Face Recognition System with GUI Using Digital Image Processing

K. Sharath Reddy, M.C. Sankalp, K. Pradeep Kumar, K.P. Shashidhar

ABSTRACT- A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. An approach to the detection and identification of human faces is presented, and a working, near-real-time face recognition system which tracks a subject's head and then recognizes the person by comparing characteristics of the face to those of known individuals is described. This approach treats face recognition as a two-dimensional recognition problem, taking advantage of the fact that faces are normally upright and thus may be described by a small set of 2-D characteristic views. Face images are projected onto a feature space (`face space') that best encodes the variation among known face images. The face space is defined by the `Eigen faces', which are the eigenvectors of the set of faces; they do not necessarily correspond to isolated features such as eyes, ears, and noses. The framework provides the ability to learn to recognize new faces in an unsupervised manner. In this report we discuss about the feature based recognition and Eigen face method for facial analysis. In feature based facial recognition method the importance is given to the facial features, whereas the Eigen face method gives preference to the face. By combining both the above methods we obtain

Keywords— "Feature Based Eigen face Method" for facial recognition.

I. INTRODUCTION

GRAPHICAL USER INTERFACE

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. In general, these objects already have meanings to most computer users. For example, when you move a slider, value changes; when you press an OK button, your settings are applied and the dialog box is dismissed. Of course, to leverage this built-in familiarity, you must be consistent in how you use the various GUI-building components. Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by the design of the interface. The sections that follow describe how to create GUIs with MATLAB. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs.

Manuscript Received on December 2014.

K. Sharath Reddy, Dept. of Electronics and Communication Engineering, MLRIT, Dundigal, Hyderabad Telangana, India.
M.C. Sankalp, Dept. of Electronics and Communication Engineering, MLRIT, Dundigal Hyderabad Telangana, India
K. Pradeep Kumar, Dept. of Electronics and Communication Engineering, MLRIT, Dundigal Hyderabad Telangana, India
K.P. Shashidhar, Dept. of Electronics and Communication Engineering, MLRIT, Dundigal Hyderabad Telangana, India

II. CREATING GUIS WITH GUIDE

MATLAB implements GUIs as figure windows containing various styles of uicontrol objects. You must program each object to perform the intended action when activated by the user of the GUI. In addition, you must be able to save graphical user interface development environment.

GUI DEVELOPMENT ENVIRONMENT

The process of implementing a GUI involves two basic tasks:

- Laying out the GUI components
- Programming the GUI components

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the callbacks - the functions that execute and launch your GUI. All of these tasks are simplified by GUIDE.

THE IMPLEMENTATION OF A GUI

While it is possible to write an M-file that contains all the commands to lay out a GUI, it is easier to use GUIDE to lay out the components interactively and to generate two files that save and launch the GUI:

FIG-file -contains a complete description of the GUI figure and all of its
children (uicontrols and axes), as well as the values of all object properties.

M-file -contains the functions that launch and control the GUI and the
callbacks, which are defined as sub functions. This M-file is referred to as the
application M-file in this documentation.

Note that the application M-file does not contain the code that lays out the uicontrols; this information is saved in the FIG-file.

The following diagram illustrates the parts of a GUI implementation

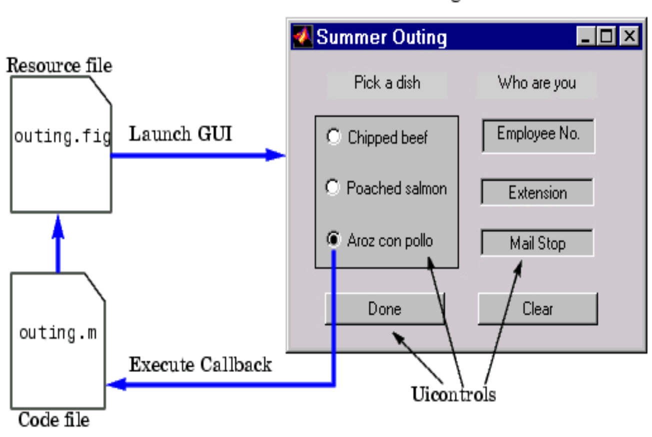

FIGURE 1: Parts of GUI Implementation.

