Face Recognition Approaches and Implementation of Eigenfaces Based Recognition

Bhuwan Mehta and Rahul Gupta

April 18, 2009

1 Introduction

Face recognition is challenging because it is a real world problem. The human face is complex, natural object that tends not to have easily (automatically) identified edges and features. Because of this, it is difficult to develop a mathematical model of the face that can be used as prior knowledge when analyzing a particular image.

Applications of face recognition are widespread. Perhaps the most obvious is that of human computer interaction. One could make computers easier to use if when one simply sat down at a computer terminal, the computer could identify the user by name and automatically load personal preferences. This identification could even be useful in enhancing other technologies such as speech recognition, since if the computer can identify the individual who is speaking, the voice patterns being observed can be more accurately classified against the known individual's voice.

Human face recognition technology could also have uses in the security domain. Recognition of the face could be one of several mechanisms employed to identify an individual. Face recognition as a security measure has the advantage that it can be done quickly, perhaps even in real time, and does not require extensive equipment to implement. It also does not pose a particular inconvenience to the subject being identified, as is the case in retinal scans.

A final domain in which face recognition techniques could be useful is search engine technologies. In combination with face detection systems, one could enable users to search for specific people in images. This could be done by having the user provide an image of the person to be found.

In this report we have analyzed the different ways to recognize a face and compared their performances. Initially we have discussed the steps involved in feature-based face recognition. It evaluates the entire image as a whole. Then we discussed the Holistic matching approach in which the machine automatically determines which features to use. Then we have discussed the face recognition using Eigenfaces. Lastly we have explained the working of Fisherfaces.

2 Feature-Based Face Recognition

This face recognition system is based on local features. Interesting feature points in the face image are located by Gabor filters, which give us an automatic system that is not dependent on accurate detection of facial features. The feature points are typically located at positions with high information content (such as facial features), and at each of these positions we extract a feature vector consisting of Gabor coefficients. There are 3 processing steps involved in it.

- Segmentation: To eliminate the background
- Scaling: Performance decreases quickly if the scale is misjudged.
- Rotation: Symmetry operator to estimate head orientation [1]. Most of the time Scaling and Rotation can be controlled easily but Segmentation is necessary for higher accuracy.

2.1 Segmentation

It is dividing the images into (semantically meaningful) regions that appear to be the images of different surfaces. Reliable segmentation is possible with a priori info, which is not available in most of the cases. The two major approaches for doing segmentation are Histogram based segmentation and spatial coherence based Segmentation.

2.1.1 Histogram based segmentation

The image is converted into Gray-scale and then a binary mask is applied to it using a threshold. There are several ways for finding the threshold. One of the most popular way is using Histogramming. The Gray-level histogram gives the number of cells having a particular gray-level as shown in fig. 1.

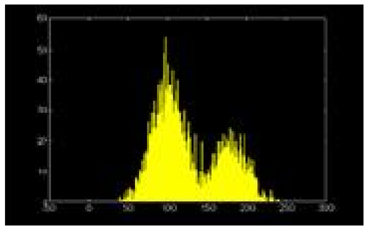

Figure 1: Gray Level Histogram

Ideally, object & background have constant different brightness inside their regions so we put a threshold between peak values in histogram as shown in fig. 2.

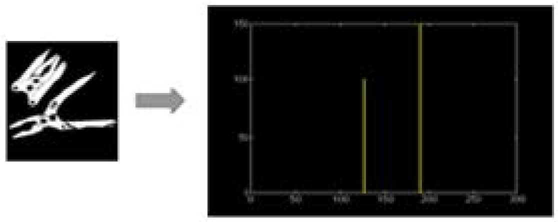

Figure 2: The left side picture has only 2 colours which are clealy visible in Histogram on left side

But in practice, brightness is not constant; there is some spread due to measurement noise, non-uniform illumination and non-uniform reflection from the surfaces. Such a case is shown in fig. 3.

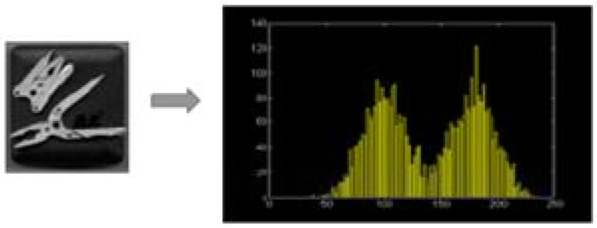

Figure 3: Effect of noise on Histogram

Some of the ways to choose the manual thresholding are:

1. **Histogram shape-based**: Analyze peaks, valleys, curvatures of smoothed histogram.

2. **Clustering-based**: Iteratively, finding a threshold, clustering based this threshold.

3. **Entropy-based**: Choosing a threshold which max the info content in histogram.

4. **Attribute-based**: Similarities between edge, etc of image & its thresholded version.

5. **Spatial thresholding (higher order stats)**: Threshold selection on higher order statistics of spatial neighbours.

6. **Local thresholding**: Finding threshold values at each neighborhood using local stats.

In real time cases, its difficult to do manual thresholding so we go for automatic thresholding. Some of the ways to do automatic thresholding are:

1. **P-tile method**: It uses the a priori knowledge about the size of the object. For eg: Assume an object with size p. Choose the threshold such that %p of the overall histogram is determined as shown in fig. 4. It has got a very limited use as we not always have a priori information about the object.

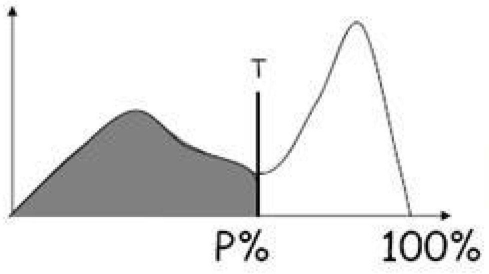

Figure 4: Histogram divided into 2 parts of P and 100-P percent

2. **Mode method**: First the "peaks" and "valleys" of the histogram are located and then threshold is set to the pixel value of the "valley". Local peaks are ignored and we choose peaks at a distance;then the valley between those peaks is found. Then we maximize "peakiness" (difference btw peaks & valleys) to find the threshold as valley as shown in fig. 5.

