A Comparison of Face Recognition Methods

Final Project Report

A Combined Project for
CSCI 5561
Computer Vision
&
CSCI 5512
Artificial Intelligence II

Created by

Christopher Burns
burn0275@umn.edu
ID 1855624

Ryan Morlok
morl0041@umn.edu
ID 2279064

May 8, 2003

Introduction

The Problem

Human face recognition is a difficult problem in computer vision. Early artificial vision experiments tended to center around toy problems in which the world being observed was carefully controlled and constructed. Perhaps boxes in the shapes of regular polygons were identified, or simple objects such as a scissors were used. In most cases the background of the image was carefully controlled to provide excellent contrast between objects being analyzed and the surrounding world. Clearly face recognition does not fall into this category of problems.

Face recognition is challenging because it is a real world problem. The human face is a 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. It has the disadvantage, however, that it is not a foolproof method of authentication, since human face appearance is subject to various sporadic changes on a day-to-day basis (shaving, hair style, acne, etc…), as well gradual changes over time (aging). Because of this, face recognition is perhaps best used as an augmentation for other identification techniques.

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 either having the user provide an image of the person to be found, or simply providing the name of the person for well-known individuals. A specific application of this technology is criminal mug shot databases. This environment is perfectly suited for automated face recognition since all poses are standardized and lighting and scale are held constant. Clearly, this type of technology could extend online searches beyond the textual clues that are typically used when indexing information.

Structure of the Project

Throughout the course of this project we have examined several different techniques for representing and classifying facial images. We have implemented some of these methods, while

others are simply discussed in this paper as a matter of completeness in regards to the topic. For those methods that we have implemented, we have carried out experiments to gauge their relative effectiveness at performing their designed task, and have reported the results in the experiments portion of this paper. Source code is provided on the accompanying CD in order to illustrate the implementation techniques.

Face Database

To conduct experiments and test our face recognition systems, we used the Olivetti Research Laboratories (ORL) database of faces, as shown in Figure 1. This face database provides 10 sample images of each of 40 subjects. The different images for each subject provide variation in views of the individual such as lighting, facial features (such as glasses), and slight changes in head orientation. We chose to use this face database because it seemed to be a standard set of test images used in much of the literature we encountered dealing with face recognition. It should be noted, however, that this face database contains somewhat of a bias as to the type of faces represented. Males outweigh females in the subject population by a factor greater than three to one. Also, very few races outside Caucasian are represented in the database. The primary age grouping of the individuals captured seems to range from the late 20's to the mid 30's. Clearly there is an under representation of young and old people. Despite these shortcomings, this database does provide a starting point to test face recognition technologies, so long as one understands that performance of any system tested will most likely be a somewhat optimistic estimate.

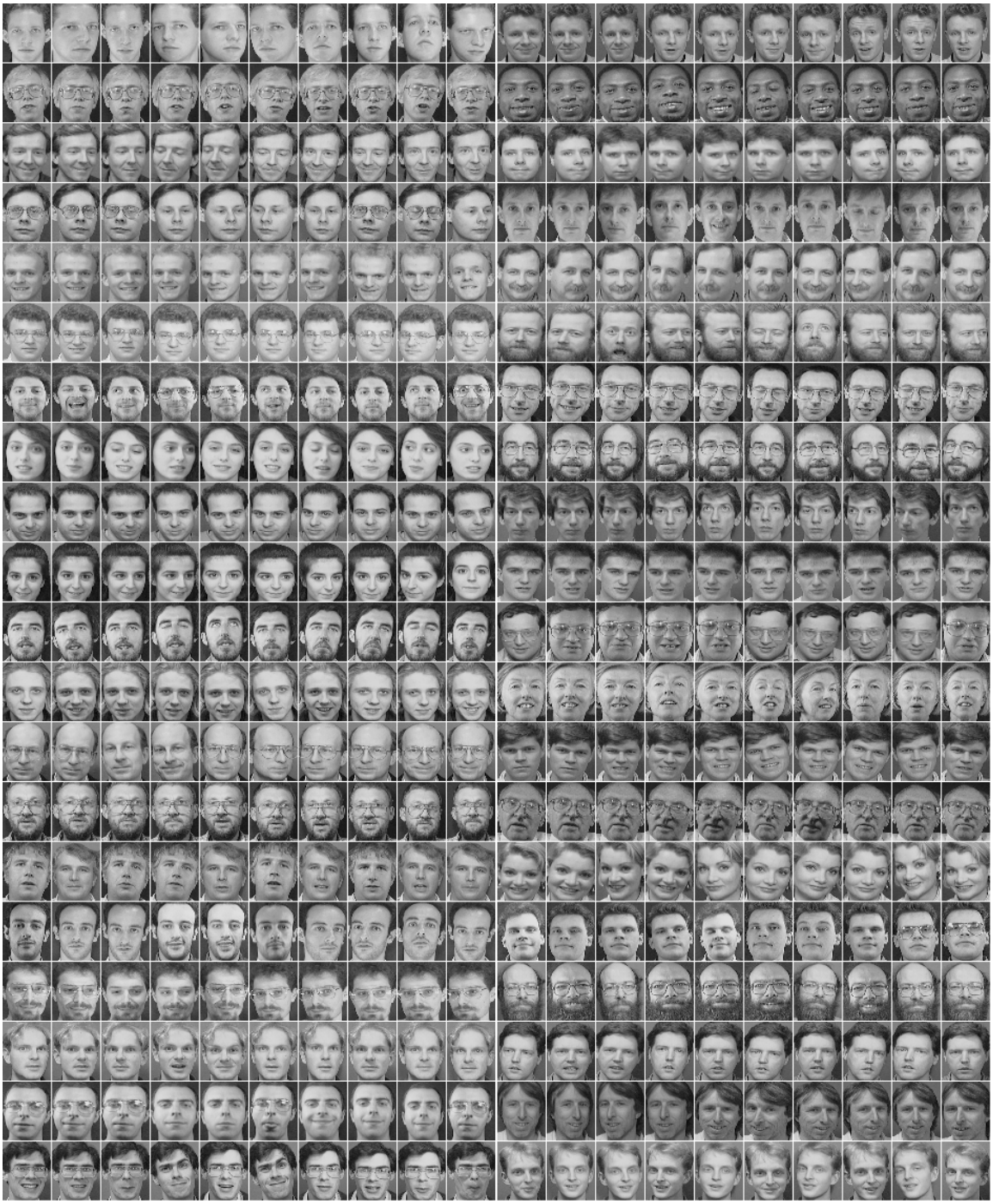


Figure 1 - Images in the ORL face database

The Eigenface Method

Introduction

The eigenface method for human face recognition is remarkably clean and simple. Where other face recognition methods are forced to attempt to identify features and classify relative distances

between them[1], the eigenface method simply evaluates the entire image as a whole. These properties make this method practical in real world implementations, outside the laboratory.

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.

This technique is similar to what is done to represent colors. Base colors are chosen and then all other colors are represented in terms of the base colors. If Red, Green, and Blue were chosen (RGB) then there would be three coefficients to represent the intensity of each color. If we wanted to represent purple, we would choose coefficients so that the intensities of red and blue were approximately equal, and the coefficient of green would be zero.

In this eigenface model, the question then becomes "What base faces do we use?" The color analogy continues to hold in this analysis, just as the choice of base colors constrains what other colors we can create, so does the choice of base faces dictate how well we can model other faces. The generation of these base faces will be 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

Notational Overview[2]

Throughout this section we will illustrate how to generate eigenfaces from the sample images. The table below provides a summary and quick reference for the terms and symbols being used in these calculations.