GUIDE TOOLS

The GUIDE tools are available from the Layout Editor shown in the figure below. The tools are called out in the figure and described briefly below. Subsequent sections show you how to use them.

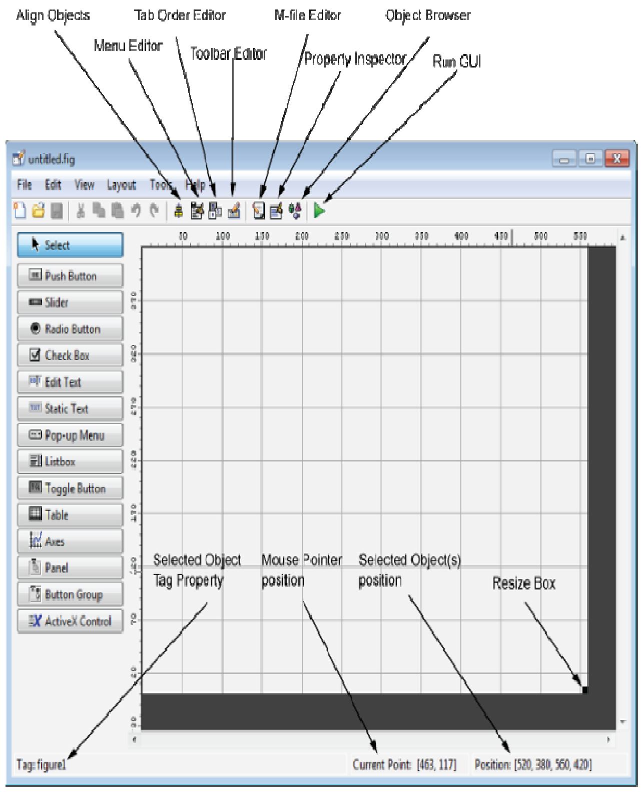

FIGURE 2: GUI layout.

TABLE 1: Tools used in GUI.

Use This Tool...	To...
Layout Editor	Select components from the component palette, at the left side of the Layout Editor, and arrange them in the layout area. See Add Components to the GUIDE Layout Area for more information.
Figure Resize Tab	Set the size at which the GUI is initially displayed when you run it. See Set the GUIDE GUI Size for more information.
Menu Editor	Create menus and context, i.e., pop-up, menus. See Create Menus for GUIDE GUIs for more information.
Align Objects	Align and distribute groups of components. Grids and rulers also enable you to align components on a grid with an optional snap-to-grid capability. See Align GUIDE GUI Components for more information.
Tab Order Editor	Set the tab and stacking order of the components in your layout. See Customize Tabbing Behavior in a GUIDE GUI for more information.
Toolbar Editor	Create Toolbars containing predefined and custom push buttons and toggle buttons. SeeCreate Toolbars for GUIDE GUIs for more information.
Icon Editor	Create and modify icons for tools in a toolbar. See Create Toolbars for GUIDE GUIs for more information.
Property	Set the properties of the components in your layout.

Use This Tool...	To...
Inspector	It provides a list of all the properties you can set and displays their current values.
Object Browser	Display a hierarchical list of the objects in the GUI. See View the GUIDE GUI Object Hierarchyfor more information.
Run	Save and run the current GUI.
Editor	Display, in your default editor, the code file associated with the GUI. See Files Generated by GUIDE for more information.
Position Readouts	Continuously display the mouse cursor position and the positions of selected objects

Now we focus on image-based face recognition. Given a picture taken from a digital camera, we'd like to know if there is any person inside, where his/her face locates at, and who he/she is. Towards this goal, we generally separate the face recognition procedure into three steps: **Face Detection**, **Feature Extraction**, and **Face Recognition** (shown at Fig).