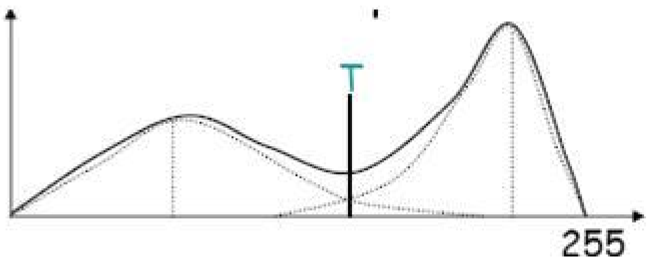

Figure 5: Calculation of threshold using mode method

3. **Iterative threshold selection**: Starting with an approximate threshold, refine it iteratively, taking into account some goodness measure e.g. T = $(\mu_1+\mu_2)$ / 2 where μ_i is the mean gray value of previous segmented region i.

4. **Adaptive Thresholding**: In the case of an uneven illumination, the global threshold has no use. So one of the approaches is to divide an image into $m \times m$ sub-images and determine a threshold for each sub-image as shown in fig. 6.

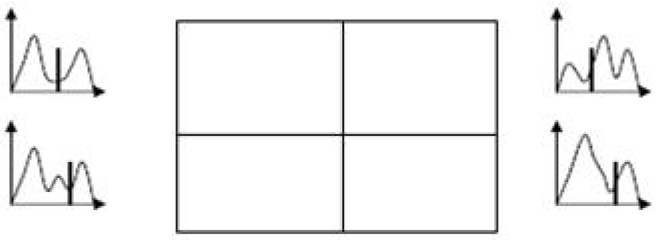

Figure 6: Calculation of threshold using adaptive thresholding

5. **Double Thresholding**: Starting from a conservative initial threshold T1, we determine the "core" parts of the object. Continuing from this core part, we grow this object by including neighboring pixels which are between T1 and T2 as shown in fig. 7.

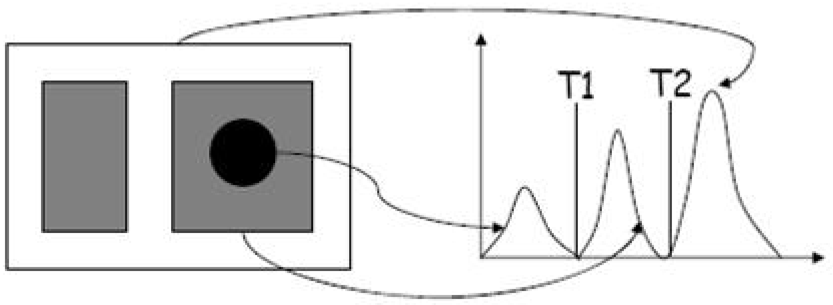

Figure 7: Calculation of threshold using double thresholding

2.1.2 Spatial Coherence

Histogram-based methods totally neglect the dependency between neighboring pixels Neglecting this dependency may cause "salt-n-pepper" noise in the resulting binary image. If spatial coherence between pixels is taken into account, such noise can be eliminated by some preprocessing. Such an approach decreases the error-rate but obviously does not guarantee being error-free. Dependency between neighboring pixels or regions could be represented in various ways. Regions can be represented in alternative forms such as:

1. Array representations : masks.
2. Hierarchical representations : quad-trees.
3. Symbolic representations : bounding box, centroid, moments, Euler number.

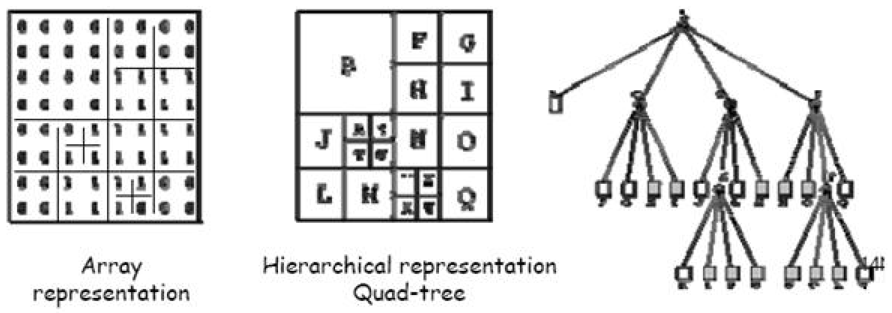

Figure 8: Representation of different regions in an image

The output of any segmentation method can be improved by simply merging similar neighboring regions together. Similarity can be measured by:

1. A simple threshold.
2. A geometrical attribute, such as "common boundary length".
3. More sophisticated methods based on statistics .

Similarly, rather than merging, splitting can be required due to geometrical attributes. A general region merge algorithm can be described as follows: Beginning from an initial segmentation, an initial RAG is prepared; for each region it is checked whether its neighboring regions are "similar", if so, regions are merged & RAG is modified. For "region similarity":

1. Compare their mean intensities: check with a predetermined threshold.
2. Compare their statistical distributions: check whether such a merge represents "observed" values better.
3. Check "weakness" of the common boundary: weak boundary: intensities on two sides differ less than a threshold.

Merge two regions if W/S $> \tau$, where W=length of weak boundary

1. S = min{S1,S2} : minimum of two boundaries.
2. S : common boundary.

2.2 Eyes Extraction

The first feature we extract is eyes. Human eyes have many characteristics. On the one hand, eyes are the darker region in human face, and have little gray value in gray graph. On the other hand, gray changes greatly around eyes region in which grads value of each point is also much big. Because eyes and skin have many differences, we implement boundary detection in the candidate facial region and project horizontally. Then we can basically identify the eyes' horizontal position in A and B as shown in fig. 9. After, project vertically above A and B, find the first peak position as C and D. Eyes' outline and left and right canthus are located in two areas made up of A,C and B,D. The mean of these two regions are considered as the position of pupils [2].