[1] So-called "Mr. Potato-head" algorithms
[2] Most notation used in this paper has been derived from that equations used by Zhujie & Yu [3] and Turk & Pentland [1]

Symbol	Meaning
M	The number of sample images.
k	The number of eigenfaces to be generated. The value can be altered to tweak the performance of the system; however $k \leq M$
t	The number of individuals known to the classification system. As a rule, $t \leq M$ and $k << t$.
$\tilde{\mathbf{A}}_1 ... \tilde{\mathbf{A}}_M$	These are the sample images as column vectors. Each sample image is of uniform size *x* pixels across and *y* pixels down. The size of each of these vectors is then is then (x*y) × 1
Ø	This is the average image found from the sample images. This column vector is of the same size as the sample images vectors
$\ddot{\mathbf{O}}_1 ... \ddot{\mathbf{O}}_M$	These vectors are the difference between each sample image and the average image. These column vectors are of the same size as the sample images vectors.
A	This is the matrix generated by considering each of the Φ vectors as a column of this matrix. The dimensions of this matrix are (x*y) × M (where x and y are the size of the sample images and M is the number of sample images)
$\lambda_1 ... \lambda_k$	These are the eigenvalues of the $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ matrices. The value of k depends on the number of eigenfaces desired. $k < M$ The eigenvalues are ranked by magnitude $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k$
$\mathbf{X}_1 ... \mathbf{X}_k$	These are the eigenvectors that correspond to the eigenvalues $\lambda_1 ... \lambda_k$ for the $\mathbf{A}^T\mathbf{A}$ matrix. These column vectors are of size k × 1.
$\mathbf{u}_1 ... \mathbf{u}_k$	These vectors are the eigenfaces generated from the sample images. They are the eigenvectors of the $\mathbf{A}\mathbf{A}^T$ matrix, each corresponding to the eigenvalues $\lambda_1 ... \lambda_k$. These column vectors are of size (x*y) × 1.
$\mathbf{U} = \lfloor \mathbf{u}_1 \ldots \mathbf{u}_k \rfloor$	**U** is the matrix formed by considering $\mathbf{u}_1 ... \mathbf{u}_k$ to be its columns. This matrix of eigenfaces (or eigenvectors) is not strictly required; however it can make calculations simpler, such as when one is projecting/recovering images to and from face-space.
$\grave{\mathbf{U}} = \{\grave{u}_1, \grave{u}_2, \ldots, \grave{u}_k\}$	$\grave{\mathbf{U}}$ represents the vector of weights that result from a projection into face-space. Each $\grave{u}_1, \grave{u}_2, \ldots, \grave{u}_k$ is a scalar (positive or negative) such that when each is multiplied by its corresponding eigenface and then all the weighted eigenfaces are added together, the original image results. The representation will only be accurate for images that are structured for natural representation in face space (namely face images).

Generating the 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.

We will assume that M sample images are being used. Each sample image will be referred to as $\tilde{\mathbf{A}}_n$ where n indicates that we are dealing with n^{th} sample image ($1 \le n \le M$). Each $\tilde{\mathbf{A}}_n$ should be a column 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 $\tilde{\mathbf{A}}_n$ 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) × 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.

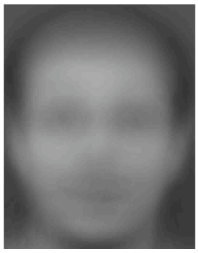
Calculating the Average Face

The next step is to calculate the average image, $\boldsymbol{\Psi}$, as follows:

$$\boldsymbol{\Psi} = \frac{1}{M}\sum_{l=1}^{M} \tilde{\mathbf{A}}_l$$

This average image will be a column vector of the same size as the sample images ((x*y) × 1). If one were to reinterpret the vector as a normal image, it would appear as one might expect, as shown in Figure 2.

(a)



(b)

Figure 2 - (a) The average of two faces (b) the average image for the face database

Generating the Difference Faces

The next step is to calculate the difference faces by subtracting the average face from each sample image.

$$\ddot{\mathbf{O}}_n = \tilde{\mathbf{A}}_n - \emptyset, \qquad 1 \le n \le M$$

Each will be a column vector the same size as our sample image vectors ((x*y) × 1). The purpose of calculating these difference faces is to allow us to calculate the covariance matrix for our sample images. The covariance matrix is defined by $\mathbf{AA}^T$ where $\mathbf{A} = \left[\ddot{\mathbf{O}}_1 \ \ddot{\mathbf{O}}_2 \ \ddot{\mathbf{O}}_3 \ldots \ddot{\mathbf{O}}_M\right]$, that is the columns of the $\mathbf{A}$ matrix are formed by the differences faces $\ddot{\mathbf{O}}_n$. The matrix $\mathbf{A}$ will be of size (x*y) × M.

Calculating the Eigenfaces

Our goal is to calculate the eigenvectors (they are referred to as eigenfaces, which will be discussed later) of the $\mathbf{AA}^T$ matrix. This cannot be done directly, because the size of the $\mathbf{AA}^T$ is (x*y) × (x*y). Even for small sample images this is very large. A standard sample image might be approximately 200x200 pixels. Clearly, doing these calculations on the resulting matrix of size 40000 × 40000 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 $\mathbf{AA}^T$ matrix can actually be found by considering linear combinations of the eigenvectors of the $\mathbf{A}^T\mathbf{A}$ matrix. This is extremely usefully when one realizes that the size of the $\mathbf{A}^T\mathbf{A}$ matrix is M ×M. For practically all real world situations M << (x*y), with M perhaps ranging from 100 to 400 images. The eigenvectors of this matrix can be readily found through the following formula:

$$\mathbf{u}_k = \frac{\sum_{l=1}^{M} \ddot{\mathbf{O}}_l \mathbf{X}_{lk}}{\sqrt{\lambda_k}}$$

For this equation $\mathbf{u}_k$ is the k[th] eigenface of the training data (the eigenvector of the $\mathbf{AA}^T$ matrix). To calculate $\mathbf{u}_k$ we sum all of the difference faces $\ddot{\mathbf{O}}_n$, multiplying each by the l[th] value of the k[th] eigenvector of the $\mathbf{A}^T\mathbf{A}$ matrix (which is $\mathbf{X}_{lk}$)[3]. The summation is then divided by the square root of the k[th] eigenvalue, which was calculated for the $\mathbf{A}^T\mathbf{A}$ matrix. Here we employ the fact that the eigenvalues for the $\mathbf{A}^T\mathbf{A}$ and $\mathbf{AA}^T$ are the same (though if we were going to calculate all of the eigenvalues of the $\mathbf{AA}^T$ matrix, we could get more values, the eigenvectors of the $\mathbf{A}^T\mathbf{A}$ only represent the most important subset of the eigenvalues of the $\mathbf{AA}^T$ matrix).

[3] Note that it is important to use the l[th] value of the k[th] eigenvector rather than the k[th] value of the l[th] eigenvector, doing so incorrectly will produce strange, incorrect eigenfaces!

Figure 3 - Eigenfaces generated from the face database

Figure 3 shows nine eigenfaces generated from the face database. Each of these images represents the image interpretation of one of the $\mathbf{u}_k$ calculated previously. As one can see, the images appear almost as ghosts, each with a different portion of the face accented. These images have been scaled in intensity so that the features can be seen more readily.

The Discriminating Power of Eigenfaces

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 eigenvalue is 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. [6] 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 section.

We have shown the eigenfaces represent images with the most discriminating power in face space. It should be noted, however, that although our selection of basis faces guarantees discriminating power, it does not guarantee a good system for classifying similar faces, namely grouping different pictures of the same person's face. Experimental results have shown, however, that there is evidence to support the claim that this system can be used as the basis for a face classification system.

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. We examine this problem experimentally later in this report.

The number of eigenfaces used also depends on the problem under examination. It is possible to do reconstruction of projected face images using the eigenfaces. In this situation, many more basis-faces are needed in order to obtain an accurate reconstruction. Once again, the actual number of eigenfaces needed is not known, and needs to be determined experimentally.

Projecting Faces into Face-Space

Now that the eigenfaces have been created, we must be able to project a face image into face-space in order to recognize or analyze it. Doing this is a very straightforward procedure, as illustrated by the following formula.

$$\grave{\mathbf{U}} = \mathbf{U}^T\left(\tilde{\mathbf{A}}_n - \mathbf{\Ø}\right)$$

Put simply, the vector of weights is found by multiplying the transpose of the matrix $\mathbf{U}^T$ ($\mathbf{U}^T$ is formed by letting each eigenface form a column of the matrix) by a vector that is found by subtracting the average face image ($\mathbf{\Ø}$, a column vector) from a sample or test image ($\tilde{\mathbf{A}}_n$, a column vector). It should be noted that although $\tilde{\mathbf{A}}_n$ represents the nth sample image in our nomenclature, this image could be any sample or test image, so long as it has already been converted into a column vector.

