Computer Science
Technical Report

ANALYSIS OF PCA-BASED AND FISHER DISCRIMINANT-BASED IMAGE RECOGNITION ALGORITHMS

Wendy S. Yambor

July 2000

Technical Report CS-00-103

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

THESIS

ANALYSIS OF PCA-BASED AND FISHER DISCRIMINANT-BASED IMAGE RECOGNITION ALGORITHMS

Submitted by

Wendy S. Yambor

Department of Computer Science

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2000

COLORADO STATE UNIVERSITY

July 6, 2000

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY WENDY S. YAMBOR ENTITLED ANALYSIS OF PCA-BASED AND FISHER DISCRIMINANT-BASED IMAGE RECOGNITION ALGORITHMS BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Advisor

Co-Advisor

Department Head

ABSTRACT OF THESIS

ANALYSIS OF PCA-BASED AND FISHER DISCRIMINANT-BASED IMAGE RECOGNITION ALGORITHMS

One method of identifying images is to measure the similarity between images. This is accomplished by using measures such as the L_1 norm, L_2 norm, covariance, Mahalanobis distance, and correlation. These similarity measures can be calculated on the images in their original space or on the images projected into a new space. I discuss two alternative spaces in which these similarity measures may be calculated, the subspace created by the eigenvectors of the covariance matrix of the training data and the subspace created by the Fisher basis vectors of the data. Variations of these spaces will be discussed as well as the behavior of similarity measures within these spaces. Experiments are presented comparing recognition rates for different similarity measures and spaces using hand labeled imagery from two domains: human face recognition and classifying an image as a cat or a dog.

Wendy S. Yambor
Computer Science Department
Colorado State University
Fort Collins, CO 80523
Summer 2000

Acknowledgments

I thank my committee, Ross Beveridge, Bruce Draper, Micheal Kirby, and Adele Howe, for their support and knowledge over the past two years. Every member of my committee has been involved in some aspect of this thesis. It is through their interest and persuasion that I gained knowledge in this field.

I thank Jonathon Phillips for providing me with the results and images from the FERET evaluation. Furthermore, I thank Jonathon for patiently answering numerous questions.

Table of Contents

1. Introduction

Two image recognition systems are examined, eigenspace projection and Fisher discriminants. Each of these systems examines images in a subspace. The eigenvectors of the covariance matrix of the training data create the eigenspace. The basis vectors calculated by Fisher discriminants create the Fisher discriminants subspace. Variations of these subspaces are examined. The first variation is the selection of vectors used to create the subspaces. The second variation is the measurement used to calculate the difference between images projected into these subspaces. Experiments are performed to test hypotheses regarding the relative performance of subspace and difference measures.

Neither eigenspace projection nor Fisher discriminants are new ideas. Both have been examined by researches for many years. It is the work of these researches that has helped to revolutionize image recognition and bring face recognition to the point where it is now usable in industry.

1.1 Previous Work

Projecting images into eigenspace is a standard procedure for many appearance-based object recognition algorithms. A basic explanation of eigenspace projection is provided by [20]. Michael Kirby was the first to introduce the idea of the low-dimensional characterization of faces. Examples of his use of eigenspace projection can be found in [7,8,16]. Turk & Pentland worked with eigenspace projection for face recognition [21].

More recently Shree Nayar used eigenspace projection to identify objects using a turntable to view objects at different angles as explained in [11].

R.A. Fisher developed Fisher's linear discriminant in the 1930's [5]. Not until recently have Fisher discriminants been utilized for object recognition. An explanation of Fisher discriminants can be found in [4]. Swets and Weng used Fisher discriminants to cluster images for the purpose of identification in 1996 [18,19,23]. Belhumeur, Hespanha, and Kriegman also used Fisher discriminants to identify faces, by training and testing with several faces under different lighting [1].

1.2 A General Algorithm

An image may be viewed as a vector of pixels where the value of each entry in the vector is the grayscale value of the corresponding pixel. For example, an 8x8 image may be unwrapped and treated as a vector of length 64. The image is said to sit in N-dimensional space, where N is the number of pixels (and the length of the vector). This vector representation of the image is considered to be the original space of the image.

The original space of an image is just one of infinitely many spaces in which the image can be examined. Two specific subspaces are the subspace created by the eigenvectors of the covariance matrix of the training data and the basis vectors calculated by Fisher discriminants. The majority of subspaces, including eigenspace, do not optimize discrimination characteristics. Eigenspace optimizes variance among the images. The

exception to this statement is Fisher discriminants, which does optimize discrimination characteristics.

Although some of the details may vary, there is a basic algorithm for identifying images by projecting them into a subspace. First one selects a subspace on which to project the images. Once this subspace is selected, all training images are projected into this subspace. Next each test image is projected into this subspace. Each test image is compared to all the training images by a similarity or distance measure, the training image found to be most similar or closest to the test image is used to identify the test image.

1.3 Why Study These Subspaces?

Projecting images into subspaces has been studied for many years as discussed in the previous work section. The research into these subspaces has helped to revolutionize image recognition algorithms, specifically face recognition. When studying these subspaces an interesting question arises: under what conditions does projecting an image into a subspace improve performance. The answer to this question is not an easy one. What specific subspace (if any at all) improves performance depends on the specific problem. Furthermore, variations within the subspace also effect performance. For example, the selection of vectors to create the subspace and measures to decide which images are a closest match, both effect performance.

1.4 Organization of Following Sections

I discuss two alternative spaces commonly used to identify images. In chapter 2, I discuss eigenspaces. Eigenspace projection, also know as Karhunen-Loeve (KL) and Principal Component Analysis (PCA), projects images into a subspace such that the first orthogonal dimension of this subspace captures the greatest amount of variance among the images and the last dimension of this subspace captures the least amount of variance among the images. Two methods of creating an eigenspace are examined, the original method and a method designed for high-resolution images know as the snapshot method. In chapter 3, Fisher discriminants is discussed. Fisher discriminants project images such that images of the same class are close to each other while images of different classes are far apart. Two methods of calculating Fisher discriminants are examined. One method is the original method and the other method first projects the images into an orthonormal basis defining a subspace spanned by the training set.

Once images are projected into one of these spaces, a similarity measure is used to decide which images are closest matches. Chapter 4 discusses variations of these two methods, such as methods of selecting specific eigenvectors to create the subspace and similarity measures. In chapter 5, I discuss experiments performed on both these methods on two datasets. The first dataset is the Cat & Dog dataset, which was developed at Colorado State University. The second dataset is the FERET dataset, which was made available to me by Jonathan Phillips at the National Institute of Standard and Technology [10,12,13].

2. Eigenspace Projection

Eigenspace is calculated by identifying the eigenvectors of the covariance matrix derived from a set of training images. The eigenvectors corresponding to non-zero eigenvalues of the covariance matrix form an orthonormal basis that rotates and/or reflects the images in the N-dimensional space. Specifically, each image is stored in a vector of size N.

$$x^i = \begin{bmatrix} x_1^i & \dots & x_N^i \end{bmatrix}^T \qquad (1)$$

The images are mean centered by subtracting the mean image from each image vector[1].

$$\bar{x}^i = x^i - m \text{, where } m = \frac{1}{P}\sum_{i=1}^{P} x^i \qquad (2)$$

These vectors are combined, side-by-side, to create a data matrix of size NxP (where P is the number of images).

$$\bar{X} = \begin{bmatrix} \bar{x}^1 & | & \bar{x}^2 & | & \dots & | & \bar{x}^P \end{bmatrix} \qquad (3)$$

The data matrix X is multiplied by its transpose to calculate the covariance matrix.

$$\Omega = \bar{X}\bar{X}^T \qquad (4)$$

This covariance matrix has up to P eigenvectors associated with non-zero eigenvalues, assuming P<N. The eigenvectors are sorted, high to low, according to their associated eigenvalues. The eigenvector associated with the largest eigenvalue is the eigenvector

[1] The bar notation here is slightly nonstandard, but is intended to suggest the relationship to the mean. A complete glossary of symbols appears in Appendix I.

that finds the greatest variance in the images. The eigenvector associated with the second largest eigenvalue is the eigenvector that finds the second most variance in the images. This trend continues until the smallest eigenvalue is associated with the eigenvector that finds the least variance in the images.

2.1 Recognizing Images Using Eigenspace, Tutorial on Original Method

Identifying images through eigenspace projection takes three basic steps. First the eigenspace must be created using training images. Next, the training images are projected into the eigenspace. Finally, the test images are identified by projecting them into the eigenspace and comparing them to the projected training images.

1. Create Eigenspace

The following steps create an eigenspace.

1. **Center data:** Each of the training images must be centered. Subtracting the mean image from each of the training images centers the training images as shown in equation (2). The mean image is a column vector such that each entry is the mean of all corresponding pixels of the training images.
2. **Create data matrix:** Once the training images are centered, they are combined into a data matrix of size NxP, where P is the number of training images and each column is a single image as shown in equation (3).
3. **Create covariance matrix:** The data matrix is multiplied by its transpose to create a covariance matrix as shown in equation (4).

4. **Compute the eigenvalues and eigenvectors:** The eigenvalues and corresponding eigenvectors are computed for the covariance matrix.

$$\Omega V = \Lambda V \qquad (5)$$

here V is the set of eigenvectors associated with the eigenvalues Λ.

5. **Order eigenvectors:** Order the eigenvectors $v_i \in V$ according to their corresponding eigenvalues $l_i \in \Lambda$ from high to low. Keep only the eigenvectors associated with non-zero eigenvalues. This matrix of eigenvectors is the eigenspace V, where each column of V is an eigenvector.

$$V = \begin{bmatrix} v_1 & | & v_2 & | & \ldots & | & v_P \end{bmatrix} \qquad (6)$$

2. Project training images

Each of the centered training images ($\bar{x}^i$) is projected into the eigenspace. To project an image into the eigenspace, calculate the dot product of the image with each of the ordered eigenvectors.

$$\tilde{x}^i = V^T \bar{x}^i \qquad (7)$$

Therefore, the dot product of the image and the first eigenvector will be the first value in the new vector. The new vector of the projected image will contain as many values as eigenvectors.

3. Identify test images

Each test image is first mean centered by subtracting the mean image, and is then projected into the same eigenspace defined by V.

$$\bar{y}^i = y^i - m, \text{ where } m = \frac{1}{P}\sum_{i=1}^{p} x^i \qquad (8)$$

and

$$\tilde{y}^i = V^T \bar{y}^i \qquad (9)$$

The projected test image is compared to every projected training image and the training image that is found to be closest to the test image is used to identify the training image. The images can be compared using any number of similarity measures; the most common is the L_2 norm. I will discuss the different similarity measures in section 4.3.