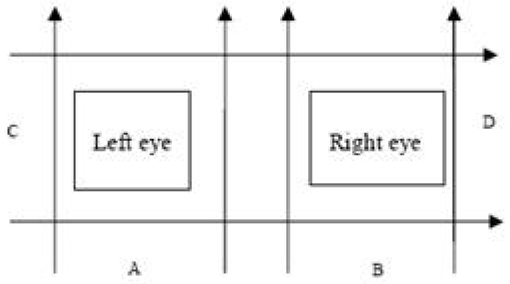

Figure 9: Eyes Extraction

2.3 Mouth Extraction

In aspect of mouth extraction, we take lip color into account. In the bottom of face, we can consider the regions which satisfy the condition below as mouth [2]:

$$\Theta = across\left(\frac{.5 \times (2R - G - B)}{\sqrt{(R-G)(R-B) + (G-B)}}\right)$$

2.4 Nose Extraction

After locating the eyes and mouth, we can identify nose according to some transcendent knowledge. If the distance between two pupils is seen as 1, the distance from nose to the middle of eyes is from 0.7 to 1. Searching the darker region around this we can probably find the position of nostril, and then find the greatest luminance point above some certain areas of two nostrils as the top of nose [2].

3 Holistic Matching

Feature matching can exploit the efficiency found in manually tuning features for a particular training image set. However, these rules have a severe limitation on the type of object classes that can be found by the image retrieval system. Objects greatly different than those for which the system was designed will not be retrieved accurately or efficiently. For example, features tuned to automatically find a human face will probably be useless for retrieving an image of a car.

An alternative to feature matching is the "Holistic Matching" approach in which the machine automatically determines which features to use. The representation of the system is at the signal level instead of at the knowledge (e.g., shape) level. In this type of framework, a training phase finds salient features to use in the subsequent recognition phase of the system. These types of approaches can deal directly with complex, real-world images [3], [4], [5] because the system is general and adaptive.

The efficient selection of good features, however, is an important issue to consider [6]. In this type of matching, we project the higher dimension image vector onto a lower dimension feature space constructed using the training data. Various dimensionality reduction techniques are used for the same. In the subsequent subsections, we will be discussing the following two major holistic matching/recognition techniques based on different dimensionality reduction techniques:

1. Eigenfaces technique based on Principle Component Analysis(PCA).

2. Fisherfacestechnique based on Linear DiscriminantAnalysis(LDA).

3.1 Eigenfaces Method

Introduction

The eigenface method for human face recognition is remarkably clean and simple. The basic concept behind the eigenface method is information reduction. When one evaluates even a small image, there is an incredible amount of information present. From all the possible things that could be represented in a given image, pictures of things that look like faces clearly represent a small portion of this image space. Because of this, we seek a method to break down pictures that will be better equipped to represent face images rather than images in general. To do this, we generate "base-faces" and then represent any image being analyzed by the system as a linear combination of these base faces. The generation of these base faces is discussed in detail later when we look at the mathematical basis for this face recognition method.

Once the base faces have been chosen we have essentially reduced the complexity of the problem from one of image analysis to a standard classification problem. Each face that we wish to classify can be projected into face-space and then analyzed as a vector. A k-nearest-neighbor approach, a neural network, or even a simply Euclidian distance measure can be used for classification. The problem is straightforward at this point.

Let us take an in-depth look at how the eigenface method works. The technique can be broken down into the following components:

1. Generate the eigenfaces.
2. Project training data into face-space to be used with a predetermined classification method.
3. Evaluate a projected test element by projecting it into face space and comparing to training data.

Generation of Eigenfaces

Before any work can be done to generate the eigenfaces, sample faces are needed. These images will be used as examples of what an image in face-space looks like. These images do not necessarily need to be images of the people the system will later be used to identify (though it can help); however the image should represent variations one would expect to see in the data on which the system is expected to be used, such as head tilt/angle, a variety of shading conditions,etc. Ideally these images should contain pictures of faces at close to the same scale, although this can be accomplished through preprocessing if necessary. It is required that all of the images being used in the system, both sample and test images, be of the same size. The resulting eigenfaces will also be of this same size once they have been calculated.

It should be noted that it is assumed that all images being dealt with are grayscale images, with pixel intensity values ranging from 0 to 255. While it is possible to generate color eigenfaces, we will not be dealing with it here.

Suppose, there are K images in our data set. Each sample image will be referred to as X_i where n indicates that we are dealing with i^{th} sample image (1 $\leq$ i $\leq$ K). Each X_i is a collumn vector. Generally images are thought of as pixels, each having (x,y) coordinates with (0,0) being at the upper left corner (or one could think of an image as a matrix with y rows and x columns). Converting this to a column form is a matter of convenience, it can be done in either column or row major form, so long as it is done consistently for all sample images it will not affect the outcome. The size of the resulting X_i column vector will depend on the size of the sample images. If the sample images are x pixels across and y pixels tall, the column vector will be of size (x*y) $\times$ 1. These original image sizes must be remembered if one wishes to view the resulting eigenfaces, or projections of test images into face-space. This is to allow a normal image to be constructed from a column vector of image pixels.