The previous formula calculates all of the weights simultaneously. In an environment that does not easily support such matrix operations, one may wish to calculate the weights one at a time. The following formula accomplishes this:

$$\grave{u}_n = \mathbf{u}_n^T\left(\tilde{\mathbf{A}}_n - \mathbf{\Ø}\right)$$

Here, each $\grave{u}_n$ is calculated by considering the nth eigenface (a column vector) as well as the average image and image to be projected into face space. This process is repeated for $n = 1, 2, \ldots k$ so that the weights are calculated for each eigenface.

Now that a method of projecting images into face space has been defined, the problem of face recognition becomes one of everyday pattern recognition. [1] Later sections detail several approaches to address pattern recognition issues.

Reconstructing Faces From Weight vectors

The eigenface recognition 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. [1] [7] Because of this, it is possible to reconstruct an original face image from the known eigenface weights.

$$\tilde{\mathbf{A}}' = \left(\sum_{n=1}^{k} \mathbf{u}_n \grave{u}_n \right) + \mathbf{\Ø}$$

In this equation, the recovered face image $\tilde{\mathbf{A}}'$ is found by adding the weighted eigenfaces $\mathbf{u}_n$ together and then reintroducing the average face vector. The face vector $\tilde{\mathbf{A}}'$ is still in column vector form after this reconstruction, and must be converted into a normal image prior to viewing.

Figure 4 - Images and their reconstruction from face-space representations; original image on far left; 5, 10, 50, 100, 150, 200 eigenface reconstructions from left to right.

Figure 4 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. As one can see, this face has a more accurate reconstruction than any of the other images, including the second, which is of the same individual as the first, but was not accounted for when calculating the eigenfaces.

The third 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.

This image reconstruction procedure is not particularly useful in the recognition domain, however it can provide insight into how well a given face image is being represented by its projection into face-space. Once again, it should be noted that a good representation in face-space do not necessarily imply good discriminating ability, however a poor representation would most likely indicate a loss of information, and guarantee a poor basis for a classification technique.

Classifying Faces

When a face classification system is given a test input image, one of three possible classifications can result:

1. Not a face
2. Unknown face
3. Recognized face

Not a Face

The “Not a face” classification is useful in situations where face detection is done automatically. Several automatic face detection mechanisms currently exist, and most function with a good degree of accuracy. There is always the possibility, however, that a “phantom” face image will be detected. Because of this, a face recognition system that is to be used in this type of environment should be able to seamlessly handle erroneous test faces. The interpretation of “not a face” in the eigenface system 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. In situations where faces are guaranteed to be in the image being tested, such as in a mug shot database-type system, or one where a human locates faces prior to presenting them to the system, this feature is not required. In these cases, this additional classification option could actually introduce further error into the results and should probably be omitted.

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. In the eigenface system, "unknown face" is interpreted as a point in face-space that is not close enough to any cluster of points for a known individual, yet it is close enough to the cluster of all face images so that the image is still classified as a face. Clearly, if one knows ahead of time that only pre-trained individuals will be tested against the system, one can remove this type of classification and possibly experience a gain in accuracy by allowing the system to make its best guess as to which cluster of points the test image is closest.

In other situations, however, an opposite approach might be taken. If the face recognition system is to be used in a security environment, the cost of classifying an unknown individual as known and authorized is extremely high. Yet the cost of classifying a known individual as unknown is relatively low, since that individual will simply have to repeat the authorization process. Because of this, it might be beneficial for the system to reject any images that are not extremely close to a known individual, and to err on the side of caution.

A final use for the "unknown face" classification is to allow the system to learn. 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. The individuals would remain without proper classification until a name was provided to go with the face, so to speak, but this technique could still be useful to perhaps identify recurring unknown individuals in a security system, which could trigger red flags that suspicious activity was taking place.

Recognized Face

The final classification type is fairly straightforward in meaning. 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. In the eigenface technique, a classification of know corresponds to the vector generated by the test image being sufficiently close to a cluster of a known individual.

If one wishes to make the system adaptive, a vector of a recognized individual's test image could be added to the existing system. This would allow recognition to be uninterrupted by gradual changes in individuals' appearances. This has the downside of allowing the system to continually train itself while in production, which always has the potential side effect of allowing the recognition system to over-recognize (recognize when a test image should be unrecognized) individuals as time goes on.

Classification Techniques

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.

Euclidian Distance

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. The idea behind the Euclidian distance classifier is that one computes the average of several training vectors for each possible categorization and then classifies a given test vector by determining to which cluster the average is the vector nearest.

The classifier is constructed as follows:

First consider that there are t distinct individuals to be recognized by the system.

$$\overline{\mathbf{U}}_{0 \le j \le t} = \frac{1}{\#samples\ in\ j} \sum_{samples \in j} \grave{\mathbf{U}}_{\in samples}$$

For each individual j, $0 \le j \le t$, we calculate the average feature vector $\overline{\mathbf{U}}_j$ by summing the sample vectors that belong to j and then dividing by the number of samples used. It should be noted that it is not necessary to use all training samples in this average. In fact, it is possible that using many training samples with a wide variety of poses will actually be detrimental to the system.

Consider a situation in which training data contains two distinct types of poses: one in which the subject is posing normally and one in which the subject is sticking out his or her tongue. This change in pose could create two distinct clusters in face-space, as shown in Figure 5. One can observe that when the mean (represented by the square) is blindly calculated for the entire dataset, it falls between the two clusters and is not particularity close to any of the points. This is especially a problem when cutoffs are employed in order to allow for classifications of "not a face" and "unrecognized face", as discussed later. A more intelligent approach is to consider the two clusters separately and calculate the mean for each (as shown in Figure 6). This allows all points to lie close to one of the two means, and each of these means can then be mapped to the original individual.

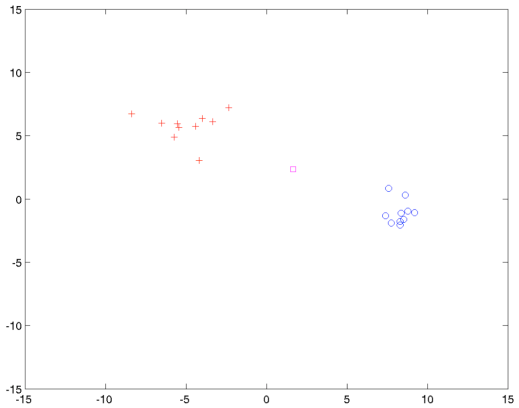


Figure 5 – A mean that poorly represents the data

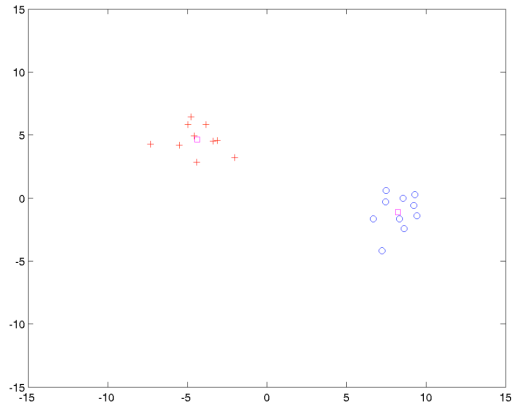


Figure 6 – Better representation of the data

In our example, it is easy to think of separating the sample data into poses with and without a tongue showing. In general, however, it is not always practical to have human intervention when training the classifier. In these situations, it may be appropriate to use unsupervised clustering techniques on the samples given for each subject. By using some predetermined criteria of what defines a distinct cluster, the system could identify one or more clusters of points in face-space for each individual, calculate the mean for each cluster, and then indicate that all means are to be classified as the same person. It should be noted that these ideas of unsupervised clustering are not currently implemented on the system on which experiments are being carried out, but rather are an area for future work.

