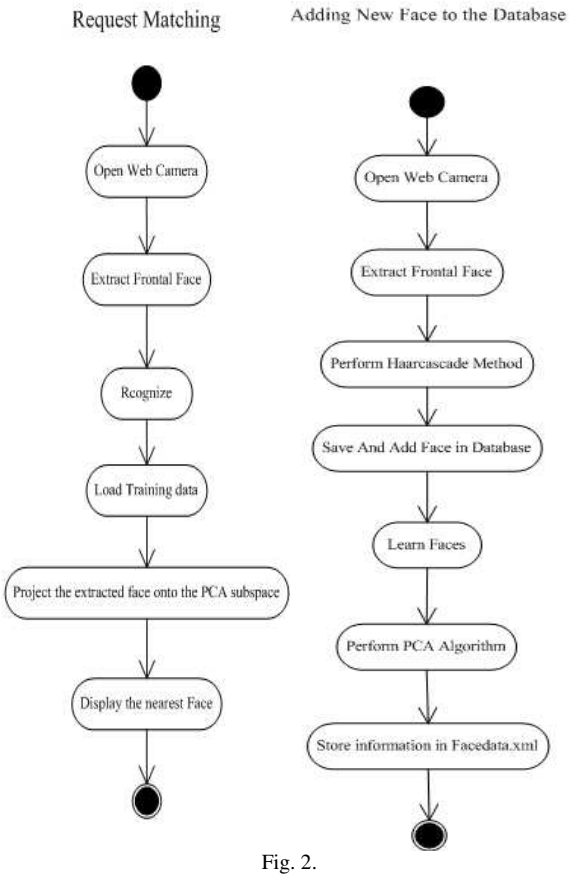

Fig. 2.

A. 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 ExtractFace(). The ExtractFace() 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.

B. 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

The PCA subspace is calculated by calling the built-in OpenCV function for doing PCA, cvCalcEigen Objects(). The remainder of doPCA() creates the output variables that will hold the PCA results when cvCalcEigenObjects() returns [5].

To do PCA, the dataset must first be "centered." For our face images, this means finding the average image - an image in which each pixel contains the average value for that pixel across all face images in the training set. The dataset is centred by subtracting the average face's pixel values from each training image. It happens inside cvCalcEigenObjects().

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.

C. Recognise and Identification

Recognize() function, which implements the recognition phase of the Eigenface program [5]. It has just three steps. Two of them - loading the face images and projecting them onto the subspace - are already familiar. The call to loadFaceImgArray() loads the face images, listed in the train.txt, into the faceImgArr and stores the ground truth for person ID number in personNumTruthMat. Here, the number of face images is stored in the local variable, n TestFaces.

We also need to load the global variable n TrainFaces as well as most of the other training data - nEigens, EigenVectArr, pAvgTrainImg, and so on. The functionloadTrainingData() does that for us. OpenCV locates and loads each data value in the XML file by name.

After all the data are loaded, the final step in the recognition phase is to project each test image onto the PCA subspace and locate the closest projected training image. The call to cvEigenDecomposite(), projects the test image, is similar to the face-projection code in the learn() function.

As before, we pass it the number of eigen values (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 findNearestNeighbor() 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 neighbour with the smallest distance also has the smallest squared distance, so we can save some computation time by comparing squared values.

IV. Experiment and Result

The step of the experiments process are given below:

1. 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;

2. 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.	Camera started.	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 PCA Algorithm	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 performed a set of experiments to demonstrate the efficiency of the proposed method. 30 different images of 10 persons are used in training set. Figure 3 shows a sample binary image detected by the ExtractFace() function using Paul-Viola Face extracting Frame work detection method. From table 2 it is been observed that with the increasing of face angle with respect to camera face detection and recognition rate is become decreases.

V. CONCLUSION AND FUTURE WORK

In order to obtain the attendance of individuals and to record their time of entry and exit, the authors proposed the attendance management system based on face recognition technology in the institutions/organizations. The system takes attendance of each student by continuous observation at the entry and exit points. The result of our preliminary experiment shows improved performance in the estimation of the attendance compared to the traditional black and white attendance systems. Current work is focused on the face detection algorithms from images or video frames.

In further work, authors intend to improve face recognition effectiveness by using the interaction among our system, the users and the administrators. On the other hand, our system can be used in a completely new dimension of face recognition application, mobile based face recognition, which can be an aid for common people to know about any person being photographed by cell phone camera including proper authorization for accessing a centralized database.

REFERENCES

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

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

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

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

[5] Y. Cui, J. S. Jin, S. Luo, M. Park, and S. S. L. Au, "Automated Pattern Recognition and Defect Inspection System," in *proc. 5th International Conference on Computer Vision and Graphical Image*, vol. 59, pp. 768 – 773, May 1992

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

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

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

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

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

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