Let $\overline{X}$ be the mean of all X_i (1 $\leq$ i $\leq$ K). This is the step to calculate an avergae face of the database. If one were to reinterpret the vector as a normal image, it would appear as one might expect, as shown in Figure 10

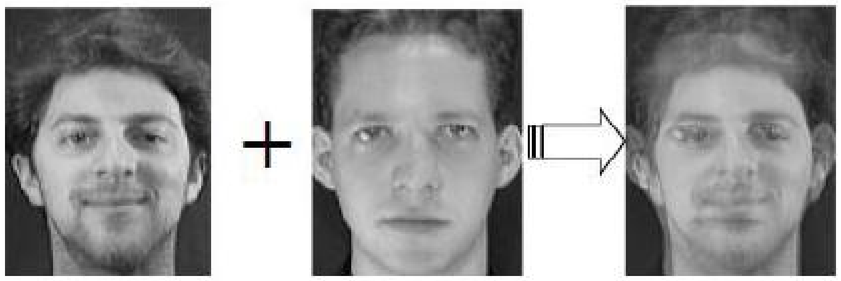

Figure 10: In this figure, averaging of two faces is shown

.

The next step is to calculate difference faces U_i such that $U_i = X_i - \overline{X}$ and form a matrix U, such that $U = [U_1 U_2 .....U_K]$. Our goal now is to generate the eigenfaces which is done by calculating the eigenvectors of the covariance matrix UU^T.This cannot be done directly as the size of UU^T is (x*y)*(x*y) which is very large. Clearly, doing these calculations on a resulting matrix of this size is going to be taxing on all but the most specialized, advance hardware. To avoid this problem, a trick from linear algebra is applied. The eigenvectors of the UU^T matrix can actually be found by considering linear combinations of the eigenvectors of the $U^T U$ matrix. This is extremely usefully when one realizes that the size of the $U^T U$ matrix is K × K. For practically all real world situations K $<<$ (x*y). The eigenvectors w_j of this matrix can be readily found through the following formula:

$$w_j = \frac{\sum_{l=1}^{K} U_l E_{lj}}{\sqrt{\lambda_j}}$$

where $E_{lj}$is the l^{th}calue of the j^{th}eigenvector of $U^T U$ and $\lambda_j$is the corresponding eigenvalue of $w_j$and E_j. The linear algebra part of this trick is given below:

Let the eigenvectors of $U^T U$ be E_j $(1 \le j \le K)$ and the corresponding eigenvalues be λ_j. Hence, we can write

$$U^T U E_j = \lambda_j E_j$$

Multiplying both the sides by U,

$$(UU^T)UE_j = \lambda_j (UE_j)$$

Thus, $w_j = UE_j$ is the j^{th} eigenvector of $UU^T$with corresponding eigenvalue λ_j.

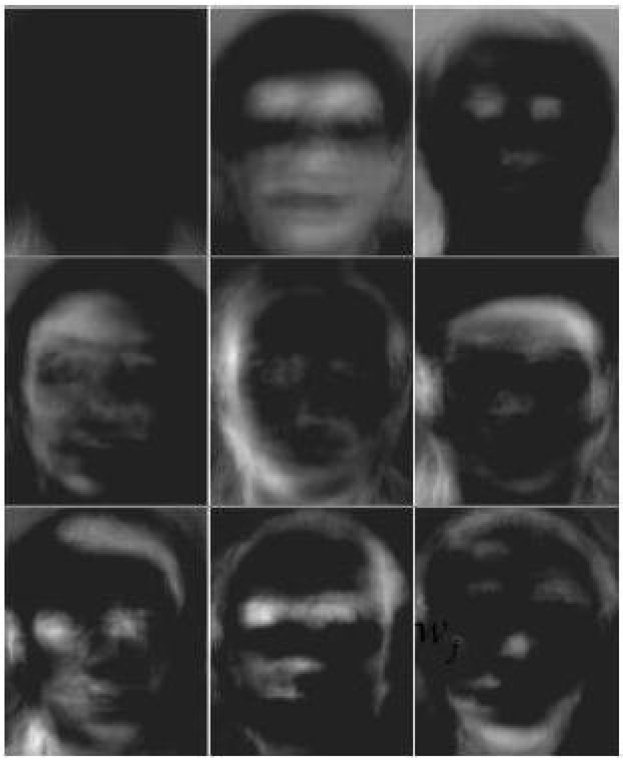

Figure 11: shows nine eigenfaces generated from a face database. Each of these images represents the image interpretation of one of the w_j calculated previously. As one can see, the images appear almost as ghosts, each with a different portion of the face accented.

Here we employ the fact that the eigenvalues for the $UU^T$and U^TU are the same (though if we were going to calculate all of the eigenvalues of the UU^T matrix, we could get more values, the eigenvectors of the U^TU only represent the most important subset of the eigenvalues of the UU^T matrix).

Significance of Eigenvectors/Eigenvalues

Now that we have generated the eigenvectors for the covariance matrix of the differences faces, let us evaluate what we have actually created. An eigenvector whose corresponding eigenvalueis of greatest magnitude represents the direction of greatest variance in a covariance matrix. The eigenvector corresponding to the second largest eigenvalue represents the direction of greatest variance in the covariance that is perpendicular to the first eigenvector. The third eigenvector represents the direction of greatest variance such that the eigenvector is perpendicular to the first two. [7] This continues for all of the eigenvectors, as we have ordered them by the magnitude of their corresponding eigenvalues,

$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_K$.

The intuitive way of thinking of this process is that the first eigenvector has the most discriminating power for the vectors that make up the columns of the covariance matrix, the second one has the second most discriminating power, the third one has the third most, and so on... Furthermore, the eigenvectors are perpendicular, so what we are essentially doing is creating a coordinate system (which we will refer to as face-space) that has the most possible discriminating power for the vectors we used in creating it. This means that each one of our eigenfaces actually represents an axis in our face-space. When an image is represented in face space, it is really stored as a vector of coefficients that indicate how much each eigenface is to contribute to the final image. When these individual contributions are added together, the original image is formed (assuming the eigenfaces form a perfect basis for face-space). Projection of an image into face space will be discussed in the next subsection.