Once the Euclidian distance classifier has been created, classification of a given test image is a straightforward process. First the test image is projected into face-space, as described in the eigenface section of this paper. Next, the distance to the closest mean for a known individual is found. This distance is compared to two thresholds, which determine if the point is close enough to a cluster to be considered a face image, as well as if it is close enough to be considered a known face. Both the "not a face" (NAF) distanced and the "unknown face" distance must be determined experimentally (which is carried out in the experiments portion of this project). It should be noted that classification time in the Euclidian distance method is linear with the number of possible known people (assuming one cluster per subject; if more than one cluster per subject is used than classification time is based on the product of the two quantities). This classification technique is fast enough to be performed in real time.

Neural Network

The neural network classifier is a more advanced alternative to the Euclidian distance method. The primary advantage of the neural network is that no assumptions need to be made as to how clustering will occur in face-space. Given the correct network configuration, situations like the tongue out versus normal pose problem should be handled automatically by the system, with no additional design consideration.

The configuration used to construct the neural network for face recognition was based on work by Zhujie and Yu. [3] The network contains three fully connected layers, as in the diagram shown in Figure 7.

It should be noted that this network configuration is designed to accept the weight values that are obtained by projecting a test face into face-space. This projection is accomplished in the same way as described in the eigenface section, and is the same way used in the Euclidian distance classification method. Other possible network configurations include an additional layer, which takes the place of the projection into face-space. In this configuration, the projection is learned along with the rest of the classification. While it is possible that this setup might allow the system to degrade more gracefully with error, we have not implemented it for our experiments because of the time required to train the network. With the inputs to the network being the individual pixels of the image, we would expect significantly longer training time than for a network accepting the weights as data, which was already a significant amount of time. This is another area we leave to future work.

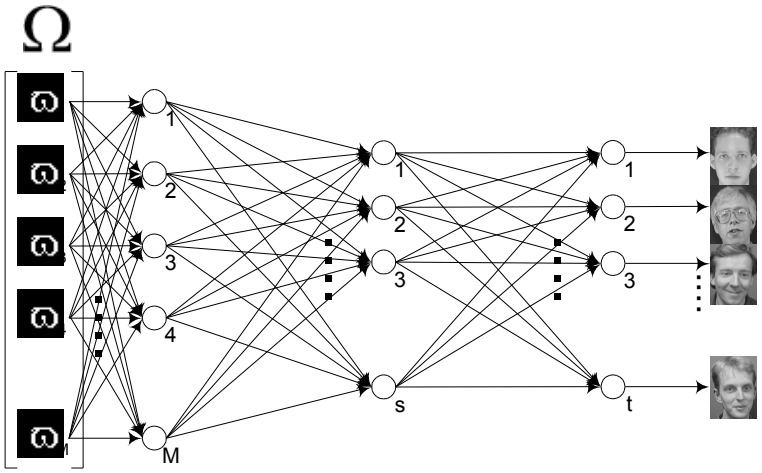


Figure 7 - Neural network architecture for classification

Figure 7 shows the network configuration used for classification. The network contains three layers. The first layer of the network contains M nodes, where M is the dimension of face-space (the number of eigenfaces generated). The second layer contains s nodes, where, according to [3], s is a positive integer determined experimentally. Due to time constrains in continually retraining our network, we simply set s = M and achieved reasonable rates of classification.[4] It is possible that a smaller value of s would have been able to obtain the same results, however.

The output layer of the network contains t nodes, where t is the number of individuals known to the system. This configuration for the output nodes allows a simple method for classifying an individual. For individual j, $0 \le j \le t$, the j^{th} output is expected to be 1, and the rest should all be zero. This makes it extremely easy to classify an image based on the network's output. One simply has to find the node whose output is closest to one (the greatest) and then classify the image accordingly.

[4] under certain conditions

To identify images that are not recognized or not a face, a similar scheme to the Euclidian distance classifier is used. Thresholds are employed to indicate when the output of the network is "confident" enough to indicate the image is of a face or even a known face. One classifies an image as not a face if the value of the highest output is not sufficiently close to one. The same is true for an unknown face classification, with a more strict cutoff (closer to 1). It is not clear if this is an efficient or valid way to make these classifications; this problem is examined experimentally later.

For the actual neural network implementation, we used the Matlab neural network toolbox. This provides an efficient implementation for the algorithms necessary to build the classifier. In actually constructing the network, we used the tangent sigmoid ('tansig' in Matlab) and the Levenberg-Marquardt ('trainlm' in Matlab) backpropagation algorithm. Both of these are default, general algorithms provided for building feed-forward networks.

Experiments

Experiment #1 - The Dimensionality of Face-Space

Goal

The purpose of this experiment is to find the correlation between the number of eigenfaces used and the error rate in classification.

Procedure

This will be accomplished by the following procedure:

1. Create different numbers of eigenfaces from the training data
2. Create a Euclidian classifier to function in the face-space specified by the eigenfaces created
3. Train the classifier on the training data
4. Test the classifier on the test data, as well as on the training data, recording results separately

Only the Euclidian distance classifier will be used for this experiment. One assumption being made is that the optimal number of eigenfaces is independent of the type of classifier being used. This simplifying assumption and others are listed below, along with parameters used to implement the classifier.

Simplifying Assumptions/Parameters Used

- More eigenfaces will always produce greater classification accuracy, since more information is available
- The number of eigenfaces needed for classification is not dependent on the method of classification; i.e. if X eigenfaces are needed for Euclidian distance than the same number of eigenfaces is optimal for all classifiers, therefore experiments can be performed on only one type of classifier
- The number of eigenfaces is an independent parameter; this parameter can be optimized independently of all other parameters for the classifier; to carry out this experiment, the Euclidian distance classifier will be used.

- 4 faces per subject will be used to generate eigenfaces (160 total images)
- 4 faces per subject (same images as were used to generate the eigenfaces) will be averaged together to find the average face for a given individual
- The number of eigenfaces will be measured at intervals of 5, starting at 5 and going to 120
- Face images will not be allowed to be classified as unknown or not a face, nor will images which are not faces be presented to the classifier

Results

Figure 8 shows the relationship between classification success and the number of eigenfaces used. The graphs level off from their rapid rate of increase in classification somewhere between 20 and 40 eigenfaces. It is interesting that one can obtain a classification rate of over 75% when using less than 10 eigenfaces. It would appear that that by 40 eigenfaces the rate of increase in the accuracy of classification has sufficiently slowed, therefore we will use 40 eigenfaces for our subsequent experiments. This is slightly conservative, as the curves have probably flattened out by 30 eigenfaces; however, since there is little penalty for using a few more bases-faces, we will use 40 in hopes of slightly increased accuracy.

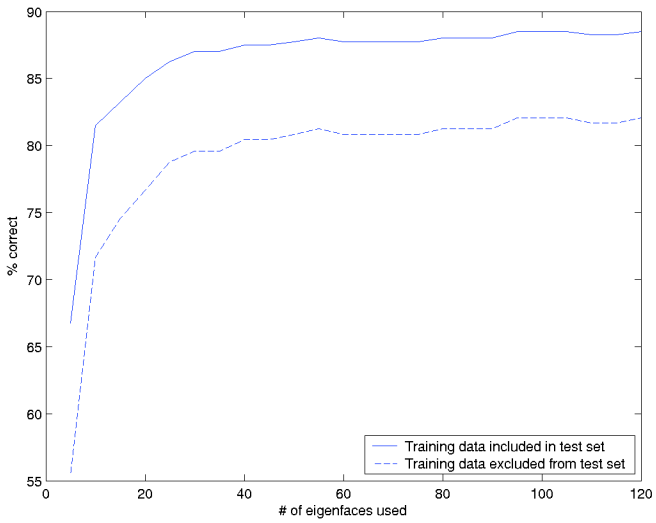


Figure 8 - The number of eigenfaces used vs. success in classification

It should be noted that this experiment is very strictly limited in how it was performed. Since we did not allow the classifier to return "not a face" or "unknown face" we allowed the classifier to take its best guess at every test image. One would expect subsequent experiments to show less accuracy as the classifier is forced to make more distinctions. The low 80% classification rate (on test-only data) is still impressive, however, given the simplicity of the classification system as well as the variation in poses found in the test data.