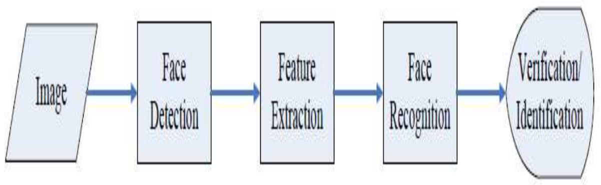

Figure: 3 Configuration of a general face recognition structure Face Detection Feature Extraction Face Recognition Image Verification/ Identification.

III. FACE DETECTION

The main function of this step is to determine (1) whether human faces appear in a given image, and (2) where these faces are located at. The expected outputs of this step are patches containing each face in the input image. In order to make further face recognition system more robust and easy to design, **face alignment** are per-formed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for re-gion-of-interest detection, retargeting, video and image classification, etc.

VIOLA-JONES ALGORITHM

The **Viola–Jones object detection framework** is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. This algorithm is implemented in open CV as cv Haar Detect Objects ()

FEATURE TYPES AND EVALUATION

The features employed by the detection framework universally involve the sums of image pixels within rectangular areas. As such, they bear some resemblance to [[Haar-like features| Haar basis functions]], which have been used previously in the realm of image-based object detection. However, since the features used by Viola and Jones all rely on more than one rectangular area, they are generally more complex. The figure at right illustrates the four different types of features used in the framework. The value of any given feature is always simply the sum of the pixels within clear rectangles subtracted from the sum of the pixels within shaded rectangles. As is to be expected, rectangular features of this sort are rather primitive when compared to alternatives such as [[steerable filter]]s. Although they are sensitive to vertical and horizontal features, their feedback is considerably coarser. However, with the use of an image representation called the [[summed area table|integral image]], rectangular features can be evaluated in "constant" time, which gives them a considerable speed advantage over their more sophisticated relatives. Because each rectangular area in a feature is always adjacent to at least one other rectangle, it follows that any two-rectangle feature can be computed in six array references, any three-rectangle feature in eight, and any four-rectangle feature in just nine.

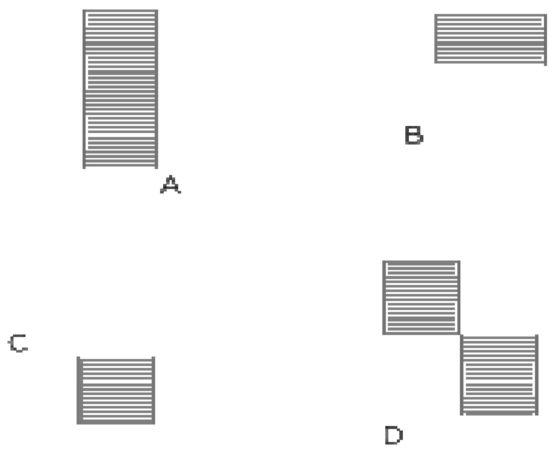

FIGURE 4: Feature types used by Viola and Jones.

LEARNING ALGORITHM

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub window, there are a total of 162,336 possible features, and it would be prohibitively expensive to evaluate them all. Thus, the object detection framework employs a variant of the learning algorithm [[AdaBoost]] to both select the best features and to train classifiers that use them.

CASCADE ARCHITECTURE

The evaluation of the strong classifiers generated by the learning process can be done quickly, but it isn't fast enough to run in real-time. For this reason, the strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the cascade a classifier rejects the sub-window under inspection, no further processing is performed and continue on searching the next sub-window (see figure at right). The cascade therefore has the form of a degenerate tree. In the case of faces, the first classifier in the cascade – called the attentional operator – uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40%. The effect of this single classifier is to reduce by roughly half the number of times the entire cascade is evaluated.