Since the eigenfaces have been ranked by their discriminating ability, it is not necessary to use all of the eigenfaces generated in classification. It is possible to only consider a small subset of the best eigenvectors and still maintain discriminating power. When the eigenfaces are ranked by magnitude of their corresponding eigenvalues, only the top k eigenfaces need to be used. An experimental evalaution of the number of optimum eigenfaces can also be used and an example is shown in Fig. 12, where the optimum number of Eigenfaces to be used comes out to be around 40 on a particular database. The perofrmance of the recognition system as can be seen saturates after a particular number of eigenfaces used.

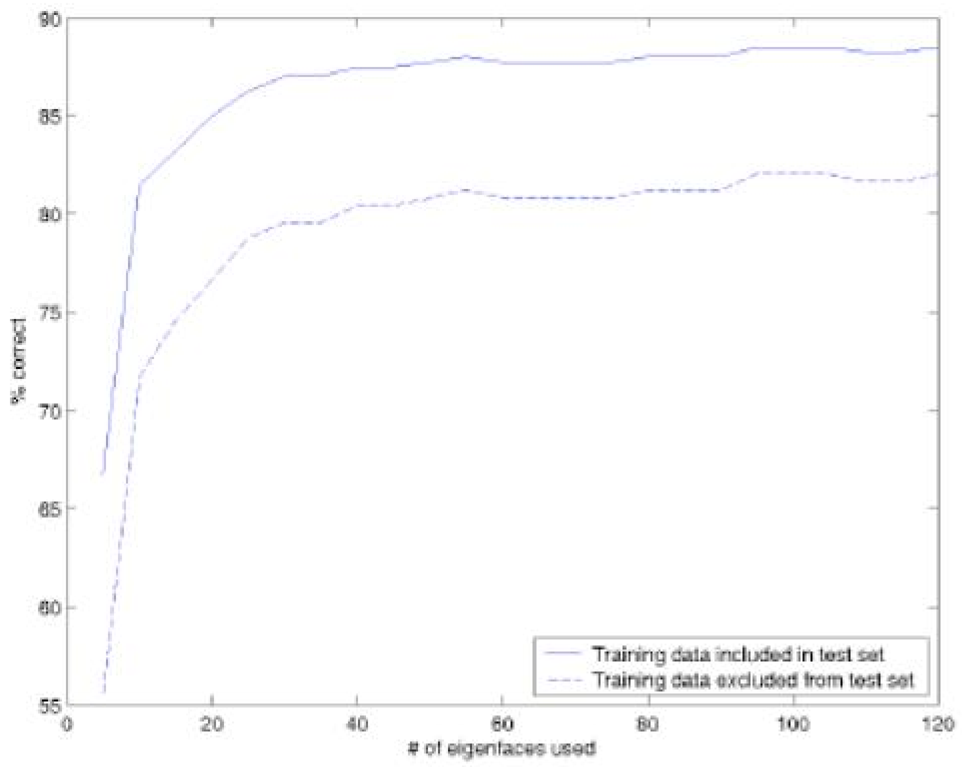

Figure 12: The number of eigenfacesused vs. success in classification for a database. For this database, the optimum number of eigenfacescomes out to be around 40.

Projecting Images into Face Space

Any image Z can be projected into face space by using the following formula:

$$W' = W^T(Z - \overline{X})$$

Put simply, the vector of weights is found by multiplying the transpose of the matrix W(W isformed by letting each eigenfaceform a column of the matrix) by a vector that is found by subtracting the average face image ($\overline{X}$, a column vector) from a sample or test image (Z, a column vector). Note that, Z could be any training or test image converted into column vector. Now that a method of projecting images into face space has been defined, the problem of face recognition becomes one of everyday pattern recognition[4]. One such pattern recognition technique using Euclidean distance model is discussed briefly later.

Reconstruction of image from Eigenfaces

The eigenfacerecognition method was derived from work on analyzing the loss of information by representing faces through weights of basis-faces. The basis faces were found through principle component analysis techniques. Because of this, it is possible to reconstruct an original face image from the known eigenfaceweights.

$$Z' = WU^T + \overline{X}$$

Note that, the reconstructed face vector Z' is still in column vector form after this reconstruction, and must be converted into a normal image prior to viewing.

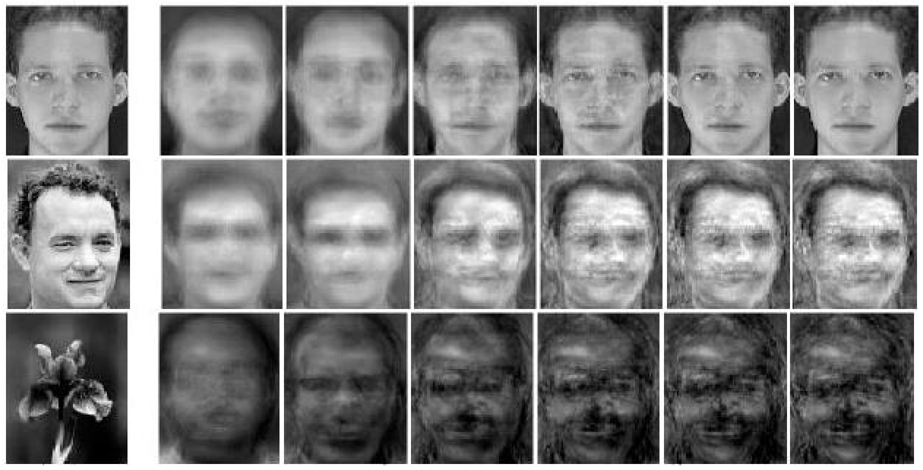

Figure 13: Images and their reconstruction from face-space representations; original image on far left; 5, 10, 50, 100, 150, 200 eigenfacereconstructions from left to right. The three rows are the cases of known subject, unknown subject and not a face respectively.