While carrying out this experiment we noticed that the ideal number of eigenfaces approximately equaled the number of individuals encountered by the system during training. This raised the question of whether our measurements of the dimensionality of face-space were closely tied to the number of training subjects we used. To test this connection we repeated the same experiment with varying numbers of individuals (the classifier was trained and tested on only a

subset of the full face database) to observe the optimum number of eigenfaces for each subject population size. The results of this experiment are shown in Figure 9. Note that this graph only shows the success at classifying test data only (training data excluded) therefore the smaller subject populations are *not* influenced by a higher percentage of test data also being training data.

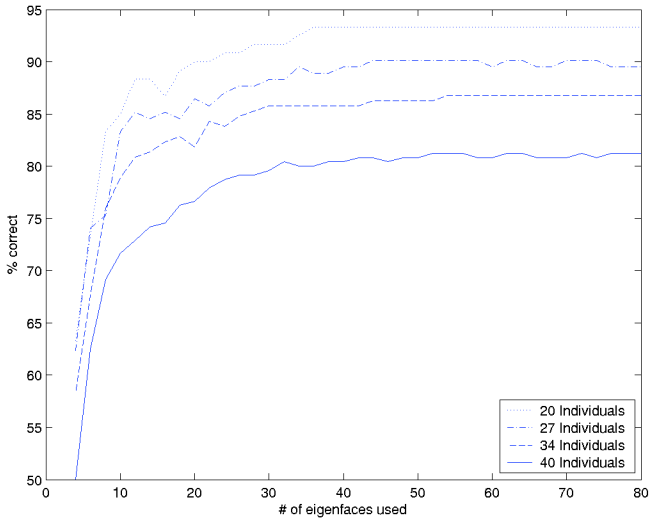


Figure 9 - Multiple subject population sizes and success vs. number of eigenfaces

The results seem to indicate that there is not a direct connection between the number of subjects to be identified and the number of eigenfaces needed to represent that population. Figure 9 shows that the curves for all four different sizes of subject populations flatten out in approximately the same area. It is interesting to note that the smaller subject populations obtained higher classification rates than the larger ones for any given number of eigenfaces. This suggests that this method of classification may break down if larger populations need to be identified.

Experiment #2 – Determining the Threshold for Unknown Faces

Goal

To determine the proper settings for the cutoff parameters that determine when a face is classified as unknown for both the Euclidian and neural network classifiers.

Procedure

The procedure for finding the setting of the unknown face cutoff parameter is identical for the Euclidian distance classifier and the neural network classifier except for the range on which the cutoff may exist.

For the Euclidian distance classifier, this parameter represents the maximum distance between a test point and the center of the closest cluster of training points for a known individual. The parameter essential tells the system that all points for a given individual must lie within a sphere of radius specified by this parameter in n-space, where n-space is the dimensionality of face-space.

For the neural network classifier unknown face threshold parameter represents minimum output strength for the most positive output node. Since the network is trained to have an output of one on the output node specified to indicate a given individual, if none of the output nodes are presenting a value within a reasonable tolerance of one, we will conclude that none of the output nodes are recognizing the test image as that node's known individual.

To accomplish the test, the following will be performed for each classifier:
1. Create different numbers of eigenfaces from the training data
2. Create a Euclidian/neural network classifier to function in the face-space specified by the eigenfaces created
3. Train the classifiers on the training data
4. Test each classifier on the test data, as well as on the training data, recording results separately. The classifiers will be shown both faces of individuals on which they have been trained as well as face images of individuals that are unknown.

Simplifying Assumptions/Parameters Used

- The parameter that determines whether a face is known or unknown is independent; this parameter can be optimized independently of all other parameters for the classifier
- 40 eigenfaces will be used
- The number of known vs. unknown individuals will be equal: 20 known individuals (on which the system will be trained) and 20 unknown individuals
- 4 faces per known subject will be used to generate eigenfaces (80 total images)
- 120 images of known individuals will be presented to the system as test-only data (20 individuals * 6 untrained images per individual)
- 200 images of unknown individuals will be presented to the system as test-only data (20 individuals * 10 untrained images per individual)
- 4 faces per subject (same images as were used to generate the eigenfaces) will be averaged together to find the average face for a given individual for the Euclidian classifier
- 4 faces per subject (same images as were used to generate the eigenfaces) be presented to the neural network for use as training data
- The distances tested for Euclidian distance will begin at 0 and go to 7000 in increments of 50
- The minimum confidence for the neural network will be tested on the interval of 0 to 1, in increments of 0.005
- Face images will only be allowed to be classified as a specific, identified face, or as an unknown face, the "not a face" classification will not be permitted

Results

Figure 10 shows the results for the Euclidian classifier. The overall maxim success for classification (for both test only and test + training data) occurred when the "unknown face" threshold was set at 2550. Note that at this point, about 95% of images of unknown individuals were correctly classified as unknown.

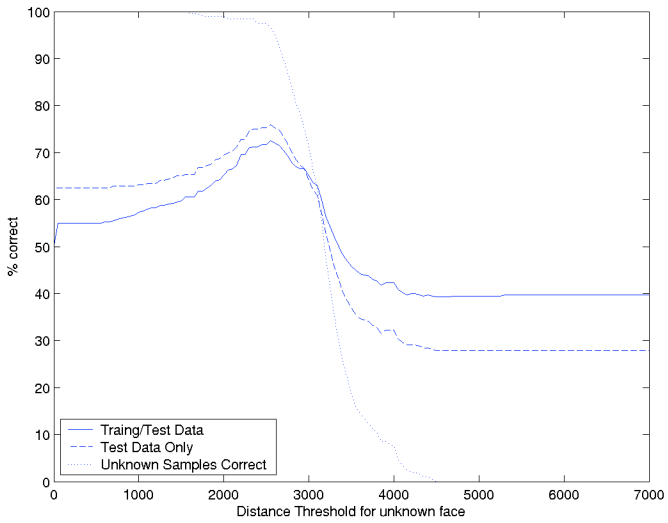


Figure 10 - "Unknown" threshold vs. classification success for Euclidian distance classifier

One can observe from the graph that after a threshold of 3000, the test data line crossed the training + test data line. Prior to 3000, the test-only data was actually performing better than the training + test data. This may seem somewhat unusual at first, but as one thinks about the dataset being presented, it is somewhat logical. For the training + test data, the images are exactly ½ known individuals and ½ unknown individuals. For the test only data, there are actually more images of unknown individuals than known ones. (120 images of known individuals, 200 of unknown individuals) In tests prior to the threshold being set to about 2000, the classifier is almost classifying every face image as unknown. Because of this the testing data, with more unknown individuals, will logically receive more correct classifications. This problem touches on an important point: it is important to know the prior probability of encountering an unknown individual in the domain in which the classifier is used. For this experiment, we have assumed that it is approximately equally likely that the system will encounter an unknown face versus a known face. In practice this might not be the case. Knowing the prior probability would allow this experiment to be carried out in a way that will better predict the accuracy of the classifier.

The results of the experiment on the neural network classifier are shown in Figure 11. The maximum classification accuracy was achieved when the threshold was set to .96. As one can see, the neural network did not perform as well as the Euclidian distance classifier. It is not clear if this method of determining if a face image is unknown is not valid in the neural network architecture, or if this is simply a drawback to using neural networks in this domain.

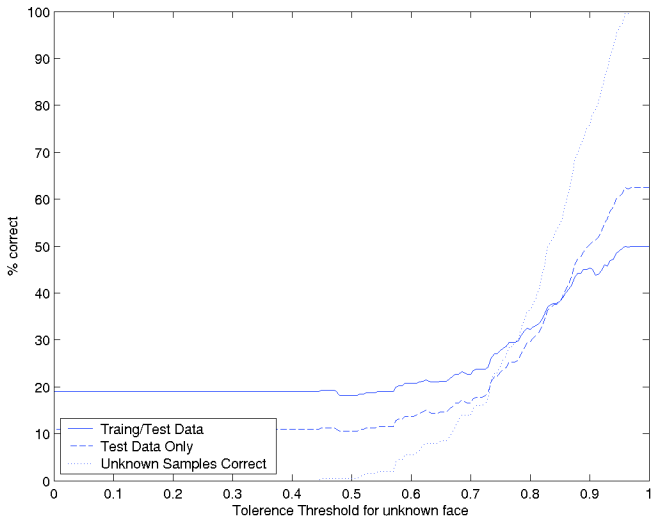


Figure 11 - "Unknown" threshold vs. classification success for neural network classifier

The following table summarizes the results from this experiment:

	Euclidian	Neural Network
Optimal Threshold Setting	2550	0.96
Accuracy for Train+Test Data	72.5%	50.0%
Accuracy for Test Data Only	75.9%	62.5%

Experiment #3 – Determining the Threshold for Images that are not Faces

Goal

To determine the proper settings for the cutoff parameters that decide when a face is classified as "not a face" for both the Euclidian and neural network classifiers.