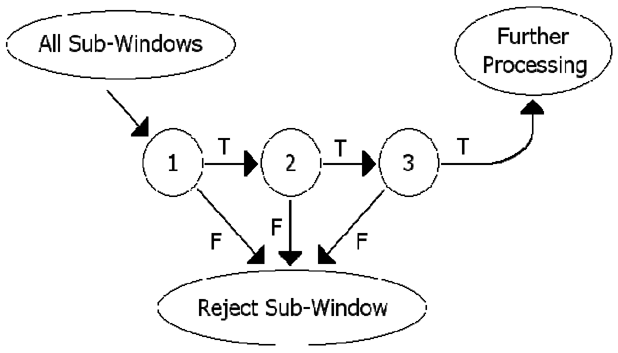

FIGURE 5: Example of Cascading architecture .

The cascade architecture has interesting implications for the performance of the individual classifiers. Because the activation of each classifier depends entirely on the behavior of its predecessor, the false positive rate for an entire cascade is:

$$F = \prod_{i=1}^{K} f_i.$$

Similarly, the detection rate is:

$$D = \prod_{i=1}^{K} d_i.$$

Thus, to match the false positive rates typically achieved by other detectors, each classifier can get away with having surprisingly poor performance. For example, for a 32-stage cascade to achieve a false positive rate of 10^{-6}, each classifier need only achieve a false positive rate of about 65%. At the same time, however, each classifier needs to be exceptionally capable if it is to achieve adequate detection rates. For example, to achieve a detection rate of about 90%, each classifier in the aforementioned cascade needs to achieve a detection rate of approximately 99.7%.

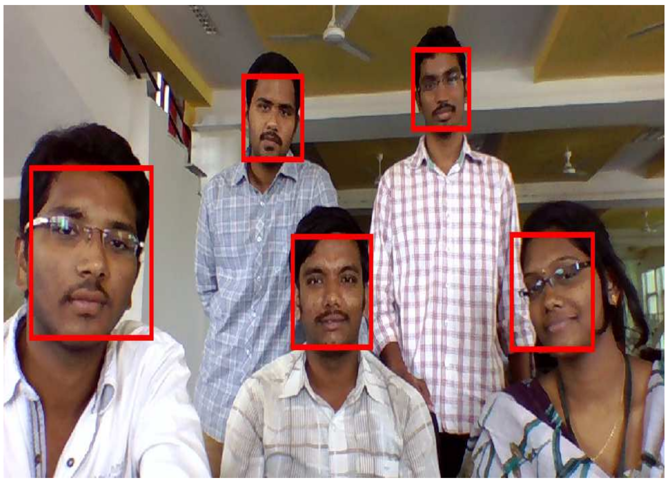

FIGURE 6: Face detection using viola-Jones algorithm.

IV. FEATURE EXTRACTION

After the face detection step, human-face patches are extracted from images. Directly using these patches for face recognition have some disadvantages, first, each patch usually contains over 1000 pixels, which are too large to build a robust recognition system. Second, face patches may be taken from different camera alignments, with different face expressions, illuminations, and may suffer from occlusion and clutter. To overcome these drawbacks, feature extractions are performed to do in-formation packing, dimension reduction, salience extraction, and noise cleaning. After this step, a face patch is usually transformed into a **vector with fixed dimension** or a set of **fiducial points and their corresponding locations**.

In pattern recognition and in image processing, **feature extraction** is a special form of dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be very redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called *feature extraction*. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input.

SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cubes.