Fig. 13 shows several images and their reconstruction from their weights in face space. The first face is a training image used in the calculation of the eigenfaces. The second image is a face that is of an individual that was not present in any of the images used to calculate the eigenfaces. This image, once again, does not exactly project into face space. This is compounded by the fact that this face image has a light background where as the training images all had black backgrounds. It is easy to forget that the eigenface derivations take into account the entire image rather than just the portion that contains the face. This means that accuracy in the reconstruction of the face can be lost as a balance is formed between favoring representing the face and representing the background. The last image is not a face image and is therefore represented poorly in face space. Because of this, the reconstructed image is not recognizable due to loss of information. It should be noted that for all of face images, the accuracy of the reconstruction improves as the number of eigenfaces being used to represent the image increases. As one can see for the flower image this does not appear to be the case. There is no significant improvement in the representation for all number of eigenfaces used. That is because of this fact that non-face images live at such extreme points in face space that we are able to classify images as non-face.

Classifying Faces

All the training sample images have been projected onto face space. Form a cluster of all the images belonging to the same class (images of same user). Now, if a test image is fed as an input, we should first project the test image onto face space by the procedure discussed earlier. There can be 3 outputs possible for a test image as given below,

1. **Not a Face**: It can be classified as 'not a face'. The interpretation of "not a face" in the eigenfacesystem is that the projection of an image into face-space not only does not yield a vector close to any know clusters formed by a single individuals, but it is also significantly far away from all clusters.

2. **Unknown Face**: The "unknown face" classification indicates that a test image contains a face, but the face is not recognized by the classification system. This classification is used to indicate that the face presented to the classification system does not closely match any face images on which it has been trained.
   If one wishes the classification system to learn to recognize new individuals, the face space vectors of unknown faces could be recorded and then unsupervised clustering methods could be employed to attempt to recognize unknown recurring individuals.

3. **Recognized Face**: The "recognized face" classification indicates that the face recognition system was able to find a known individual that was sufficiently similar to the one presented in the test image. Along with the "recognized" classification, the system then provides the identity of the individual in the test image.

Image Classification Technique

Since projection into face space allows us to consider images being tested to be vectors in a reasonably low dimensional space, one could choose among several commonly used pattern recognition techniques like Euclidean Distance Classifier, k-Nearest Neighbour classifier, neural networks, etc. We will describe the euclidean distance classifier in brief here.

The Euclidian distance classifier is an efficient classification technique in areas where clusters of points representing the different entities to be classified are spaced far apart in feature space. A cluster represents all the images of a subject in the database. An average of all the points corresponding to the lcuster is calculated and plotted as a point in the face space as a point corresponding to the cluster/sunbject.Given an image, it is first projected into the face space as described in the preceding sections and the image is classified based on to whichever cluster average the image vector is nearest to.

Salient Features of EigenfacesMethod

1. Run-time performance is very good.
2. Construction is computationally intense, but need to be done infrequently.
3. Need to rebuild the eigenspace if adding a new person. Although not a necessary step, but in the case of a sercurity based application, such an action is generally preferred.
4. Starts to break down when there are too many classes as Eigenfaces method might be a good method for representation of images but it is not so effective when it comes to discriminating among the classes. It would be discussed in more detail later.
5. Retains unwanted variations due to lighting and facial expression.

Implementation of Eigenfaces Method on Matlab and Results.

The Matlab code for the implementation of Eigenfaces method is geven in Appendix A. An example has also been shown with regard to the reconstruction of image using the code.

3.2 Fisherfaces Method

Although the Eigenface projection is well-suited to object representation, the features produced are not necessarily good for discriminating among classes defined by the set of samples. The Eigenface method is also based on linearly projecting the image space to a low dimensional feature space.

Let W be a projection matrix that projects a vector into the subspace. Vector Z=WY is a new feature vector from samples of c classes with class means m_i, i= 1, 2, ..., c and the grand sample mean M for all the samples in all the classes. In PCA, the projection W^* is chosen to maximize the determinant of the total scatter matrix of the projected samples, i.e.,

$$W^* = arg\, max_W \sum_{j=1}^{N} (Y_j - M)(Y_j - M)^T$$

where N is the total number of images in the database. Thus, the Eigenface method, which uses principal components analysis (PCA) for dimensionality reduction, yields projection directions that maximize the total scatter across all classes, i.e., across all images of all faces. In choosing the projection which maximizes total scatter, PCA retains unwanted variations due to lighting and facial expression. As illustrated in Figs. 14 and stated by Moses et al., "the variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity" [8]. Thus, while the PCA projections are optimal for reconstruction from a low dimensional basis, they may not be optimal from a discrimination standpoint.

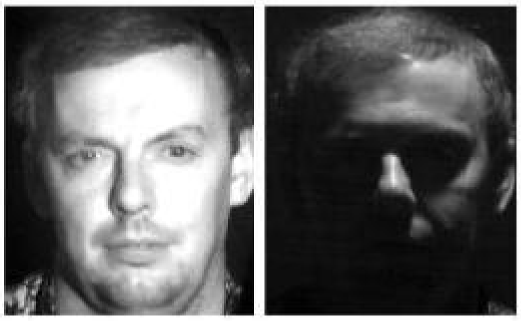

Figure 14: The variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity. Here is an example of such a case, when the same person seen under different lighting conditions can appear dramatically different.