The following is an example of identifying images through eigenspace projection. Let the four images in Figure 1 be training images and let the additional image in Figure 1 be a test image. The four training images and the mean image are:

$$x^1 = \begin{bmatrix} 225 \\ 229 \\ 48 \\ 251 \\ 33 \\ 238 \\ 0 \\ 255 \\ 217 \end{bmatrix} \quad x^2 = \begin{bmatrix} 10 \\ 219 \\ 24 \\ 255 \\ 18 \\ 247 \\ 17 \\ 255 \\ 2 \end{bmatrix} \quad x^3 = \begin{bmatrix} 196 \\ 35 \\ 234 \\ 232 \\ 59 \\ 244 \\ 243 \\ 57 \\ 226 \end{bmatrix} \quad x^4 = \begin{bmatrix} 255 \\ 223 \\ 224 \\ 255 \\ 0 \\ 255 \\ 249 \\ 255 \\ 235 \end{bmatrix} \quad m = \begin{bmatrix} 171.50 \\ 176.50 \\ 135.5 \\ 248.25 \\ 27.50 \\ 246.00 \\ 127.25 \\ 205.50 \\ 170.00 \end{bmatrix}$$

The centered images are:

$$\bar{x}^1 = \begin{bmatrix} 53.50 \\ 52.50 \\ -84.50 \\ 2.75 \\ 5.50 \\ -8.00 \\ 127.25 \\ 49.50 \\ 47.00 \end{bmatrix} \quad \bar{x}^2 = \begin{bmatrix} -161.50 \\ 42.50 \\ -108.50 \\ 6.75 \\ -9.50 \\ 1.00 \\ -110.25 \\ 49.50 \\ -168.00 \end{bmatrix} \quad \bar{x}^3 = \begin{bmatrix} 24.50 \\ -141.50 \\ 101.50 \\ -16.25 \\ 31.50 \\ -2.00 \\ 115.75 \\ -148.50 \\ 56.00 \end{bmatrix} \quad \bar{x}^4 = \begin{bmatrix} 83.50 \\ 46.50 \\ 91.50 \\ 6.75 \\ -27.50 \\ 9.00 \\ 121.75 \\ 9.49.50 \\ 65.00 \end{bmatrix}$$

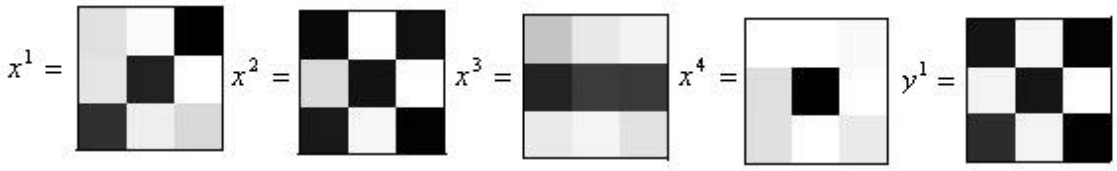

Figure 1. Four training images and one test image.

Combine all the centered training images into one data matrix:

$$\overline{\mathrm{X}} = \begin{bmatrix} 53.50 & -161.50 & 24.50 & 83.50 \\ 52.50 & 42.50 & -141.50 & 46.50 \\ -84.50 & -108.50 & 101.50 & 91.50 \\ 2.75 & 6.75 & -16.25 & 6.75 \\ 5.50 & -9.50 & 31.50 & -27.50 \\ -8.00 & 1.00 & -2.00 & 9.00 \\ -127.25 & -110.25 & 115.75 & 121.75 \\ 49.50 & 49.50 & -148.50 & 49.50 \\ 47.00 & -168.00 & 56.00 & 65.00 \end{bmatrix}$$

Calculate the covariance matrix:

$$\Omega = \overline{\mathrm{X}}\overline{\mathrm{X}}^{T} = \begin{bmatrix} 36517 & -3639 & 23129 & -778 & 304 & 113 & 24000 & -4851 & 36446 \\ -3639 & 26747 & -19155 & 3045 & -5851 & 324 & -22083 & 28017 & -9574 \\ 23129 & -19155 & 37587 & -1997 & 1247 & 1188 & 45603 & -20097 & 25888 \\ -778 & 3045 & -1996 & 363 & -746.5 & 78 & -2153 & 3217 & -1476 \\ 304 & -5851 & 1247 & -747 & 1869 & -364 & 645 & -6237 & 1831 \\ 113 & 324 & 1188 & 78 & -364 & 150 & 1772 & 396 & -71 \\ 24000 & -22083 & 45603 & -2153 & 645.5 & 1772 & 56569 & -22919 & 26937 \\ -4851 & 28017 & -20097 & 3218 & -6237 & 396 & -22919 & 29403 & -11088 \\ 36446 & -9574 & 25888 & -1476 & 1831 & -71 & 26937 & -11088 & 37794 \end{bmatrix}$$

The ordered non-zero eigenvectors of the covariance matrix and the corresponding eigenvalues are:

$$v_1 = \begin{bmatrix} 0.356 \\ -0.279 \\ 0.480 \\ -0.031 \\ 0.035 \\ 0.009 \\ 0.560 \\ -0.296 \\ 0.402 \end{bmatrix} \quad v_2 = \begin{bmatrix} -0.552 \\ -0.489 \\ 0.044 \\ -0.048 \\ 0.105 \\ -0.004 \\ 0.112 \\ 0.492 \\ -0.432 \end{bmatrix} \quad v_3 = \begin{bmatrix} -0.264 \\ 0.347 \\ 0.309 \\ 0.064 \\ -0.222 \\ 0.078 \\ 0.585 \\ 0.401 \\ -0.391 \end{bmatrix}$$

$$l_1 = 153520 \qquad l_2 = 50696 \qquad l_3 = 22781$$

The eigenspace is defined by the projection matrix

$$V = \begin{bmatrix} 0.356 & -0.552 & -0.264 \\ -0.279 & -0.489 & 0.347 \\ 0.480 & 0.044 & 0.309 \\ -0.031 & -0.048 & 0.064 \\ 0.035 & 0.105 & -0.222 \\ 0.009 & -0.004 & 0.078 \\ 0.560 & 0.112 & 0.585 \\ -0.296 & 0.492 & 0.401 \\ 0.402 & -0.432 & -0.391 \end{bmatrix}$$

The four centered training images projected into eigenspace are:

$$\tilde{x}^1 = V^T \bar{x}^1 = \begin{bmatrix} -103.09 \\ -117.31 \\ -96.57 \end{bmatrix} \quad \tilde{x}^2 = V^T \bar{x}^2 = \begin{bmatrix} -265.92 \\ 98.29 \\ 47.45 \end{bmatrix}$$

$$\tilde{x}^3 = V^T \bar{x}^3 = \begin{bmatrix} 229.76 \\ 125.90 \\ -46.14 \end{bmatrix} \quad \tilde{x}^4 = V^T \bar{x}^4 = \begin{bmatrix} 139.24 \\ -106.88 \\ 95.26 \end{bmatrix}$$

The test image viewed as a vector and the centered test image are:

$$y^1 = \begin{bmatrix} 20 \\ 244 \\ 44 \\ 246 \\ 21 \\ 244 \\ 4 \\ 255 \\ 2 \end{bmatrix} \qquad \bar{y}^1 = \begin{bmatrix} -151.5 \\ 67.5 \\ -88.5 \\ -2.25 \\ -6.5 \\ -2 \\ -123.25 \\ 49.5 \\ -168 \end{bmatrix}$$

The projected test image is:

$$\tilde{y}^1 = V^T \bar{y}^1 = \begin{bmatrix} -266.65 \\ 80.75 \\ 50.6 \end{bmatrix}$$

The L_2 norms are 296, 18, 508 and 449 of the test image y^1 and the training images x^1, x^2, x^3 and x^4 respectively. By comparing the L_2 norms, the second training image x^2 is found to be closest to the test image y^1, therefore the test image y^1 is identified as belonging to the same class of images as the second training image x^2. By viewing the original images, one sees image y^1 is most like x^2.

2.2 Tutorial for Snapshot Method of Eigenspace Projection

The method outlined above can lead to extremely large covariance matrices. For example, images of size 64x64 combine to create a data matrix of size 4096xP and a covariance matrix of size 4096x4096. This is a problem because calculating the covariance matrix and the eigenvectors/eigenvalues of the covariance is computationally demanding. It is known that for a NxM matrix the maximum number of non-zero eigenvectors the matrix can have is min(N-1,M-1) [6,7,20]. Since the number of training

images (P) is usually less than the number of pixels (N), the most eigenvectors/eigenvalues that can be found are P-1.

A common theorem in linear algebra states that the eigenvalues of $\overline{XX}^T$ and $\overline{X}^T\overline{X}$ are the same. Furthermore, the eigenvectors of $\overline{XX}^T$ are the same as the eigenvectors of $\overline{X}^T\overline{X}$ multiplied by the matrix $\overline{X}$ and normalized [6,7,20]. Using this theorem, the Snapshot method can be used to create the eigenspace from a PxP matrix rather than a NxN covariance matrix. The following steps should be followed.