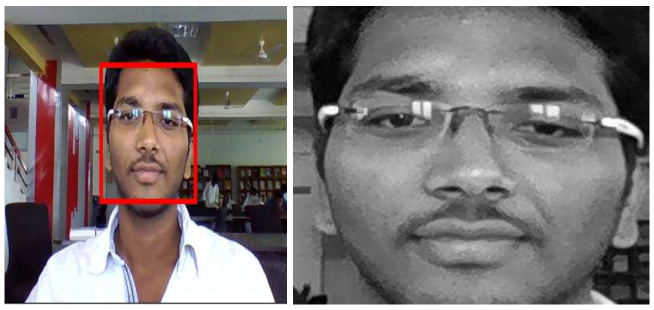

FIGURE 7: Segmented image which is a part of original image

ENHANCEMENT

Edge enhancement an image-processing filter that increases the contrast of shape borders to improve its acutance (apparent sharpness).

Shadow and highlight Enhancement an image processing technique to correct exposure.

V. FACE RECOGNITION

After formulizing the representation of each face, the last step is to recognize the identities of these faces. In order to achieve automatic recognition, a face database is required to build. For each person, several images are taken and their features are extracted and stored in the database. Then when an input face image comes in, we perform face detection and feature extraction, and compare its feature to each face class stored in the database. There have been many researches and algorithms pro-posed to deal with this classification problem, and we’ll discuss them in later sections. There are two general applications of face recognition, one is called **identification** and another one is called **verification**. Face identification means given a face image, we want the system to tell who he / she is or the most probable identification; while in face verification, given a face image and a guess of the identification, we want the system to tell true or false about the guess.

EIGEN FACES

Eigen faces is the name given to a set of Eigen vectors when they are used in the computer vision problem of human face recognition. The approach of using Eigen faces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The Eigen faces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

EIGEN FACE GENERATION

A 'set of Eigen faces' can be generated by performing a mathematical process called [[principal component analysis]] (PCA) on a large set of images depicting different human faces. Informally, Eigen faces can be considered a set of "standardized face ingredients", derived from [[statistical analysis]] of many pictures of faces. Any human face can be considered to be a combination of these standard faces. For example, one's face might be composed of the average face plus 10% from Eigen face 1. 55% from Eigen face 2, and even -3% from Eigen face 3. Remarkably, it does not take many Eigen faces combined together to achieve a fair approximation of most faces. Also, because a person's face is not recorded by a [[digital photograph]], but instead as just a list of values (one value for each Eigen face in the database used), much less space is taken for each person's face.

The Eigen faces that are created will appear as light and dark areas that are arranged in a specific pattern. This pattern is how different features of a face are singled out to be evaluated and scored. There will be a pattern to evaluate [[symmetry]], if there is any style of facial hair, where the hairline is, or evaluate the size of the nose or mouth. Other Eigen faces have patterns that are less simple to identify, and the image of the Eigen face may look very little like a face. The technique used in creating Eigen faces and using them for recognition is also used outside of facial recognition. This technique is also used for [[Graphology| handwriting analysis]], [[lip reading]], [[Speaker recognition| voice recognition]], [[sign language]]/hand [[gestures]] interpretation and [[medical imaging]] analysis. Therefore, some do not use the term Eigen face, but prefer to use 'Eigen image'.

PRACTICAL IMPLEMENTATION

To create a set of Eigen faces, one must:

1. Prepare a training set of face images. The pictures constituting the training set should have been taken under the same lighting conditions, and must be normalized to have the eyes and mouths aligned across all images. They must also be all re-sampled to a common [[pixel]] resolution ("r" × "c"). Each image is treated as one vector; simply by [[concatenation| concatenating]] the rows of pixels in the original image, resulting in a single row with "r" × "c" elements. For this implementation, it is assumed that all images of the training set are stored in a single [[Matrix (mathematics) |matrix]] "'T"', where each column of the matrix is an image.

2. Subtract the [[mean]]. The average image "'a"' has to be calculated and then subtracted from each original image in "'T"'.