Since. the learning set is labeled, it makes sense to use this information to build a more reliable method for reducing the dimensionality of the feature space. Here we argue that using class specific linear methods for dimensionality reduction and simple classifiers in the reduced feature space, one may get better recognition rates than with either the Linear Subspace method or the Eigenface method. Fisher's Linear Discriminant (FLD) [9] is an example of a class specific method, in the sense that it tries to "shape" the scatter in order to make it more reliable for classification. This method selects W in in such a way that the ratio of the between-class scatter and the within class scatter is maximized, where we can define a within class scatter as,

$$S_w = \sum_{i=1}^{c} \sum_{j=1}^{n_i} (Y_j - M_i)(Y_j - M_i)^T$$

where $n_i$is the number of samples in class i. Also, the between class scattered matrix is defined as,

$$S_b = \sum_{i=1}^{c} (M_i - M)(M_i - M)^T$$

In discriminant analysis, we want to determine the projection matrix W that maximizes the ratio $\frac{det(S_b)}{det(S_w)}$ In other words, we want to maximize the between-class scatter while minimizing the within-class scatter. It has been proven that this ratio is maximized when the column vectors of projection matrix W are the eigenvectors of $S_w^{-1}S_b$, associated with the largest eigenvalues. These Eigenvectors are known as Fisherfaces. Thus, Fisherfaces method tries to project away variations in lighting and facial expression using the knowledge of class while maintaining discriminability. Rest of the discussion is same as the one carried out in the case of Eigenfaces.

Conclusions

In this paper, we have discussed primarily two types of fae matching techniques namely, feature based matching and holistic matching. Feature Based Face Recognition is computationally less intense and retains only the information of the object by eleminating the background but such type of recognition systems are very object specific.The feature based recognition systems exploit the efficiency found in manually tuning features for a particular training image set. However, these rules have a severe limitation on the type of object classes that can be found by the image retrieval system.

Then, we discussed Holistic matching in which the generation and matching of the features is done by the computer itself. Under Holistic matching, we discussed two techniques Eigenfaces based matching and Fisherfaces based matching. We discussed the generation of Eigenfaces and Fisherfaces and discussed other isues related to them. We also learnt that Eigenfaces based recognition is very much vulnerable to variations in light, facial expressions, etc and found out that fisherfaces tend to project away such unneccesary variations. In the end, we also implemented the Eigenfaces based face recognition on MATLAB and were satisfied with its working.

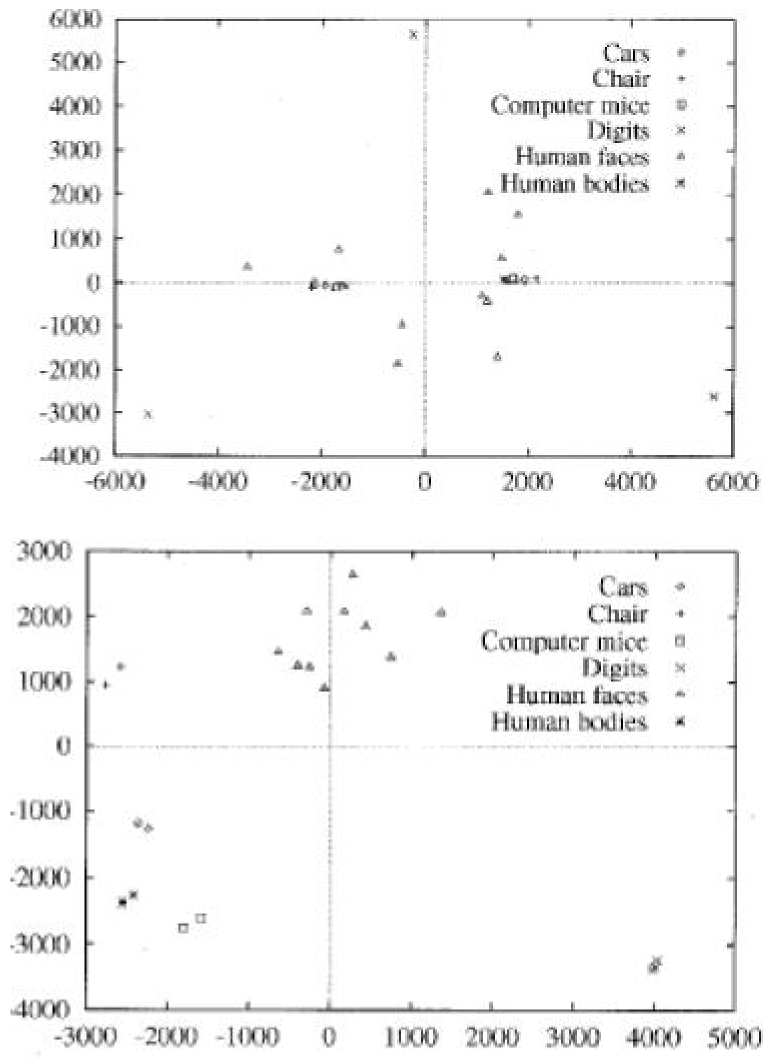

Figure 15: Distribution of some samples using the best two features in the PCA and the FDA based spaces respectively. In the FDA based subspace, objects of the same class are clustered much more tightly than in the PCA based space

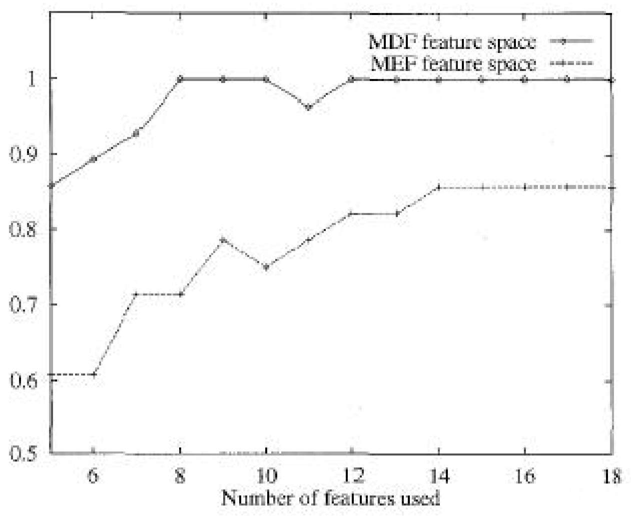

Figure 16: The performance of the system for different numbers of MEF(Eigenfaces) and MDF(Fisherfaces) features, respectively. The number of features from the subspace used was varied to show how the MDF subspace outperforms the MEF subspace. 95% of the variance for the MDF subspace was attained when 15 features were used; 95% of variance for the MEF subspace did not occur until 37 features were used. Using 95% of the MEF variance resulted in an 89% recognition rate, and that rate was not improved using more features

References

[1] Christopher J. Parsons and Mark S. Nixon, “Introducing Focus in the Generalized Symmetry Operator”, IEEE Signal Processing Letters, Vol. 6, No. 3, March 1999.