Procedure

The procedure for determining the "not a face" cutoff parameter is the same as for determining the "unknown" cutoff parameter, except images will be used that are not faces instead of unknown individuals, and the "not a face" threshold will be varied, while not allowing the system to classify an image as "unknown".

Simplifying Assumptions/Parameters Used

- The parameter that determines whether an image is or is not a face is independent; this parameter can be optimized independently of all other parameters for the classifier
- 40 eigenfaces will be used
- The classifiers will be trained on 40 subjects
- 4 faces per known subject will be used to generate eigenfaces (160 total images)
- 400 images of known, face images will be presented to the system as test + training data (40 individuals * 10 images per individual)
- 240 images of known, face images will be presented to the system as test-only data (40 individuals * 6 images per individual)
- 30 images of natural, non-face objects will be presented to the system for classification

- 4 faces per subject (same images as were used to generate the eigenfaces) will be averaged together to find the average face for a given individual for the Euclidian classifier
- 4 faces per subject (same images as were used to generate the eigenfaces) be presented to the neural network for use as training data
- The distances tested for Euclidian distance will begin at 0 and go to 7000 on increments of 50
- The minimum confidence for the neural network will be tested on the interval of 0 to 1, in increments of 0.01
- Images will only be allowed to be classified as a specific individual or "not a face"; unknown individual will not be permitted

Results

Figure 12 shows the results for the Euclidian classifier. The results for this classifier are similar to its results in the "unknown" classification experiment. The maximum success rate of classification occurs at 3650. It is logical that this distance would be greater than the "unknown" distance, since one would expect the subset of images that contain faces to be closer together than the subset of images that contain pictures of natural objects.

The maximum success rate of classification for this experiment is slightly lower than that of the "unknown" classification experiment. This may be due to the unpredictability of non-face images projecting into face-space. It is possible that these projections are somewhat random, making their classification difficult.

It should be noted that the choice of the optimum distance for this parameter once again depends on the prior probability of the system encountering non-face images. In the experiment approximately 93% of the images presented to the classifier were faces. In situations where the non-face probability is higher, perhaps a smaller value should be used for this parameter. In situations where images encountered by the system are pre-screened to be face images (such as a mug-shot database), this parameter would be set to infinity.

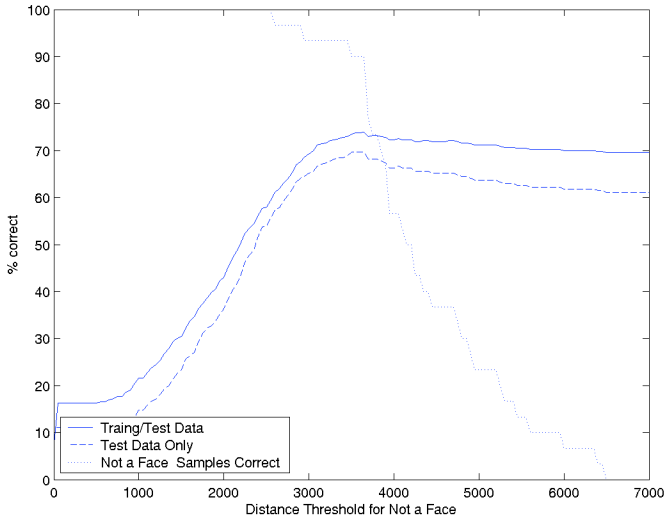


Figure 12 - "Not a face" threshold vs. classification success for Euclidian distance classifier

The neural network classifier did not perform well at this task. Figure 13 shows the results of this same experiment performed with the neural network.

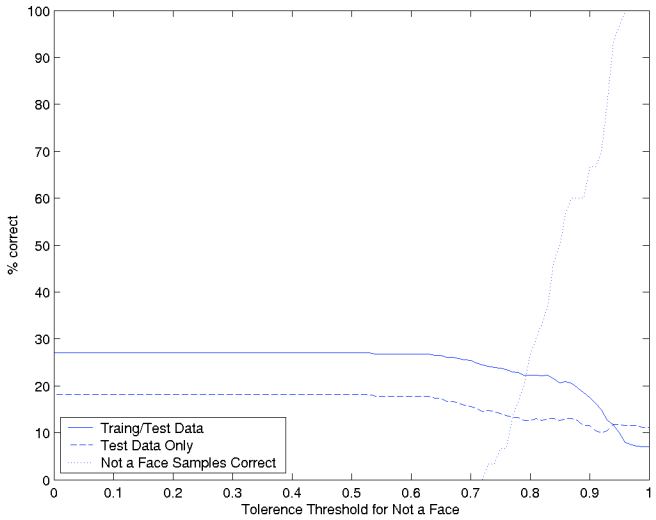


Figure 13 - "Not a face" threshold vs. classification success for neural network classifier

The optimal value for the threshold parameter is approximately .65 for the neural network, however even at this parameter value, the successful rate of classification does not exceed 30%. It is possible that this is not a value method for determining face vs. non-face with a neural network. Future work clearly is needed to evaluate other ways of determining the face/non-face distinction with neural networks.

An unfortunate side effect of the neural network's inability to correctly classify this type of problem is that it will be at somewhat of a disadvantage in subsequent experiments. In order to test the strengths of the two classifiers in various settings, known face images, unknown face images and images that are not faces will be presented to the system. Neural networks did not perform especially well on either of the latter two classifications and may be at a disadvantage for later experiments.

The following table summarizes the results from this experiment:

	Euclidian	Neural Network
Optimal Threshold Setting	3650	0.65
Accuracy for Train+Test Data	74.0%	27.0%
Accuracy for Test Data Only	69.6%	15.9%

Experiment #4 – Classifier Tolerance to Salt & Pepper Noise

Goal

The purpose of the experiment is to establish how well the two classification techniques function in the presence of varying amounts of noise.

Procedure

The procedure for this experiment is the same as for the previous ones except that in this case the parameter being varied is the amount of salt & pepper noise being introduced into the test images. The eigenfaces are generated once, and the classifiers are trained once on the noise-free data set, only the test data is changed each iteration.

Simplifying Assumptions/Parameters Used

- 40 eigenfaces will be used
- The classifiers will be trained on 37 subjects (3 subjects will be left to use as unknown)
- 4 faces per known subject will be used to generate eigenfaces (148 total images)
- All 40 individuals will be presented to the classifier (37*10 = 370 known images, 3*10 = 30 unknown face images)
- 30 images of natural, non-face objects will be presented to the system for classification
- 4 faces per known subject (same images as were used to generate the eigenfaces) will be averaged together to find the average face for a given individual for the Euclidian classifier
- 4 faces per known subject (same images as were used to generate the eigenfaces) be presented to the neural network for use as training data
- The "unknown" distance parameter for the Euclidian distance classifier will be 2550; for the neural network, .96 will be used
- The "not a face" distance parameter for the Euclidian distance classifier will be 3650; for the neural network, .65 will be used
- The classifiers will be tested with salt & pepper noise, with the magnitude of the noise ranging from 0 to 1.00 in increments of 0.01

Results

Figure 14 shows what is being done visually to the images for this experiment. Various levels of Salt & Pepper noise have been added to the images to present the classifiers with less-than-perfect faces to classify. For a human, it would be unreasonable to expect any amount of noise beyond 0.3 (the 4th image from the left) to be classified with any consistent accuracy. It would therefore be unreasonable to expect better from the classification system

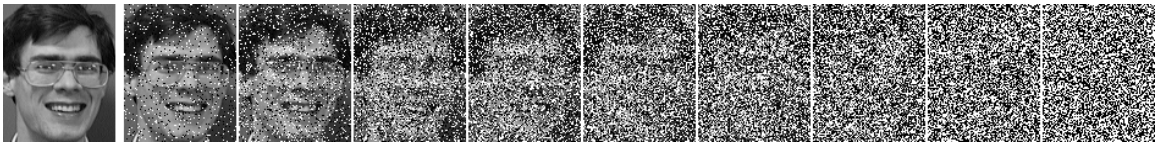


Figure 14 - (left) original image (left to right) increasing amounts of salt & pepper noise by 0.1 per image

Figure 15 shows the success rate for the Euclidian distance classifier. One can observe that the classifier degrades gracefully as the amount of noise increases, a desirable feature for any system. With zero noise, the classifier is only performing at rate between 55% and 60% after the 0.30 or 0.35 noise level, the accuracy slips below 50% and descends from there.