3. Calculate the [[Eigenvectors and Eigen values| Eigenvectors and Eigen values]] of the [[covariance matrix]] "'S"'. Each eigenvector has the same dimensionality (number of components) as the original images, and thus can itself be seen as an image. The eigenvectors of this covariance matrix are therefore called Eigen faces. They are the directions in which the images differ from the mean image. Usually this will be a computationally expensive step (if at all possible), but the practical applicability of Eigen faces stems from the possibility to compute the eigenvectors of "'S"' efficiently, without ever computing "'S"' explicitly, as detailed below.

4. Choose the [[principal components]]. The 'D' x 'D' covariance matrix will result in "D" eigenvectors, each representing a direction in the "r" × "c"-dimensional image space. The eigenvectors (Eigen faces) with largest associated Eigen value are kept.

These Eigen faces can now be used to represent both existing and new faces: we can project a new (mean-subtracted) image on the Eigen faces and thereby record how that new face differs from the mean face. The Eigen values associated with each Eigen face represent how much the images in the training set vary from the mean image in that direction. We lose information by projecting the image on a subset of the eigenvectors, but we minimize this loss by keeping those Eigen faces with the largest Eigen values. For instance, if we are working with a 100 x 100 image, then we will obtain 10,000 eigenvectors.

In practical applications, most faces can typically be identified using a projection on between 100 and 150 Eigen faces, so that most of the 10,000 eigenvectors can be discarded.

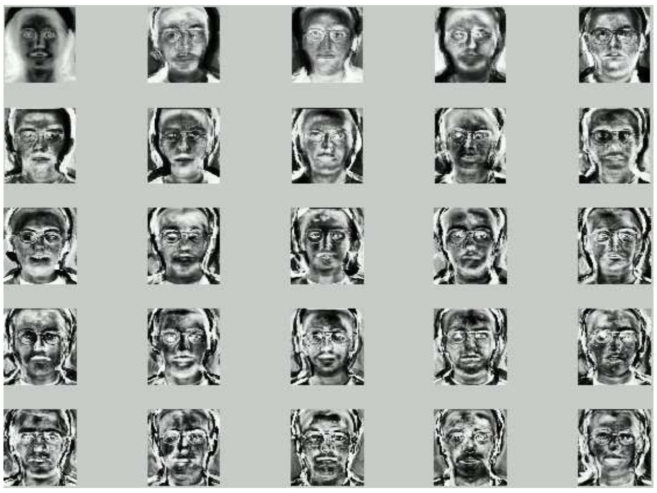
FIGURE 8: Sample Eigen faces

COMPUTING THE EIGENVECTORS

Performing PCA directly on the covariance matrix of the images is often computationally infeasible. If small, say 100 × 100, grayscale images are used, each image is a point in a 10,000-dimensional space and the covariance matrix **S** is a matrix of 10,000 × 10,000 = 10^8 elements. However the rank of the covariance matrix is limited by the number of training examples: if there are N training examples, there will be at most $N - 1$ eigenvectors with non-zero Eigen values. If the number of training examples is smaller than the dimensionality of the images, the principal components can be computed more easily as follows.

Let **T** be the matrix of preprocessed training examples, where each column contains one mean-subtracted image. The covariance matrix can then be computed as $\mathbf{S} = \mathbf{T}\mathbf{T}^T$ and the eigenvector decomposition of **S** is given by

$$\mathbf{S}\mathbf{v}_i = \mathbf{T}\mathbf{T}^T\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

However $\mathbf{T}\mathbf{T}^T$ is a large matrix, and if instead we take the Eigen value decomposition of

$$\mathbf{T}^T\mathbf{T}\mathbf{u}_i = \lambda_i\mathbf{u}_i$$

Then we notice that by pre-multiplying both sides of the equation with **T**, we obtain

$$\mathbf{T}\mathbf{T}^T\mathbf{T}\mathbf{u}_i = \lambda_i\mathbf{T}\mathbf{u}_i$$

Meaning that, if $\mathbf{u}_i$ is an eigenvector of $\mathbf{T}^T\mathbf{T}$, then $\mathbf{v}_i = \mathbf{T}\mathbf{u}_i$ is an eigenvector of **S**. If we have a training set of 300 images of 100 × 100 pixels, the matrix $\mathbf{T}^T\mathbf{T}$ is a 300 × 300 matrix, which is much more manageable than the 10,000 × 10,000 covariance matrix. Notice however that the resulting vectors $\mathbf{v}_i$ are not normalized; if normalization is required it should be applied as an extra step.