[2] Qian ZHANG and Zhi-Jing LIU, “Face Detection Based on Complexional Segmentation and Feature Extraction”, Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP’06).

[3] H. Murase and S.K. Nayar, ”Illumination Planning for Object Recognition in Structured Environments,” Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, pp. 31-38, Seattle, June 1994

[4] M. Turk and A. Pentland, ”Eigenfaces for Recognition,” J. Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86,1991.

[5] J. Weng, N. Ahuja, and T.S. Huang, ”Learning Recognition and Segmentation Using the Cresceptron,” Proc. Int’l Conf Computer Vision, pp. 121-128, Berlin, May 1993.

[6] D. Beymer and T. Poggio, “Face Recognition from One Example View,” Proc. Int’l Conf. Computer Vision, pp. 500-507,1995.

[7] E. Gose, R. Johnsonbaugh, S. Jost, “Pattern Recognition and Image Analysis”, Prentice Hall, Inc., Upper Saddle River, NJ, 1996.

[8] Y. Moses, Y. Adini, and S. Ullman, “Face Recognition: The Problem of Compensating for Changes in Illumination Direction,” European Conf. Computer Vision, 1994, pp. 286-296.

[9] R.A. Fisher, “The Use of Multiple Measures in Taxonomic Problems,” Ann. Eugenics, vol. 7, pp. 179-188, 1936.

Appendix - A

```
%Sample data is less so accuracy may not be very high. But if the images
%from the database are fed as testimg then the results are very accurate
%and we get exactly the same image.
%test1.jpg is used as an example to show that it can also try to generate
%an image not previously present in database.

%image names must be: 1.jpg, 2.jpg......
%numimg = the number of last image
numimg=9;
%adjust row and col dimensions to match with image resolution i.i same ratio
and keep them
%below 100. the resolution 600x800 means 800 rows and 600 col.
row=80;
col=60;

%images are read
for i=1:numimg
s=['a' num2str(i) '=imread(''' num2str(i) '.jpg'');'];
eval(s)
end

%resize
for i=1:numimg
s=['a' num2str(i) '=imresize(a' num2str(i) ',[row,col]);'];
eval(s)
end

%convert to gray scale if not previously converted
% for i=1:numimg
% s=['imwrite(rgb2gray(a' num2str(i) '),''' num2str(i) '.jpg'');'];
% eval(s)
% end



%image converted to col. vector
for t=1:numimg
    m=1;
    for i=1:col
      for j=1:row
            s=['A' num2str(t) '(m,1)=a' num2str(t) '(j,i);'];
            eval(s);
            m=m+1;
      end
    end
end

%Avg vector O is created
O=zeros(row*col,1);
for i=1:numimg
    s=['A' num2str(i) '=double(A' num2str(i) ');'];
    eval(s);
    s=['O=O+A' num2str(i) ';'];
    eval(s);
```

```
end
O=O/9;


a=1;b=1;
for i=1:row*col
    avg(a,b)=O(i,1);
    if (rem(i,row)==0)
        b=b+1;
        a=1;
    else
        a=a+1;
    end;
end
%Now To create overall average image just type imshow(uint8(avg))

%Create single matrix An containing all the images as its columns
An=A1;
for i=2:numimg
    s=['An=cat(2,An,A' num2str(i) ');'];
    eval(s);
end


for i=1:numimg
On(:,i)=An(:,i)-O;
end

cov=On*On';

cov2=On'*On;

%Calculate eigen values for cov2 (dimension=numimgxnumimg) since its more
%difficult to calculate eigen values for cov1 (dimension=row*col x row*col)
[V,D]=eig(cov2);

%u is the matrix containing eigenfaces as its columns.
sum=0;
for k=1:numimg
    for l=1:numimg
        sum=sum+On(:,l).*V(l,k);
    end
    u(:,k)= sum/sqrt(D(k,k));
    sum=0;
end

%utilda represents the vector of weights that result from a projection into
facespace.
for i=1:numimg
    utilda(i,:)=(u(:,i))'*(An(:,i)-O);
end

%Atilda is what we get by applying weights to eigenfaces
sum=zeros(row*col,1);
for i=1:numimg
    sum(:,1) = sum(:,1)+u(:,i)*utilda(i);
end
```

```
Atilda(:,1) = sum + O;


% a=1;b=1;
% for i=1:row*col
%     final(a,b)=Atilda(i,1);
%     if (rem(i,row)==0)
%         b=b+1;
%         a=1;
%     else
%         a=a+1;
%     end;
% end



%testimg is the test image
testimg=imread('test1.jpg');
testimg=imresize(testimg,[row,col]);

%use below command if image is not in gray scale
%testimg=rgb2gray(testimg);


m=1;
for i=1:col
    for j=1:row
        Atest(m,1)=testimg(j,i);
        m=m+1;
    end
end
Atest=double(Atest);

%Calculate the weights for the test image
for i=1:numimg
utildatest(i,:)=(u(:,i))'*(Atest(:,1)-O);
end

%Calculate the generated image as a col vector
sum=zeros(row*col,1);
for i=1:numimg
    sum(:,1) = sum(:,1)+u(:,i)*utildatest(i);
end
    Atildatest(:,1) = sum + O;

%Convert generated img to row x col format
a=1;b=1;
for i=1:row*col
    finalimg(a,b)=Atildatest(i,1);
    if (rem(i,row)==0)
        b=b+1;
        a=1;
    else
        a=a+1;
    end
end
imshow(uint8(finalimg))
```