1. **Center data:** (Same as original method)
2. **Create data matrix:** (Same as original method)
3. **Create covariance matrix:** The data matrix's transpose is multiplied by the data matrix to create a covariance matrix.

$$\Omega' = \overline{X}^T\overline{X} \qquad (10)$$

4. **Compute the eigenvalues and eigenvectors of O':** The eigenvalues and corresponding eigenvectors are computed for Ω'.

$$\Omega' V' = \Lambda' V' \quad (11)$$

5. **Compute the eigenvectors of $\overline{XX}^T$:** Multiply the data matrix by the eigenvectors.

$$\hat{V} = \overline{X}V' \qquad (12)$$

Divide the eigenvectors by their norm.

$$v_i = \frac{\hat{v}_i}{\left\|\hat{v}_i\right\|} \qquad (13)$$

6. **Order eigenvectors:** (Same as original method)

The following is the same example as used previously, but the eigenspace is calculated using the Snapshot method. The same training and test images will be used as shown in Figure 1. The revised covariance matrix is:

$$\Omega' = \overline{\mathrm{X}}^T \overline{\mathrm{X}} = \begin{bmatrix} 33712 & 11301 & -33998 & -115015 \\ 11301 & 82627 & -50914 & -43014 \\ -33998 & -50914 & 70771 & 14141 \\ -11015 & -43014 & 14141 & 39888 \end{bmatrix}$$

The ordered eigenvectors and corresponding non-zero eigenvalues of the revised covariance matrix are:

$$v_1' = \begin{bmatrix} -0.263 \\ -0.679 \\ 0.586 \\ 0.355 \end{bmatrix} \quad v_2' = \begin{bmatrix} 0.521 \\ -0.437 \\ -0.559 \\ 0.475 \end{bmatrix} \quad v_3' = \begin{bmatrix} -0.640 \\ 0.314 \\ -0.306 \\ 0.631 \end{bmatrix}$$

$$l_1 = 153520 \qquad l_2 = 50696 \qquad l_3 = 22781$$

The data matrix multiplied by the eigenvectors are:

$$\hat{v}_1 = \begin{bmatrix} 139.5734 \\ -109.108 \\ 187.906 \\ -12.435 \\ 13.699 \\ 3.452 \\ 219.448 \\ -116.108 \\ 157.593 \end{bmatrix} \quad \hat{v}_2 = \begin{bmatrix} 124.311 \\ 109.995 \\ -9.981 \\ 10.777 \\ -23.655 \\ 0.781 \\ -25.098 \\ 11.715 \\ 97.366 \end{bmatrix} \quad \hat{v}_3 = \begin{bmatrix} -39.787 \\ 52.380 \\ 46.675 \\ 9.591 \\ -33.493 \\ 11.725 \\ 88.213 \\ 60.533 \\ -58.978 \end{bmatrix}$$

Below are the normalized eigenvectors. Note that they are the same eigenvectors that were calculated using the original method.

$$v_1 \begin{bmatrix} 0.356 \\ -0.279 \\ 0.480 \\ -0.032 \\ 0.035 \\ 0.009 \\ 0.560 \\ -0.296 \\ 0.402 \end{bmatrix} \quad v_2 \begin{bmatrix} -0.552 \\ -0.489 \\ 0.044 \\ -0.048 \\ 0.105 \\ -0.004 \\ 0.112 \\ 0.492 \\ -0.432 \end{bmatrix} \quad v_3 \begin{bmatrix} -0.264 \\ 0.347 \\ 0.309 \\ 0.064 \\ -0.222 \\ 0.078 \\ 0.585 \\ 0.401 \\ -0.391 \end{bmatrix}$$

2.3 Variations

Centering the images by subtracting the mean image is one common method of modifying the original images. Another variant is to subtract the mean of each image from all of the pixel values for that image [20]. This variation simplifies the correlation calculation, since the images are already mean subtracted. Yet another variation is to normalize each image by dividing each pixel value by the norm of the image, so that the vector has a length of one [20]. This variation simplifies the covariance calculation to a dot product. An image cannot be both centered and normalized, since these actions counteract the one another. But an image can be centered and mean subtracted or mean subtracted and normalized. For all my work, I use only centered images.

3. Fisher Discriminants

Fisher discriminants group images of the same class and separates images of different classes. Images are projected from N-dimensional space (where N is the number of pixels in the image) to C-1 dimensional space (where C is the number of classes of images). For example, consider two sets of points in 2-dimensional space that are projected onto a single line (Figure 2a). Depending on the direction of the line, the points can either be mixed together (Figure 2b) or separated (Figure 2c). Fisher discriminants find the line that best separates the points. To identify a test image, the projected test image is compared to each projected training image, and the test image is identifed as the closest training image.

3.1 Fisher Discriminants Tutorial (Original Method)

As with eigenspace projection, training images are projected into a subspace. The test images are projected into the same subspace and identified using a similarity measure. What differs is how the subspace is calculated. Following are the steps to follow to find the Fisher discriminants for a set of images.

1. **Calculate the within class scatter matrix:** The within class scatter matrix measures the amount of scatter between items in the same class. For the i^{th} class, a scatter matrix (S_i) is calculated as the sum of the covariance matrices of the centered images in that class.

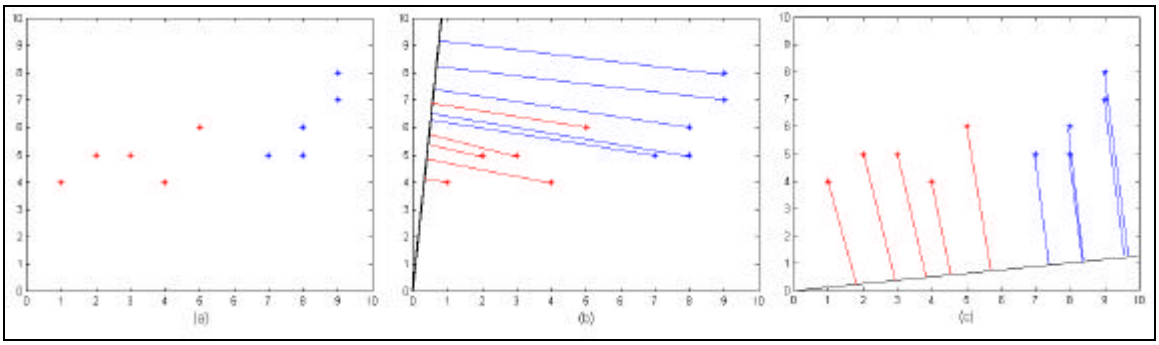

Figure 2. (a) Points in 2-dimensional space. (b) Points mixed when projected onto a line. (c) Points separated when projected onto a line.

$$S_i = \sum_{x \in X_i} (x - m_i)(x - m_i)^T \qquad (14)$$

where m_i is the mean of the images in the class. The within class scatter matrix (S_W) is the sum of all the scatter matrices.

$$S_W = \sum_{i=1}^{C} S_i \qquad (15)$$

where C is the number of classes.

2. **Calculate the between class scatter matrix:** The between class scatter matrix (S_B) measures the amount of scatter between classes. It is calculated as the sum of the covariance matrices of the difference between the total mean and the mean of each class.

$$S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^T \qquad (16)$$

where n_i is the number of images in the class, m_i is the mean of the images in the class and m is the mean of all the images.

3. **Solve the generalized eigenvalue problem:** Solve for the generalized eigenvectors (V) and eigenvalues (Λ) of the within class and between class scatter matrices.

$$S_B V = \Lambda S_W V \qquad (17)$$

4. **Keep first C-l eigenvectors:** Sort the eigenvectors by their associated eigenvalues from high to low and keep the first $C-1$ eigenvectors. These eigenvectors form the Fisher basis vectors.
5. **Project images onto Fisher basis vectors:** Project all the original (i.e. not centered) images onto the Fisher basis vectors by calculating the dot product of the image with each of the Fisher basis vectors. The original images are projected onto this line because these are the points that the line has been created to discriminate, not the centered images.

Following is an example of calculating the Fisher discriminants for a set of images. Let the twelve images in Figure 3 be training images. There are two classes; images $x^1 - x^6$ are in the first class and images $x^7 - x^{12}$ are in the second class. The training images viewed as vectors are: =

$$x^1 = \begin{bmatrix} 196 \\ 35 \\ 234 \\ 232 \\ 59 \\ 244 \\ 243 \\ 57 \\ 226 \end{bmatrix} \quad x^2 = \begin{bmatrix} 188 \\ 15 \\ 236 \\ 244 \\ 44 \\ 228 \\ 251 \\ 48 \\ 230 \end{bmatrix} \quad x^3 = \begin{bmatrix} 246 \\ 48 \\ 222 \\ 225 \\ 40 \\ 226 \\ 208 \\ 35 \\ 234 \end{bmatrix} \quad x^4 = \begin{bmatrix} 208 \\ 16 \\ 235 \\ 255 \\ 44 \\ 229 \\ 236 \\ 34 \\ 247 \end{bmatrix} \quad x^5 = \begin{bmatrix} 245 \\ 21 \\ 213 \\ 254 \\ 55 \\ 252 \\ 215 \\ 51 \\ 249 \end{bmatrix} \quad x^6 = \begin{bmatrix} 248 \\ 22 \\ 225 \\ 252 \\ 30 \\ 240 \\ 242 \\ 27 \\ 244 \end{bmatrix}$$

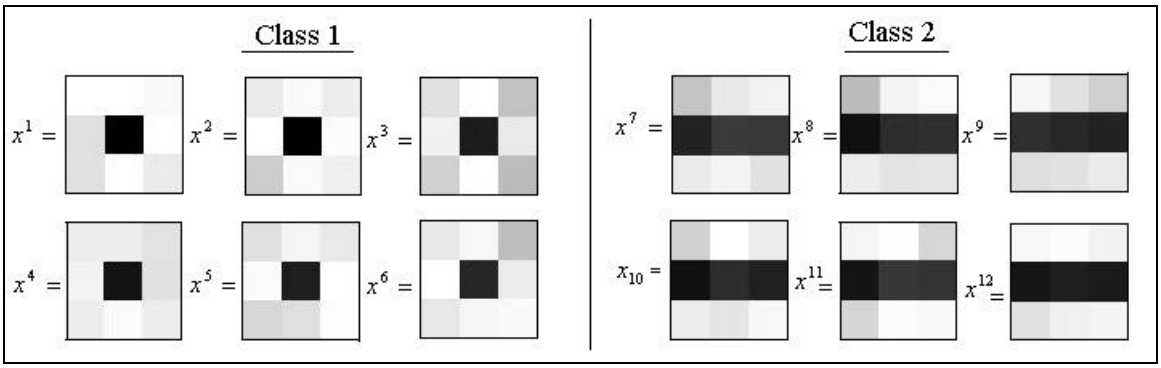

Figure 3. Twelve training images.

$$x^7 = \begin{bmatrix} 255 \\ 223 \\ 224 \\ 255 \\ 0 \\ 255 \\ 249 \\ 255 \\ 235 \end{bmatrix} \quad x^8 = \begin{bmatrix} 234 \\ 255 \\ 205 \\ 251 \\ 0 \\ 251 \\ 238 \\ 253 \\ 240 \end{bmatrix} \quad x^9 = \begin{bmatrix} 232 \\ 255 \\ 231 \\ 247 \\ 38 \\ 246 \\ 190 \\ 236 \\ 250 \end{bmatrix} \quad x^{10} = \begin{bmatrix} 255 \\ 241 \\ 208 \\ 255 \\ 28 \\ 255 \\ 194 \\ 234 \\ 188 \end{bmatrix} \quad x^{11} = \begin{bmatrix} 237 \\ 243 \\ 237 \\ 237 \\ 19 \\ 251 \\ 227 \\ 225 \\ 237 \end{bmatrix} \quad x^{12} = \begin{bmatrix} 224 \\ 251 \\ 215 \\ 245 \\ 31 \\ 222 \\ 233 \\ 255 \\ 254 \end{bmatrix}$$