USE IN FACIAL RECOGNITION

Facial recognition was the source of motivation behind the creation of Eigen faces. For this use, Eigen faces have advantages over other techniques available, such as the system's speed and efficiency. As Eigen face is primarily a dimension reduction method, a system can represent many subjects with a relatively small set of data. As a face recognition system it is also fairly invariant to large reductions in image sizing, however it begins to fail considerably when the variation between the seen images and probe image is large.

To recognize faces, gallery images, those seen by the system, are saved as collections of weights describing the contribution each Eigen face has to that image. When a new face is presented to the system for classification, its own weights are found by projecting the image onto the collection of Eigen faces. This provides a set of weights describing the probe face. These weights are then classified against all weights in the gallery set to find the closest match. A nearest neighbor method is a simple approach for finding the [[Euclidean Distance]] between two vectors, where the minimum can be classified as the closest subject.

The weights of each gallery image only convey information describing that image, not that subject. An image of one subject under frontal lighting may have very different weights to those of the same subject under strong left lighting. This limits the application of such a system. Experiments in the original Eigen face paper presented the following results: an average of 96% with light variation, 85% with orientation variation, and 64% with size variation.

Various extensions have been made to the Eigen face method such [[Eigen features]]. This method combines [[facial metrics]] (measuring distance between facial features) with the Eigen face representation. Another method similar to the Eigen face technique is '[fisher face] s' which uses [[Linear discriminate analysis]]. This method for facial recognition is less sensitive to variation in lighting and pose of the face than using Eigen faces. Fisher face utilizes labeled data to retain more of the class specific information during the dimension reduction stage.

A further alternative to Eigen faces and fisher faces is the [[active appearance model]]. This approach uses an [[Active Shape Model]] to describe the outline of a face. By collecting many face outlines, [[Principal Component Analysis]] can be used to form a basis set of models which, encapsulate the variation of different faces.

Many modern approaches still use [[Principal component analysis]] as a means of dimension reduction or to form basis images for different modes of variation.

VI. ADVANTAGES AND APPLICATIONS

ADVANTAGES

SPEED AND SIMPLICITY

Eigen faces approach is superior in its near real time speed and reasonably simple implementation, where as feature based face recognition involves complex computations such as deformable templates and active contour models.

LEARNING CAPABILITY

Feature based face recognition systems are generally trained to optimize their parameters in a supervised manner. In the Eigen faces approach, training is done in an unsupervised manner. User selects a training set that represents the rest of the face images. Eigen faces are obtained from the training set members and feature vectors are formed.

FACE BACK GROUND

Eigen faces approach is very sensitive to face background, in case feature vectors are obtained by image additions and multiplications. Feature based face recognition algorithms are less sensitive to face background due to the localization of facial contours by deformable templates.

PRESENCE OF SMALL DETAILS

Feature based face recognition algorithms can suffer when some details are present on the face image such as dark glasses or beards. For a feature based face recognition system, it is quite impossible to extract the features that are related to the eyes when dark glasses are present on the face. Eigen faces approach excels in this aspect of face recognition. Small changes in face images such as glasses, beards or mustaches does not cause a decrease in the face recognition performance because the information that is present in the rest of the face image makes it enough to be classified correctly.