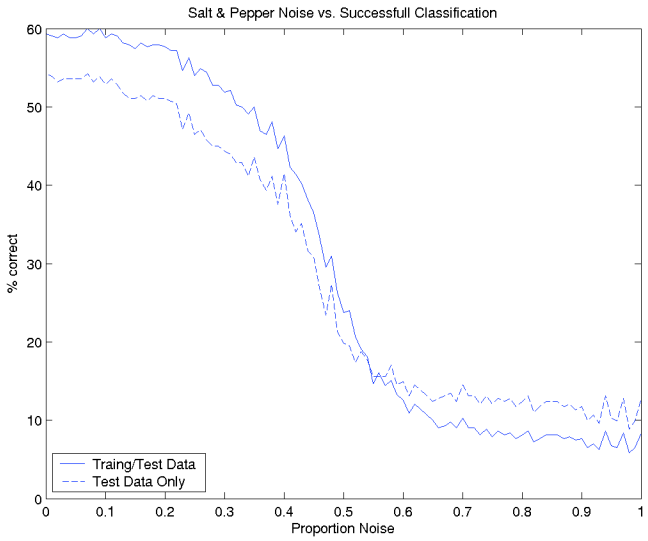


Figure 15 - The Euclidian distance classifier's performance as a function of noise present

Figure 16 show that as the noise increases, virtually all of the images are either classified as "unknown" or not a face.

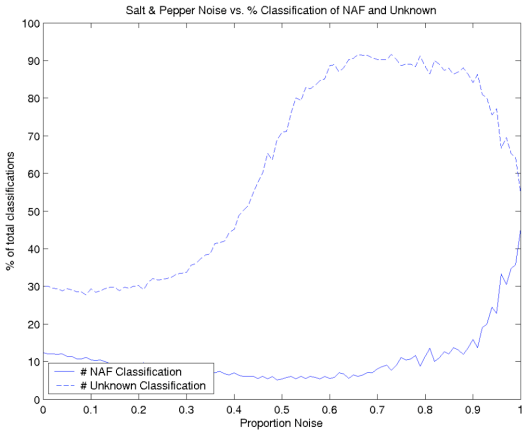


Figure 16 - The % of total images classified as NAF/unknown for various amounts of noise with the Euclidian classifier

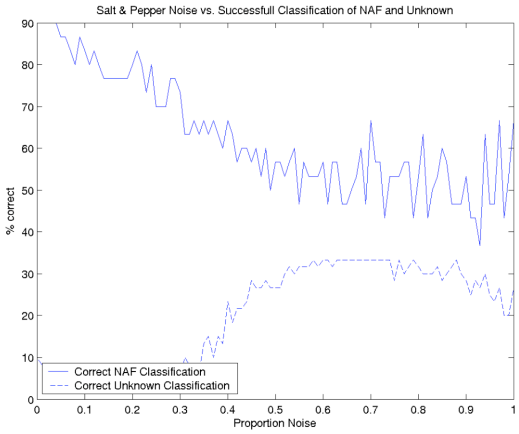


Figure 17 - Accuracy of classifications of NAF/unknown for various amounts of noise with the Euclidian classifier

The results for the neural network classifier are presented below. Unfortunately, the results do not indicate any real success in classification. The maximum classification rate does not exceed 15%. Once again it is unclear if this is due to the problems previously mentioned with the "not a face" and "unknown" classifications or if it due to the neural network having trouble classifying this type of data.

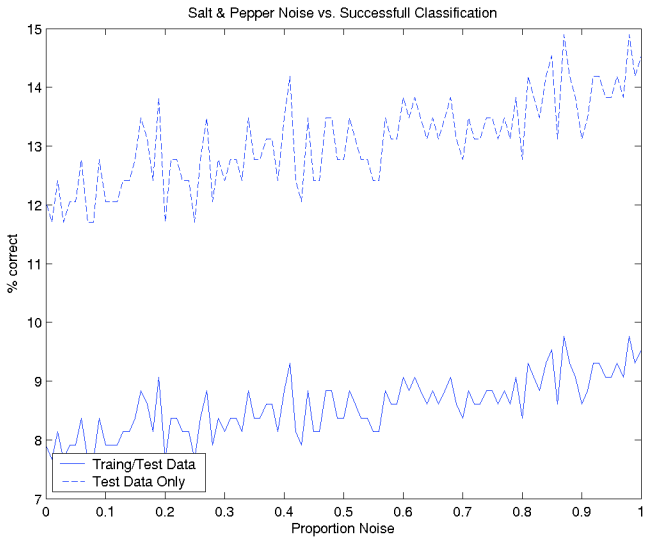


Figure 18- The neural network classifier's performance as a function of noise present

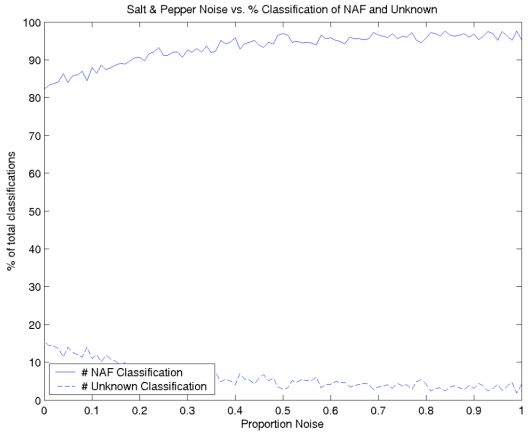


Figure 19- The % of total images classified as NAF/unknown for various amounts of noise with the neural network classifier

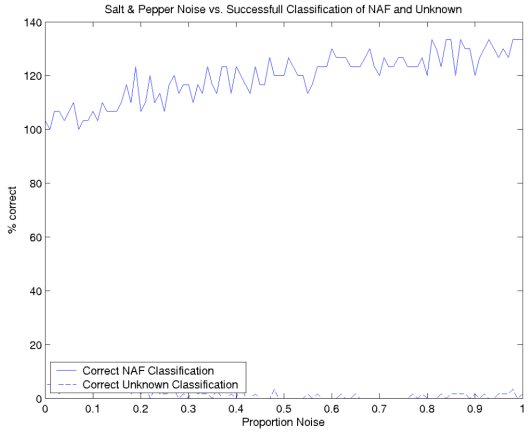


Figure 20- Accuracy of classifications of NAF/unknown for various amounts of noise with the neural network classifier

Experiment #5 – Classifying with Partially Hidden Faces

Goal

The purpose of this experiment is to observe how well the classifiers can deal with obstructions placed in front of the image. If a face recognition system is to be used on real world pictures, the classifier must be able to deal with images that are taken from a less than perfect vantage point (real-world pictures are pictures that are not specifically taken to be presented to a classification system; a picture of a person in a grass field would be a real-world picture, a mug shot would artificial one).

Procedure

In this experiment portions of the test images will be blacked out to simulate the an object between the individual in the image and the camera. Several different blockage configurations will be used to ensure that results are general rather than a special case that is easy/hard to deal with. The number of pixels blacked out will be held constant across the different obstruction types for a given obstruction level. This will allow the results across obstruction types to be easily compared.

The classifiers will be trained on unobstructed images, as they have been on previous experiments.