The scatter matrices are:

$$S_1 = \begin{bmatrix} 3.81 & 0.59 & -1.09 & 0.14 & -0.60 & 0.39 & -1.69 & -0.83 & 0.77 \\ 0.59 & 0.83 & -0.15 & -0.73 & 0.05 & -0.09 & -0.63 & 0.01 & -0.28 \\ -1.09 & -0.15 & 0.42 & -0.06 & -0.01 & -0.26 & 0.62 & 0.03 & -0.23 \\ 0.14 & -0.72 & -0.06 & 0.79 & -0.10 & 0.21 & 0.28 & -0.17 & 0.48 \\ -0.60 & 0.05 & -0.01 & -0.10 & 0.55 & 0.28 & -0.07 & 0.56 & -0.13 \\ 0.39 & -0.09 & -0.26 & 0.21 & 0.28 & 0.55 & -0.09 & 0.28 & 0.14 \\ -1.69 & -0.63 & 0.62 & 0.28 & -0.07 & -0.09 & 1.46 & 0.11 & -0.28 \\ -0.83 & 0.01 & 0.03 & -0.17 & 0.56 & 0.28 & 0.11 & 0.68 & -0.26 \\ 0.77 & -0.28 & -0.23 & 0.48 & -0.13 & 0.14 & -0.28 & -0.26 & 0.46 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 0.63 & -0.51 & 0.27 & 0.08 & -0.64 & 0.36 & 0.79 & 0.17 & 0.21 \\ -0.51 & 0.74 & -0.14 & -0.16 & 0.45 & -0.34 & -0.59 & -0.09 & 0.50 \\ 0.27 & -0.14 & 0.82 & -0.28 & 0.25 & 0.06 & -0.12 & -0.44 & 0.59 \\ 0.08 & -0.16 & -0.28 & 0.24 & -0.18 & 0.16 & -0.01 & 0.22 & -0.41 \\ -0.64 & 0.45 & 0.25 & -0.18 & 1.31 & -0.47 & -1.54 & -0.49 & -0.00 \\ 0.36 & -0.34 & 0.06 & 0.16 & -0.47 & 0.79 & -0.17 & -0.30 & -0.84 \\ 0.79 & -0.59 & -0.12 & -0.01 & -1.54 & -0.17 & 2.94 & 1.00 & 1.13 \\ 0.17 & -0.09 & -0.44 & 0.22 & -0.49 & -0.30 & 1.00 & 0.84 & 0.56 \\ 0.21 & 0.50 & 0.59 & -0.41 & -0.00 & -0.84 & 1.13 & 0.56 & 2.82 \end{bmatrix}$$

The within class scatter matrix is:

$$S_W = S_1 + S_2 = \begin{bmatrix} 4.44 & 0.08 & -0.82 & 0.22 & -1.23 & 0.76 & -0.91 & -0.66 & 0.98 \\ 0.08 & 1.57 & -0.29 & -0.88 & 0.50 & -0.42 & -1.22 & -0.08 & 0.22 \\ -0.82 & -0.29 & 1.24 & -0.34 & 0.25 & -0.20 & 0.50 & -0.40 & 0.36 \\ 0.22 & -0.88 & -0.34 & 1.03 & -0.29 & 0.37 & 0.26 & 0.06 & 0.07 \\ -1.23 & 0.50 & 0.25 & -0.29 & 1.85 & -0.20 & -1.61 & 0.07 & -0.13 \\ 0.76 & -0.42 & -0.20 & 0.37 & -0.20 & 1.33 & -0.25 & -0.02 & -0.70 \\ -0.91 & -1.22 & 0.50 & 0.26 & -1.61 & -0.25 & 4.40 & 1.11 & 0.85 \\ -0.66 & -0.08 & -0.40 & 0.06 & 0.07 & -0.02 & 1.11 & 1.52 & 0.30 \\ 0.98 & 0.22 & 0.36 & 0.07 & -0.13 & -0.70 & 0.85 & 0.30 & 3.28 \end{bmatrix}$$

The mean of each class and the total mean are:

$$m_1 = \begin{bmatrix} 221.83 \\ 26.17 \\ 227.50 \\ 243.67 \\ 45.33 \\ 236.50 \\ 232.50 \\ 42.00 \\ 238.33 \end{bmatrix} \quad m_2 = \begin{bmatrix} 234.50 \\ 244.67 \\ 220.00 \\ 248.33 \\ 19.33 \\ 246.67 \\ 221.83 \\ 243.00 \\ 234.00 \end{bmatrix} \quad m = \begin{bmatrix} 228.17 \\ 135.42 \\ 223.75 \\ 246.00 \\ 32.33 \\ 241.58 \\ 227.17 \\ 142.50 \\ 236.17 \end{bmatrix}$$

The between class scatter matrix is:

$$S_B = \begin{bmatrix} 0.008 & 0.138 & -0.005 & 0.003 & -0.017 & 0.006 & -0.007 & 0.127 & -0.003 \\ 0.138 & 2.387 & -0.082 & 0.051 & -0.284 & 0.111 & -0.117 & 2.196 & -0.047 \\ -0.005 & -0.082 & 0.003 & -0.002 & 0.010 & -0.004 & 0.004 & -0.075 & 0.002 \\ 0.003 & 0.051 & -0.002 & 0.001 & -0.006 & 0.002 & -0.003 & 0.047 & -0.001 \\ -0.017 & -0.284 & 0.010 & -0.006 & 0.034 & -0.013 & 0.014 & -0.261 & 0.006 \\ 0.006 & 0.111 & -0.004 & 0.002 & -0.013 & 0.005 & -0.005 & 0.102 & -0.002 \\ -0.007 & -0.117 & 0.004 & -0.003 & 0.014 & -0.005 & 0.006 & -0.107 & 0.002 \\ 0.127 & 2.196 & -0.075 & 0.047 & -0.261 & 0.102 & -0.107 & 2.020 & -0.044 \\ -0.003 & -0.047 & 0.002 & -0.001 & 0.006 & -0.002 & 0.002 & -0.044 & 0.001 \end{bmatrix}$$

Since there are two classes, only one eigenvector is kept. The non-zero eigenvector and corresponding eigenvalue of $S_B V = l S_W V$ are:

$$v_1 = \begin{bmatrix} 0.18 \\ 0.51 \\ 0.48 \\ 0.59 \\ 0.04 \\ -0.12 \\ 0.08 \\ 0.23 \\ -0.22 \end{bmatrix} \quad l_1 = 291.45$$

The values of the images projected onto the first eigenvector are shown in Table 1. Figure 4 shows a plot of the points; clearly illustrating the separation between the two classes.

3.2 Fisher Discriminants Tutorial (Orthonormal Basis Method)

Two problems arise when using Fisher discriminants. First, the matrices needed for computation are very large, causing slow computation time and possible problems with

Table 1. The values of the images projected onto the first eigenvector.

	x^1	x^2	x^3	x^4	x^5	x^6		x^7	x^8	x^9	x^{10}	x^{11}	x^{12}
Class 1	259	255	256	256	254	257	**Class 2**	414	413	415	412	409	412

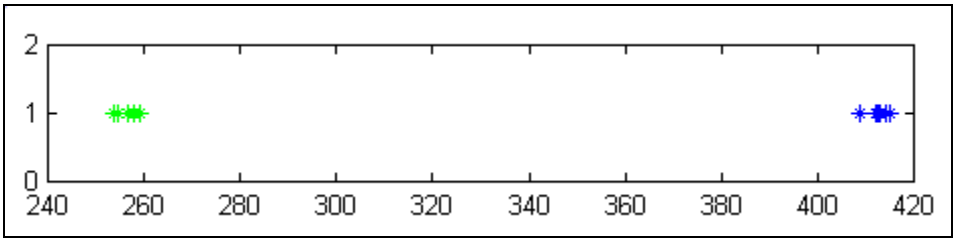

Figure 4. Plot of the images projected onto Fisher basis vectors.

numeric precision. Second, since there are fewer training images than pixels, the data matrix is rank deficient. It is possible to solve the eigenvectors and eigenvalues of a rank deficient matrix by using a generalize singular value decomposition routine, but a simplier solution exists. A simplier solution is to project the data matrix of training images into an orthonormal basis of size PxP (where P is the number of training images). This projection produces a data matrix of full rank that is much smaller and therefore decreases computation time. The projection also preserves information so the final outcome of Fisher discriminants is not affected. Following are the steps to follow to find the Fisher discriminants of a set of images by first projecting the images into any orthonormal basis.

1. **Compute means:** Compute the mean of the images in each class(m_i)and the total mean of all images (m).
2. **Center the images in each class:** Subtract the mean of each class from the images in that class.

$$\forall x \in X_i, X_i \in X, \hat{x} = x - m_i \qquad (18)$$

3. **Center the class means:** Subtract the total mean from the class means.

$$\hat{m} = m_i - m \qquad (19)$$

4. **Create a data matrix:** Combine the all images, side-by-side, into one data matrix.

5. **Find an orthonormal basis for this data matrix:** This can be accomplished by using a QR Orthogonal-triangular decomposition or by calculating the full set of eigenvectors of the covariance matrix of the training data. Let the orthonormal basis be U.

6. **Project all centered images into the orthonormal basis:** Create vectors that are the dot product of the image and the vectors in the orthonormal basis.

$$\tilde{\tilde{x}} = U^T \hat{x} \qquad (20)$$

7. **Project the centered means into the orthonormal basis:**

$$\tilde{\tilde{m}}_i = U^T \hat{m}_i \qquad (21)$$

8. **Calculate the within class scatter matrix:** The within class scatter matrix measures the amount of scatter between items within the same class. For the i^{th} class a scatter matrix (S_i) is calculated as the sum of the covariance matrices of the projected centered images for that class.

$$S_i = \sum_{x \in X_i} \tilde{\tilde{x}} \tilde{\tilde{x}}^T \qquad (22)$$

The within class scatter matrix (S_W) is the sum of all the scatter matrices.

$$S_W = \sum_{i=1}^{C} S_i \qquad (23)$$

where C is the number of classes.

9. **Calculate the between class scatter matrix:** The between class scatter matrix (S_B) measures the amount scatter between classes. It is calculated as the sum of the covariance matrices of the projected centered means of the classes, weighted by the number of images in each class.

$$S_B = \sum_{i=1}^{C} n_i \tilde{m}_i \tilde{m}_i^T \qquad (24)$$

where n_i is the number of images in the class.