APPLICATIONS IN ROBOTICS

Robotics is an applied engineering science that refers to as combination of machine tool technology, electronics and computer science. With the advent of science and technology, robot finds its place in each and every field. Robots called humanoids, having features of that of humans are being created now a day. For robots to recognize face, a facial recognition system is needed. We can implement the face recognition module using the feature based Eigen face technique because it is widely implemented and well known for its simplicity and computational efficiency. The simple implementation of robotic facial recognition system is shown below

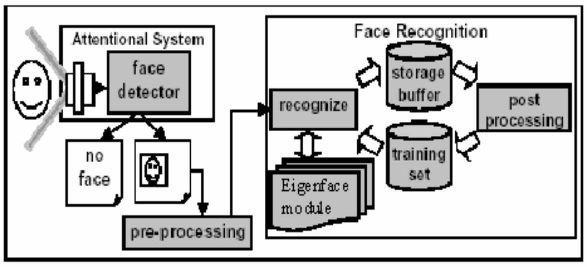

FIGURE 9: Simple Implementation of Robotic Facial Recognition System

IN CRIME INVESTIGATION

In present situation, the number of crimes committed per year is increasing tremendously. It is necessary to have a check over this crime rates. Most of the crimes are committed by changing one's identity, i.e., by changing facial features (like eyebrows, mustache etc.,). The crime investigation organizations can utilize the advantages of feature based Eigen face recognition technique to check the crimes. This can be achieved by maintaining a database of criminals, and by comparing (after Eigen face manipulation) the image of the culprit with the existing images in the database.

AIRPORT SECURITY

Airport and other transportation terminal security is not a new thing. People have long had to pass through metal detectors before they boarded a plane, been subject to questioning by security personnel, and restricted from entering "secure" areas. What has changed is the vigilance in which these security efforts are being applied.

The use of biometric identification, can enhance security efforts already underway at most airports and other major transportation hubs (seaports, train stations, etc.).This includes the identification of known terrorists before they get onto an airplane or into a secure location.

IDENTIFICATION SOLUTIONS

With regards to primary identification documents, (Passports, Driver's licenses, and ID Cards), the use of face recognition for identification programs has several advantages over other biometric technologies.

Leverage your existing identification infrastructure. This includes, using existing photo databases and the existing enrollment technology (e.g. cameras and capture stations); and increase the public's cooperation by using a process (taking a picture of one's face) that is already accepted and expected; Integrate with terrorist watch lists, including regional, national, and international "most-wanted" databases.

SCENE ANALYSIS AND SURVEILLANCE SOLUTIONS

This includes the ability to extract, categorize, and search non-facial imagery. For example, within the law enforcement application it allows you to capture, archive, and retrieve such identifying characteristics as tattoos, marks, or scars. It can also analyze scenes from either streaming or archived video, "looking" for out-of-the-ordinary occurrences, the presence of certain vehicles, specific faces, etc. This is beneficial and can save significant time and money to those individuals who spend hours, days, or weeks monitoring video streams (i.e. examining a bank's security in a criminal investigation).

VII. CONCLUSION

The human face serves as a provider of a number of social signals essential for interpersonal communication in our social life. Personality, gender, attractiveness and age can be seen from someone's face. This insists the need for face recognition. Feature based eigenface method provides a better alternative to existing methods. Feature based eigenfaces approach excels in its speed and simplicity and delivers good recognition performances under controlled conditions. Training is done in an unsupervised manner. In this paper we have specified the importance of face recognition using eigenvectors and we have highlighted the applications involving feature based eigenface approach, viz., robotics and crime investigation.

REFERENCES

BOOKS

[1] Fundamentals of Digital Image Processing by Chris Solomon and Toby Breckon.

[2] Digital Image Processing (3rd Edition) by Gonzalez and Woods.

WEBSITES

[1] http://www.mathworks.in/

[2] http://www.pages.drexel.edu/

[3] http://blog.cordiner.net/

[4] https://www.youtube.com/?gl=IN/

[5] http://stackoverflow.com/

[6] http://Wikipedia.com/

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.