Simplifying Assumptions/Parameters Used

- 40 eigenfaces will be used
- The classifiers will be trained on 37 subjects (3 subjects will be left to use as unknown)
- 4 faces per known subject will be used to generate eigenfaces (148 total images)
- All 40 individuals will be presented to the classifier (37*10 = 370 known images, 3*10 = 30 unknown face images)
- 30 images of natural, non-face objects will be presented to the system for classification
- 4 faces per known subject (same images as were used to generate the eigenfaces) will be averaged together to find the average face for a given individual for the Euclidian classifier
- 4 faces per known subject (same images as were used to generate the eigenfaces) be presented to the neural network for use as training data
- The "unknown" distance parameter for the Euclidian distance classifier will be 2550; for the neural network, .96 will be used
- The "not a face" distance parameter for the Euclidian distance classifier will be 3650; for the neural network, .65 will be used
- The classifiers will be tested with various amounts of obstruction, ranging from 0 to 0.5 (0 being the image is normal, 0.5 being the a box half the width of the picture is covering the data)

Results

Figure 21 shows the different types of obstruct used in this experiment. From left to right the images are:

- Unobstructed image
- Horizontal strip
- Vertical Strip
- Upper-right box
- Lower-left box
- Center box

Figure 21 - Various type of image obstruction (0.33 obstruction intensity)

The results for the Euclidian classifier are remarkably pleasing. The success of classification shows almost no degradation through 0.15 obscuring. It is interesting to note that the center box type of obscuring as well as the horizontal strip type caused the success of classification to degrade the fastest. This would tend to indicate that the most distinctive portions of the faces were being covered up by these types of obstructions. This type of classifier seems to do a good job dealing with obstruction.

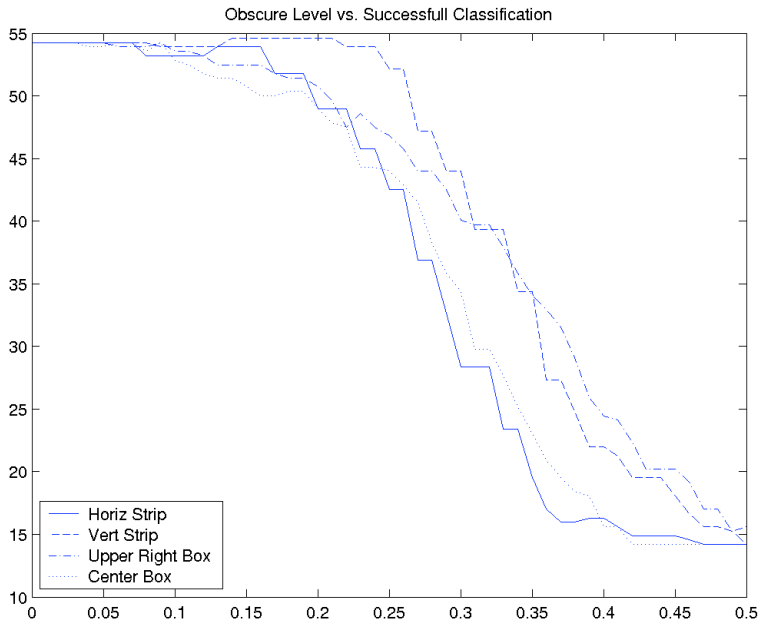


Figure 22 - Euclidian classifier success rate of classification for various types/magnitudes of image obstructions

Once again the results for the neural network (below) were meaningless, most likely from reasons discussed previously. One can tell that the data is not useful since the classification rate actually increases as the obstruction level increases for certain types of obstructions.

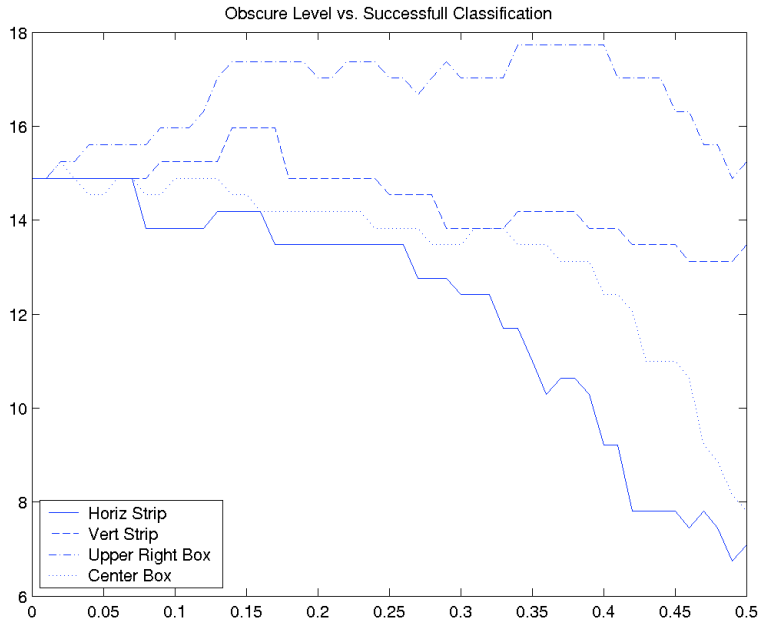


Figure 23 - Neural network classifier success rate of classification for various types/magnitudes of image obstructions

Conclusions

The primary conclusion that can be drawn from the results of our experiments is that the Euclidian distance classifier seems to have many advantages over its neural network counterpart. The Euclidian distance classifier matched or out performed the neural network classifier in accuracy for every test.

We also believe that more work needs to be done with the neural network classifier. It is unclear why this classifier did so poorly on many of the experiment that allowed for the classification of "not a face" or "unknown face" In a controlled test where only known individuals were presented to the classifier, and the NAF and "unknown" classifications were not allowed, the classifier was able to reach a level of 86.3% on training + test data, and 77.1% on test only data. It is surprising that it was not able to do better on the other tests.

Perhaps even more importantly the Euclidian distance classifier was faster than the neural network in both training and classifying. For most of the experiments, the Euclidian distance classifier could be trained within a matter of seconds, while the training of the neural network took anywhere from several minutes to an hour[5], depending on the initial random weights created in the network.

A surprising result was that the Euclidian distance classifier also out performed the neural network in classification speed. The difference in this rate was much less noticeable than the

[5] Training for the neural network typically took about 15 epochs, however this could vary wildly depending on the initial weights

training times; however for large experiments, the difference was definitely observable. The neural networks may scale better to larger numbers of known individuals, since the Euclidian distance classifier is necessarily linear with the number of known individuals, where as this does not have to be the case with neural networks.

Future Work

There are many interesting problems that remain in the area of face recognition. One problem is image preprocessing prior to the application of the eigenface method. It may be possible to gain better accuracy in classification if one segments the spectrum of people into different spaces. For example, if one was able to determine if an image was of a man or a woman, one could use this categorization to send an image to one of two classifiers, each specifically trained with that type of individual in mind. This would mean that there would be a set of eigenfaces specifically for males and one specifically for females (face spaces with gender, so to speak). Work in this area has been done by Lizama, Waldoestl and Nickolay [4], however it would be interesting to extend it to use eigenfaces to act as the gender classifier as well. A general face-space would be created in addition to the male and female face-spaces, with the sole purpose of being used to classify an image as male or female.

Another area of future work is improving our neural network classifier. As mentioned in the previously, it is possible to construct the network to take its input directly from the image data rather from the vector that results from an images projection into face-space. Perhaps learning the face projection function could increase the accuracy of the neural network classifier. Additionally, more experiments are needed to see if there are other ways to tweak the network configuration to produce better results.

[1] M. A. Turk and A. P. Pentland, “Face recognition using eigenfaces,” in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition,* Maui, Hawaii, 1991, pp. 586-591

[2] M. Turk and A. Pentland, “Eigenfaces for Recognition”, *Journal of Cognitive Neuroscience*, March 1991

[3] Zhujie and Y.L. Yu, “Face Recognition with Eigenfaces”, Unknown source, The Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, HK, pp. 434-438

[4] E. Lizama, D. Waldoestl and B. Nickolay, “An Eigenfaces-Based Automatic Face Recognition System” *1997 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, New York, NY, 5 vol. 4535, 1997, pp. 174-177

[5] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota (Florida), December 1994

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

[7] Sirovich, L. & Kirby, M. “Low dimensional procedure for characterization of human faces,” *Journal of the Optical Society of America A,* 4(3), pp. 519-524, 1987