10. **Solve the generalized eigenvalue problem:** Solve for the generalized eigenvectors (V) and eigenvalues (Λ) of the within class and between class scatter matrices.

$$S_B V = l S_W V \qquad (25)$$

11. **Keep the first C-l eigenvectors:** Sort the eigenvectors by their associated eigenvalues from high to low and keep the first $C-1$ eigenvectors. These are the Fisher basis vectors.
12. **Project images onto eigenvectors:** Project all the rotated original (i.e. Not centered) images onto the Fisher basis vectors. First project the original images into the orthonormal basis, and then project these projected images onto the Fisher basis vectors. The original rotated images are projected onto this line because these are the points that the line has been created to discriminate, not the centered images.

The same example as before will be calculated using the orthonormal basis. Let the twelve images in Figure 3 be training images. The training images viewed as vectors, the means of each class and the total mean are the same as in the previous example.

The centered images are:

$$\hat{x}^1 = \begin{bmatrix} -25.83 \\ 8.83 \\ 6.50 \\ -11.67 \\ 13.67 \\ 7.50 \\ 10.50 \\ 15.00 \\ -12.33 \end{bmatrix} \quad \hat{x}^2 = \begin{bmatrix} -33.83 \\ -11.17 \\ 8.50 \\ 0.33 \\ -1.33 \\ -8.50 \\ 18.50 \\ 6.00 \\ -8.33 \end{bmatrix} \quad \hat{x}^3 = \begin{bmatrix} 24.17 \\ 21.83 \\ -5.50 \\ -18.67 \\ -5.33 \\ -10.50 \\ -24.50 \\ -7.00 \\ -4.33 \end{bmatrix} \quad \hat{x}^4 = \begin{bmatrix} -13.83 \\ -10.17 \\ 7.50 \\ 11.33 \\ -1.33 \\ -7.50 \\ 3.50 \\ -8.00 \\ 8.67 \end{bmatrix} \quad \hat{x}^5 = \begin{bmatrix} 23.17 \\ -5.17 \\ -14.50 \\ 10.33 \\ 9.67 \\ 15.50 \\ -17.50 \\ 9.00 \\ 10.67 \end{bmatrix} \quad \hat{x}^6 = \begin{bmatrix} 26.17 \\ -4.17 \\ -2.50 \\ 8.33 \\ -15.33 \\ 3.50 \\ 9.50 \\ -15.00 \\ 5.67 \end{bmatrix}$$

$$\hat{x}^7 = \begin{bmatrix} 20.50 \\ -21.67 \\ 4.00 \\ 6.67 \\ -19.33 \\ 8.33 \\ 27.17 \\ 12.00 \\ 1.00 \end{bmatrix} \quad \hat{x}^8 = \begin{bmatrix} -0.50 \\ 10.33 \\ -15.00 \\ 2.67 \\ -19.33 \\ 4.33 \\ 16.17 \\ 10.00 \\ 6.00 \end{bmatrix} \quad \hat{x}^9 = \begin{bmatrix} -2.50 \\ 10.33 \\ 11.00 \\ -1.33 \\ 18.67 \\ -0.67 \\ -31.83 \\ -7.00 \\ 16.00 \end{bmatrix} \quad \hat{x}^{10} = \begin{bmatrix} -9.50 \\ -3.67 \\ -12.00 \\ 6.67 \\ 8.67 \\ 8.33 \\ -27.83 \\ -9.00 \\ -46.00 \end{bmatrix} \quad \hat{x}^{11} = \begin{bmatrix} 2.50 \\ -1.67 \\ 17.00 \\ -11.33 \\ -0.33 \\ 4.33 \\ 5.17 \\ -18.00 \\ 3.00 \end{bmatrix} \quad \hat{x}^{12} = \begin{bmatrix} -10.50 \\ 6.33 \\ -5.00 \\ -3.33 \\ 11.67 \\ -24.67 \\ 11.17 \\ 12.00 \\ 20.00 \end{bmatrix}$$

The centered class means are:

$$\hat{m}_1 = \begin{bmatrix} -6.33 \\ -109.25 \\ 3.75 \\ -2.33 \\ 13.00 \\ -5.08 \\ 5.33 \\ -100.50 \\ 2.17 \end{bmatrix} \quad \hat{m}_2 = \begin{bmatrix} 6.33 \\ 109.25 \\ -3.75 \\ 2.33 \\ -13.00 \\ 5.08 \\ -5.33 \\ 100.50 \\ -2.17 \end{bmatrix}$$

The orthonormal basis calculated by eigenspace projection is:

$$U = \begin{bmatrix} -0.35 & 0.80 & -0.05 & -0.03 & 0.07 & -0.16 & -0.38 & 0.14 & -0.21 \\ -0.25 & -0.09 & 0.34 & -0.39 & 0.44 & -0.09 & 0.45 & 0.13 & -0.50 \\ 0.13 & -0.13 & 0.16 & 0.61 & 0.13 & -0.42 & -0.15 & -0.37 & -0.46 \\ 0.07 & 0.11 & -0.21 & 0.10 & -0.55 & 0.44 & 0.25 & 0.02 & -0.61 \\ -0.22 & -0.39 & 0.25 & 0.01 & -0.41 & -0.26 & -0.33 & 0.61 & -0.12 \\ -0.09 & 0.12 & -0.38 & 0.01 & -0.27 & -0.66 & 0.55 & 0.06 & 0.14 \\ 0.81 & 0.17 & -0.08 & -0.06 & 0.21 & -0.12 & -0.02 & 0.48 & -0.13 \\ 0.24 & -0.05 & 0.05 & -0.67 & -0.30 & -0.28 & -0.30 & -0.46 & -0.16 \\ 0.17 & 0.36 & 0.77 & 0.10 & -0.32 & -0.01 & 0.27 & -0.06 & 0.24 \end{bmatrix}$$

The centered images projected into the orthonormal basis are:

$$\tilde{x}^1 = \begin{bmatrix} 13.07 \\ -31.39 \\ -1.15 \\ -11.44 \\ 3.31 \\ -18.46 \\ 1.56 \\ 2.67 \\ -2.14 \end{bmatrix} \quad \tilde{x}^2 = \begin{bmatrix} 31.66 \\ -27.65 \\ -5.32 \\ 4.50 \\ 1.14 \\ 4.91 \\ -1.93 \\ -3.89 \\ 2.01 \end{bmatrix} \quad \tilde{x}^3 = \begin{bmatrix} -35.92 \\ 11.31 \\ 9.95 \\ -8.87 \\ 24.40 \\ 1.89 \\ -5.71 \\ -4.29 \\ 0.52 \end{bmatrix} \quad \tilde{x}^4 = \begin{bmatrix} 12.47 \\ -6.04 \\ 4.71 \\ 15.99 \\ -7.86 \\ 11.87 \\ 3.36 \\ -2.21 \\ -0.47 \end{bmatrix} \quad \tilde{x}^5 = \begin{bmatrix} -21.62 \\ 20.47 \\ -0.89 \\ -10.14 \\ -26.12 \\ -5.75 \\ -0.62 \\ 1.62 \\ 2.58 \end{bmatrix} \quad \tilde{x}^6 = \begin{bmatrix} 0.34 \\ 33.29 \\ -7.31 \\ 9.96 \\ 5.12 \\ 5.54 \\ 3.35 \\ 6.10 \\ -2.50 \end{bmatrix}$$

$$\tilde{x}^7 = \begin{bmatrix} 27.79 \\ 31.50 \\ -18.05 \\ 1.17 \\ -3.78 \\ -7.32 \\ -9.43 \\ -5.16 \\ -1.26 \end{bmatrix} \quad \tilde{x}^8 = \begin{bmatrix} 16.17 \\ 13.28 \\ -2.16 \\ -20.17 \\ 6.27 \\ 4.06 \\ 14.82 \\ -1.94 \\ 0.77 \end{bmatrix} \quad \tilde{x}^9 = \begin{bmatrix} -29.04 \\ -11.15 \\ 25.17 \\ 11.21 \\ -10.62 \\ -4.35 \\ 4.15 \\ -4.82 \\ -1.98 \end{bmatrix} \quad \tilde{x}^{10} \begin{bmatrix} -32.09 \\ -28.19 \\ -38.76 \\ -1.89 \\ -1.66 \\ 8.19 \\ -2.02 \\ 2.03 \\ -0.33 \end{bmatrix} \quad \tilde{x}^{11} \begin{bmatrix} 1.07 \\ 2.14 \\ 3.68 \\ 22.01 \\ 12.41 \\ -10.83 \\ 1.44 \\ 4.30 \\ 2.99 \end{bmatrix} \quad \tilde{x}^{12} = \begin{bmatrix} 16.12 \\ -7.59 \\ 30.11 \\ -12.33 \\ -2.63 \\ 10.26 \\ -8.96 \\ 5.59 \\ -0.18 \end{bmatrix}$$

The centered means projected into the orthonormal basis are:

$$\tilde{m}_1 = \begin{bmatrix} 7.87 \\ 5.15 \\ -33.76 \\ 111.83 \\ -20.01 \\ 34.87 \\ -25.06 \\ 40.03 \\ 68.88 \end{bmatrix} \quad \tilde{m}_2 = \begin{bmatrix} -7.87 \\ -5.15 \\ 33.76 \\ -111.83 \\ 20.01 \\ -34.87 \\ 25.06 \\ -40.03 \\ -68.88 \end{bmatrix}$$

The within class scatter matrix is:

$$S_W = \begin{bmatrix} 6.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.59 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3.59 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.86 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.71 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.95 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.47 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.19 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04 \end{bmatrix}$$

Notice that the within class scatter matrix is a diagonal matrix and the values along the diagonal are the eigenvalues associated with the eigenvectors used to create the orthonormal basis. This occurs because the images are projected into this orthonormal basis before calculating the within class scatter matrix. Therefore each projected image is orthogonal to all other projected images.

The between class scatter matrix is:

$$S_B = \begin{bmatrix} 0.01 & 0.01 & -0.05 & 0.18 & -0.03 & 0.05 & -0.04 & 0.06 & 0.11 \\ 0.01 & 0.00 & -0.03 & 0.12 & -0.02 & 0.04 & -0.03 & 0.04 & 0.07 \\ -0.05 & -0.03 & 0.23 & -0.76 & 0.14 & -0.24 & 0.17 & -0.27 & -0.47 \\ 0.18 & 0.12 & -0.76 & 2.50 & -0.45 & 0.78 & -0.56 & 0.90 & 1.54 \\ -0.03 & -0.02 & 0.14 & -0.45 & 0.08 & -0.14 & 0.10 & -0.16 & -0.28 \\ 0.05 & 0.04 & -0.24 & 0.78 & -0.14 & 0.24 & -0.17 & 0.28 & 0.48 \\ -0.04 & -0.03 & 0.17 & -0.56 & 0.10 & -0.17 & 0.13 & -0.20 & -0.35 \\ 0.06 & 0.04 & -0.27 & 0.90 & -0.16 & 0.28 & -0.20 & 0.32 & 0.55 \\ 0.11 & 0.07 & -0.47 & 1.54 & -0.28 & 0.48 & -0.35 & 0.55 & 0.95 \end{bmatrix}$$

Since there are two classes, only one eigenvector is kept. The non-zero eigenvector and corresponding eigenvalue of $S_B V = l S_W V$ are:

$$v_1 = \begin{bmatrix} 0.0007 \\ 0.0005 \\ -0.0050 \\ 0.0322 \\ -0.0063 \\ 0.0196 \\ -0.0283 \\ 0.1113 \\ 0.9926 \end{bmatrix} \qquad l_1 = 291.4492$$

The values of the rotated images projected onto the first eigenvector are shown in Table 2. Figure 5 shows a plot of the points; you can clearly see the separation between the two classes.

Table 2. The values of the images projected onto the first eigenvector.

	x^1	x^2	x^3	x^4	x^5	x^6		x^7	x^8	x^9	x^{10}	x^{11}	x^{12}
Class 1	-259	-255	-256	-256	-254	-257	**Class 2**	-414	-413	-415	-412	-409	-412

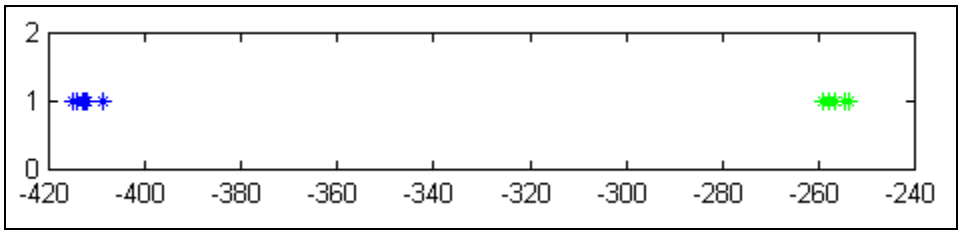

Figure 5. Plot of the images projected onto Fisher basis vectors.

4. Variations

4.1 Eigenvector Selection

Until this point, when creating a subspace using eigenspace projection we use all eigenvectors associated with non-zero eigenvalues. The computation time of eigenspace projection is directly proportional to the number of eigenvectors used to create the eigenspace. Therefore by removing some portion of the eigenvectors computation time is decrease. Furthermore, by removing additional eigenvectors that do not contribute to the classification of the image, performance can be improved. Many variations of eigenvector selection have been considered; I will discuss five. These may be applied either alone or as part of Fisher discriminants.

1. **Standard eigenspace projection:** All eigenvectors corresponding to non-zero eigenvalues are used to create the subspace.
2. **Remove the last 40% of the eigenvectors:** Since the eigenvectors are sorted by the corresponding descending eigenvalues, this method removes the eigenvectors that find the least amount of variance among the images. Specifically, 40% of the eigenvectors that find the least amount of variance are removed [10].
3. **Energy dimension:** Rather than use a standard cutoff for all subspaces, this method uses the minimum number of eigenvectors to guarantee that energy (e) is greater than

a threshold. A typical threshold is 0.9. The energy of the i^{th} eigenvector is the ratio of the sum of the first *i* eigenvalues over the sum of all the eigenvalues [7]

$$e_i = \frac{\sum_{j=1}^{i} l_j}{\sum_{j=1}^{k} l_j} \qquad (30)$$

4. **Stretching dimension:** Another method of selecting eigenvectors based on the information provided by the eigenvalues is to calculate the stretch (s) of an eigenvector. The stretch of the i^{th} eigenvector is the ratio of the i^{th} eigenvalue (l_i) over the maximum eigenvalue (l_1) [7]. A common threshold for the stretching dimension is 0.01.

$$s_i = \frac{l_i}{l_1} \qquad (31)$$

5. **Removing the first eigenvector:** The previous three methods assume that the information in the last eigenvectors work against classification. This method assumes that information in the first eigenvector works against classification. For example, lighting causes considerable variation in otherwise identical images. Hence, this method removes the first eigenvector [10].

Figure 6 shows the values for energy and stretching on the FERET dataset.

4.2 Ordering Eigenvectors by Like-Image Difference

Ideally, two images of the same person should project to the same point in eigenspace. Any difference between the points is unwanted variation. On the other hand, two images of different subjects should project to points that are as widely separated as possible. To

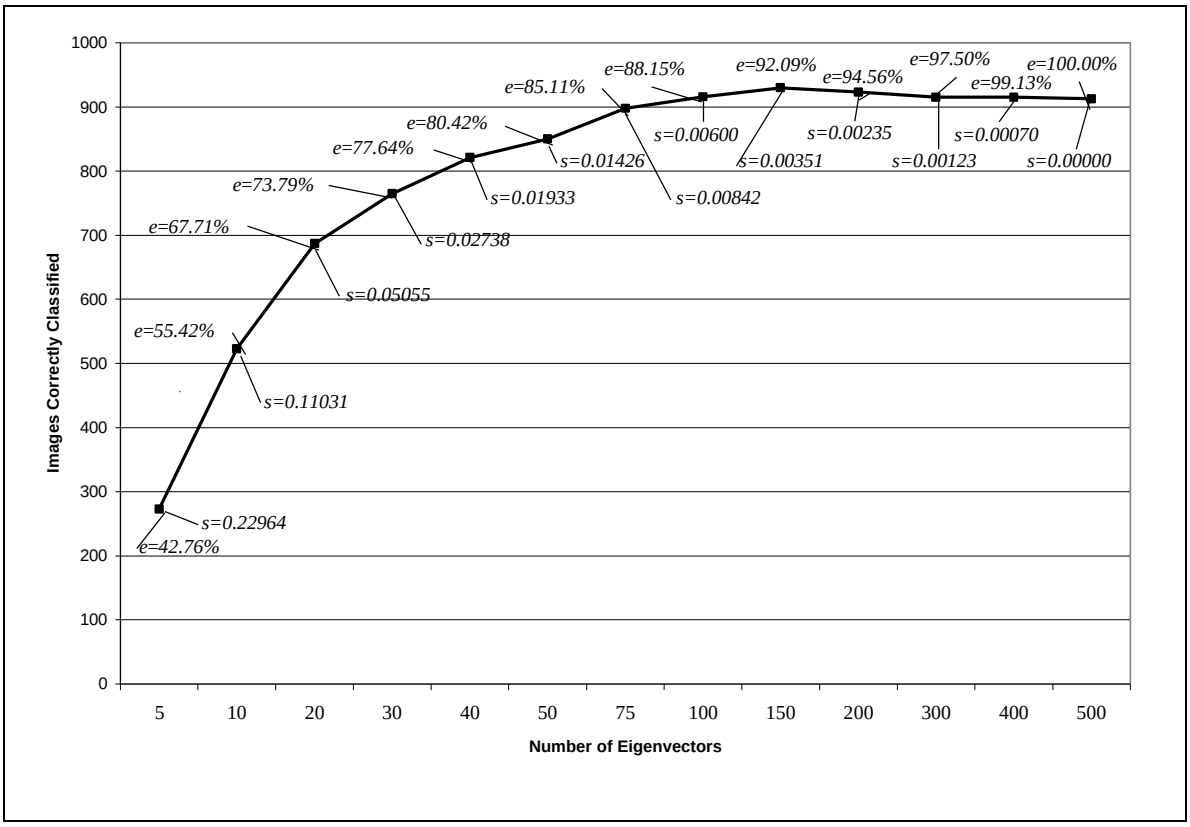

Figure 6. Example of Energy (e) and Stretching (s) dimension of a specific dataset.

capture this intuition and use it to order eigenvectors, we define a like-image difference (w) for each of the k eigenvectors [22].

To define w, we will work with pairs of images of the same people projected into eigenspace. Let X be training images and Y images of the corresponding people in the test set ordered such that $x_j \in X$ and $y_j \in Y$ are images of the same person. Define w as follows:

$$w_i = \frac{d_i}{l_i} \text{ where } d = \sum_{j=1}^{k} \left| x_j - y_j \right| \qquad (28)$$

When a difference between images that ought to match is large relative to the variance for the dimension l_i then w_i is large. Conversely, when the difference between images that ought to match is small relative to the variance, w_i is small. Since the goal is to select eigenvectors that bring similar images close to each other, we rank the eigenvectors in order of ascending w_i and remove some number of the last eigenvectors.

4.3 Similarity & Distance Measures

Once images are projected into a subspace, there is the task of determining which images are most like one another. There are two ways in general to determine how alike images are. One is to measure the distance between the images in N-dimensional space. The second way is to measure how similar two images are. When measuring distance, one wishes to minimize distance, so two images that are alike produce a small distance. When measuring similarity, one wishes to maximize similarity, so that two like images produce a high similarity value. There are many possible similarity and distance measures; I will discuss five.

***L_1* norm:** The L_1 norm is also known as the city block norm or the sum norm. It sums up the absolute difference between pixels[6,10]. The L_1 norm of an image A and an image B is:

$$L_1(A,B) = \sum_{i=1}^{N} |A_i - B_i| \qquad (29)$$

The L_1 norm is a distance measure. Figure 7 shows the L_1 distance between two vectors.

***L_2* norm:** The L_2 norm is also known as the Euclidean norm or the Euclidean distance when its square root is calculated. It sums up the squared difference between pixels [6,10,17]. The L_2 norm of an image A and an image B is:

$$L_2(A,B) = \sum_{i=1}^{N} (A_i - B_i)^2 \qquad (30)$$

The L_2 norm is a distance measure. Figure 7 shows the L_2 distance between two vectors.

Covariance: Covariance is also known as the angle measure. It calculates the angle between two normalized vectors. Taking the dot product of the normalized vectors performs this calculation [10,17]. The covariance between images A and B is:

$$\text{cov}(A,B) = \frac{A}{\|A\|} \bullet \frac{B}{\|B\|} \qquad (31)$$

Covariance is a similarity measure. By negating the covariance value, it becomes a distance measure [10]. Figure 7 shows the covariance between two vectors.

Mahalanobis distance: The Mahalanobis distance calculates the product of the pixels and the eigenvalue of a specific dimension and sums all these products [10]. The Mahalanobis distance between an image A and an image B is:

$$Mah(A,B) = -\sum_{i=1}^{N} A_i B_i C_i \qquad (32)$$

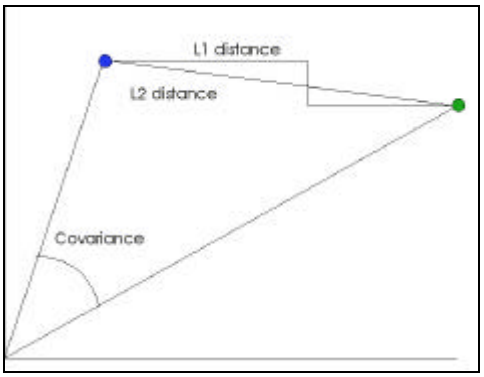

Figure 7. L1 distance, L2 distance and covariance between two vectors

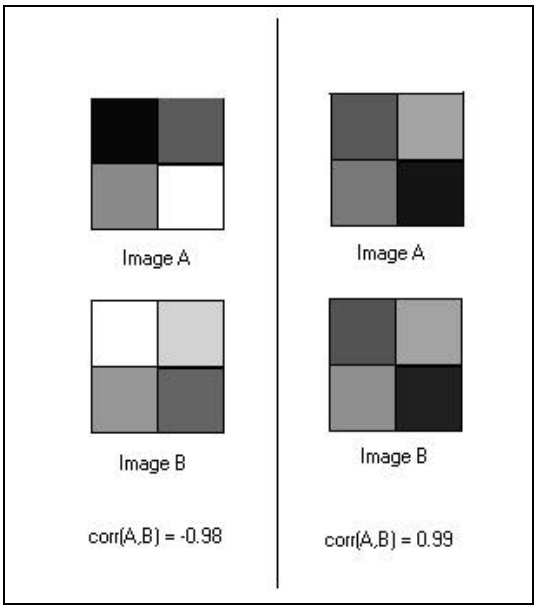

Figure 8. Two images with a negative correlation and two that correlate well

$$\text{where } C_i = \frac{1}{\sqrt{l}_i} \qquad (33)$$

Mahalanobis distance is a distance measure.

Correlation: Correlation measures the rate of change between the pixels of two images. It produces a value ranging from –1 to 1, where a value of –1 indicates the images are opposites of each other and a value of 1 indicates that the images are identical [17]. The correlation between an image A and an image B is:

$$corr(A,B) = \sum_{i=1}^{N} \frac{(A_i - m_A)(B_i - m_B)}{s_A s_B} \qquad (34)$$

where m_A is the mean of A and s_A is the standard deviation of A. Figure 8 shows an example of two images with a negative correlation and two that correlate well.

4.4 Are Similarity Measures the Same Inside and Outside of Eigenspace?

An eigenspace consisting of all eigenvectors associated with non-zero eigenvalues is an orthonormal basis. An orthonormal basis is a set of vectors where the dot product of any two distinct vectors is zero and the length of every vector is one. Orthonormal bases have the property that any image that was used to create the orthonormal basis can be projected into the full orthonormal basis with no loss of information. This means that the image can be projected into the orthonormal basis and then converted back to the original image. For example, let U be an orthonormal basis and let A be an image used to create U. Then $A' = U^T A$, where A' is the image A projected into U. A can be recovered by multiplying by U, $A = UA'$.

Given the fact that no information is lost when projecting specific images into an orthonormal basis, do the values of the similarity measures change? The answer is that it depends on the similarity measure. The L_1 norm and correlation produce different values in the two spaces. Mahalanobis distance is typically only used in conjunction with eigenspace. The L_2 norm and covariance do produce the same value in both spaces; I will prove this.

Theorem 4.1: The L_2 norm produces the same value on a pair of unprojected vectors and on a pair of projected vectors.

$$L_2(A, B) = L_2(U^T A, U^T B) \qquad (35)$$

Proof: Let U be an orthonormal basis. Let A be a vector such that $A' = U^T A$. Let B be a vector such that $B' = U^T B$. Now the L_2 norm of $A - B$ is defined in equation (30) and is the same as:

$$(A-B)^T (A-B) \qquad (36)$$

The L_2 norm of $(A' - B')$ is defined as:

$$\begin{aligned}
&L_2(A', B') = \sum_{i=1}^{N} (A_i' - B_i')^2 = (A' - B')^T (A' - B') \\
&= A'^T A' - A'^T B' - B'^T A' - B'^T B' \\
&= (U^T A)^T (U^T A) - (U^T A)^T (U^T B) - (U^T B)^T (U^T A) - (U^T B)^T (U^T B) \\
&= A^T U U^T A - A^T U U^T B - B^T U U^T A - B^T U U^T B \\
&= A^T A - A^T B - B^T A - B^T B \\
&= (A-B)^T (A-B) = \sum_{i=1}^{N} (A_i - B_i)^2 = L_2(A, B)
\end{aligned}$$

Hence, the L_2 norm produces the same value on unprojected vectors and on projected vectors.

Theorem 4.2: Covariance produces the same value on a pair of unprojected vectors and on a pair of projected vectors.

$$\text{cov}(A, B) = \text{cov}(U^T A, U^T B) \qquad (37)$$

Proof: Let U be an orthonormal basis. Let A be a vector such that $A' = U^T A$ and $A = UA'$. Let B be a vector such that $B' = U^T B$ and $B = UB'$. The covariance of A and B is defined in equation (31) and the covariance of A' and B' is defined as:

$$\text{cov}(A', B') = \frac{A'}{\|A'\|} \bullet \frac{B'}{\|B'\|} = \frac{A' \bullet B'}{\|A'\|\|B'\|} = \frac{(U^T A) \bullet B'}{\|U^T A\|\|U^T B\|} = \frac{(U^T A)^T B'}{\|U^T A\|\|U^T B\|} = \frac{A^T U B'}{\|U^T A\|\|U^T B\|} \qquad (38)$$

It is known that $B = UB'$, so

$$\text{cov}(A,B) = \frac{A^T B}{\left\|U^T A\right\|\left\|U^T B\right\|} \qquad (39)$$

By theorem 4.1 $\|A\| = \left\|U^T A\right\| = \|A'\|$ and $\|B\| = \left\|U^T B\right\| = \|B'\|$. So,

$$\text{cov}(A',B') = \frac{A^T B}{\|A\|\|B\|} = \text{cov}(A,B) \qquad (41)$$

Hence, covariance produces the same value on unprojected vectors and on projected vectors.

I will illustrate how each measure behaves with an example. Consider two vectors, $A = \begin{bmatrix} 1 \\ 5 \\ 4 \end{bmatrix}, B = \begin{bmatrix} 7 \\ 5 \\ 2 \end{bmatrix}$. Project these two points into the orthonormal basis

$$U = \begin{bmatrix} 0.7033 & -0.6594 & 0.2659 \\ 0.6189 & 0.3837 & -0.6854 \\ 0.3499 & 0.6465 & 0.6779 \end{bmatrix}.$$

$$A' = U^T A = \begin{bmatrix} 0.7033 & -0.6594 & 0.2659 \\ 0.6189 & 0.3837 & -0.6854 \\ 0.3499 & 0.6465 & 0.6779 \end{bmatrix}\begin{bmatrix} 1 \\ 5 \\ 4 \end{bmatrix} = \begin{bmatrix} 5.1973 \\ 3.8453 \\ -0.4497 \end{bmatrix}$$

$$B' = U^T B = \begin{bmatrix} 0.7033 & -0.6594 & 0.2659 \\ 0.6189 & 0.3837 & -0.6854 \\ 0.3499 & 0.6465 & 0.6779 \end{bmatrix}\begin{bmatrix} 7 \\ 5 \\ 2 \end{bmatrix} = \begin{bmatrix} 8.7169 \\ -1.4040 \\ -0.2103 \end{bmatrix}$$

Figure 9 shows points A, B, A' and B'.

***L_2* norm:** The L_2 norm produces the same value on unprojected vectors and on projected vectors. Examine the example.

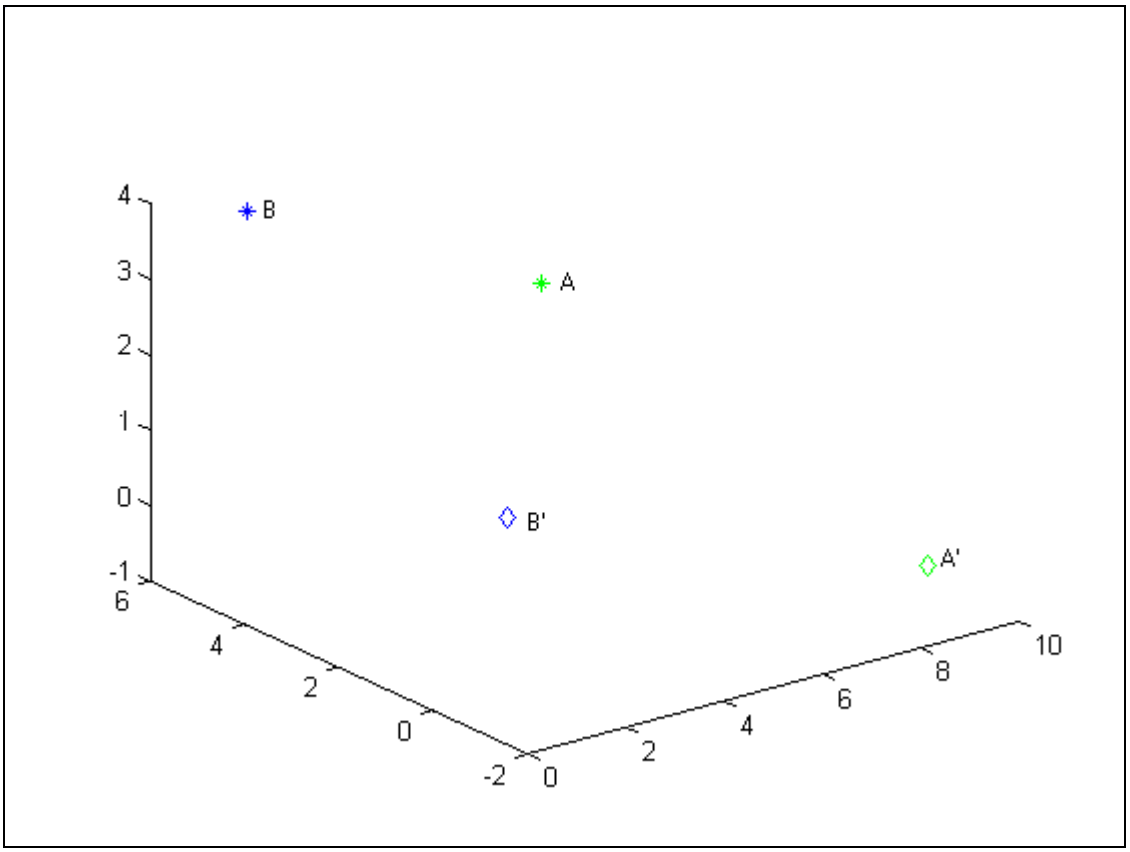

Figure 9. Plot of points A, B, A' and B'.

The L_2 norm of $A-B$ is:

$$L_2(A-B)=\sum_{i=1}^{2}(A-B)^2=(1-7)^2+(5-5)^2+(4-2)^2$$
$$=36+0+4=40$$

The L_2 norm of $A'-B'$ is:

$$L_2(A'-B')=\sum_{i=1}^{2}(A'-B')^2=(5.1973-8.7169)^2+(3.8453-(-1.4040))^2+((-0.4497)-(-0.2103))^2$$
$$=12.3876+27.5552+0.0573=40.0$$

This example shows that for this specific case they are the same, and the above proof covers the general case.

Covariance: Covariance produces the same value on unprojected vectors and on projected vectors. Examine the same example.

The covariance between A and B is:

$$\text{cov}(A,B) = \frac{A}{\|A\|} \bullet \frac{B}{\|B\|} = \frac{\begin{bmatrix}1\\5\\4\end{bmatrix}}{\left\|\begin{matrix}1\\5\\4\end{matrix}\right\|} \bullet \frac{\begin{bmatrix}7\\5\\2\end{bmatrix}}{\left\|\begin{matrix}7\\5\\2\end{matrix}\right\|} = \frac{\begin{bmatrix}1\\5\\4\end{bmatrix}}{6.4807} \bullet \frac{\begin{bmatrix}7\\5\\2\end{bmatrix}}{8.8318} = \begin{bmatrix}0.1543\\0.7715\\0.6172\end{bmatrix} \bullet \begin{bmatrix}0.7926\\0.5661\\0.2265\end{bmatrix}$$

$$= \begin{bmatrix}0.1543 & 0.7715 & 0.6172\end{bmatrix} \begin{bmatrix}0.7926\\0.5661\\0.2265\end{bmatrix} = 0.6989$$

The covariance between A' and B' is:

$$\text{cov}(A',B') = \frac{A'}{\|A'\|} \bullet \frac{B'}{\|B'\|} = \frac{\begin{bmatrix}5.1973\\3.8453\\-0.4497\end{bmatrix}}{\left\|\begin{matrix}5.1973\\3.8453\\-0.4497\end{matrix}\right\|} \bullet \frac{\begin{bmatrix}8.7169\\-1.4040\\-0.2103\end{bmatrix}}{\left\|\begin{matrix}8.7169\\-1.4040\\-0.2103\end{matrix}\right\|} = \frac{\begin{bmatrix}5.1973\\3.8453\\-0.4497\end{bmatrix}}{6.4807} \bullet \frac{\begin{bmatrix}8.7169\\-1.4040\\-0.2103\end{bmatrix}}{8.8318}$$

$$= \begin{bmatrix}0.8020\\0.5933\\-0.0694\end{bmatrix} \bullet \begin{bmatrix}0.9870\\-0.1590\\-0.0238\end{bmatrix} = \begin{bmatrix}0.8020 & 0.5933 & -0.0694\end{bmatrix} \begin{bmatrix}0.9870\\-0.1590\\-0.0238\end{bmatrix} = 0.6989$$

This example shows that for this specific case they are the same and the above proof covers the general case.

***L_1* norm:** The L_1 norm does not produce the same value on unprojected vectors and on projected vectors. Intuitively, two points that are located diagonal to each other will

produce a larger L_1 distance than the same points rotated to be horizontal to each other. This can be proven by an example.

The L_1 norm of $A - B$ is:

$$L_1(A-B) = \sum_{i=1}^{2} |A_i - B_i| = |1-7| + |5-5| + |4-2| = 6 + 0 + 2 = 8$$

The L_1 norm of $A' - B'$ is:

$$L_1(A'-B') = \sum_{i=1}^{2} |A'_i - B'_i|$$
$$= |5.1973 - 8.7169| + |3.8453 - (-1.4040)| + |(-0.4497) - (-0.2103)|$$
$$= 3.5197 + 5.2493 + 0.2393 = 9.0083$$

As you can see, these two norms are not the same; hence the L_1 norm does not produce the same value on unprojected vectors and on projected vectors.

Correlation: Correlation does not produce the same value on unprojected vectors and on projected vectors. Intuitively, Correlation is effected by the mean of the images. As images are rotated their mean changes, therefore their correlation changes. This can be proven by an example.

The correlation between A and B is:

$$m_A = 3.3333,\ m_B = 4.6667,\ s_A = 2.0817,\ s_B = 2.5166$$

$$corr(A,B) = \sum_{i=1}^{N} \frac{(A_i - m_A)(B_i - m_B)}{s_A s_B}$$

$$= \frac{(1-3.3333)(7-4.6667)}{2.0817*2.5166} + \frac{(5-3.3333)(5-4.6667)}{2.0817*2.5166} + \frac{(4-3.3333)(2-4.6667)}{2.0817*2.5166}$$

$$= \frac{(-2.3333)*2.3333 + 1.6667*0.3333 + 0.6667*-2.6667}{2.0817*2.5166}$$

$$= \frac{-5.4444 + 0.5556 + -1.7778}{5.2387} = \frac{-6.6667}{5.2387} = -1.2726$$

The correlation between A' and B' is:

$$m_{A'} = 2.8643, m_{B'} = 2.3675, s_{A'} = 2.9485, s_{B'} = 5.5310$$

$$corr(A',B') = \sum_{i=1}^{N} \frac{(A_i' - m_{A'})(B_i' - m_{B'})}{s_{A'} s_{B'}} =$$

$$\frac{(5.1973-2.8643)(8.7169-2.3675)}{2.9485*5.5310} + \frac{(3.8453-2.864)(-1.4040-2.3675)}{2.9485*5.5310}$$

$$+ \frac{(-0.4497-2.8643)(-0.2103-2.3675)}{2.9485*5.5310}$$

.

$$= \frac{2.3330*6.3494 + 0.9810*(-3.7715) + (-3.3140)*-2.5779}{2.9485*5.5310}$$

$$= \frac{14.8128 + -3.6998 + 8.5429}{16.3082}$$

$$= \frac{19.6559}{16.3082} = 1.2053$$

These two correlations are not the same; hence the correlation does not produce the same value on an unprojected vector and on a projected vector.

Mahalanobis distance: Typically the Mahalanobis distance is calculated on vectors projected into eigenspace, since it uses the eigenvalues to weight the contribution along each axis. To illustrate Mahalanobis distance as measured in eigenspace, I will calculate the Mahalanobis distance between the projected vectors A' and B'.

$$C_1 = \frac{1}{\sqrt{l_1}} = \frac{1}{\sqrt{153.2345}} = \frac{1}{12.3788} = 0.0808$$

$$C_2 = \frac{1}{\sqrt{l_2}} = \frac{1}{\sqrt{25.6096}} = \frac{1}{5.0606} = 0.1976$$

$$C_3 = \frac{1}{\sqrt{l_3}} = \frac{1}{\sqrt{1.1559}} = \frac{1}{1.0751} = 0.9301$$

$$Mah(A', B') = -\sum_{i=1}^{N} A_i' B_i' C_i$$

$$= -(5.1973 * 8.7169 * 0.0808) + (3.8453 * -1.4040 * 0.1976) + (-0.4497 * -0.2103 * 0.9301)$$

$$= -3.6606 + (-1.0668) + 0.0880 = -2.6818$$

5. Experiments

I performed four experiments, each on one of two datasets. The first experiment combines similarity measures in an attempt to improve performance. The second experiment tests whether performance improves when eigenvectors are ordered by like-image difference in eigenspace projection. The third experiment compares many variations of eigenspace projection and Fisher discriminants on the Cat & Dog dataset. The fourth experiment compares several variations of eigenspace projection and Fisher discriminants on the FERET dataset.

5.1 Datasets

The Cat & Dog dataset and the FERET dataset are used for the experiments. The FERET dataset is restructured for some of the experiments.

5.1.1 The Cat & Dog Dataset

The Cat & Dog dataset was created by Mark Stevens and is available through the Computer Vision group at Colorado State University. The original dataset of 100, 64x64 pixel, grayscale images consisted of 50 cat images and 50 golden retriever images. I collected 50 additional cat images and 50 additional images of different types of dogs. Each image shows the animal's head and is taken directly facing the animal's face. Figure 10 shows example images from the Cat & Dog dataset. This is a 2-class classification problem, since each of the test images is either a cat or a dog.

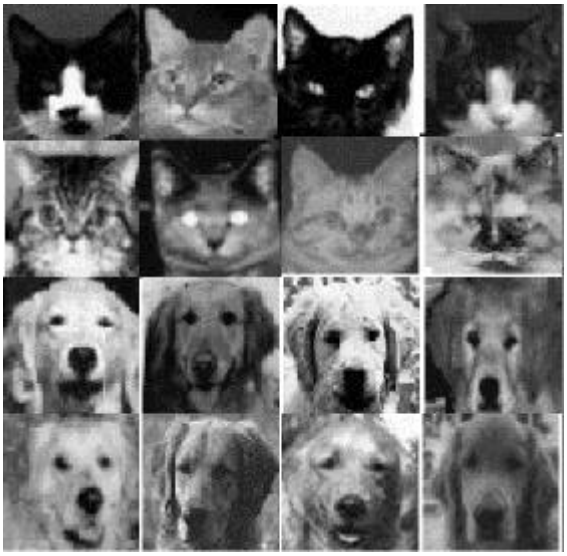

Figure 10. Sample images from the Cat & Dog dataset

Figure 11. Sample images from the FERET dataset

5.1.2 The FERET Dataset

Jonathan Phillips at the National Institute of Standards and Technology made the FERET dataset available to our department [10,12,13]. The FERET database contains images of 1196 individuals, with up to 5 different images captured for each individual. The images are separated into two sets: gallery images and probes images. Gallery images are images with known labels, while probe images are matched to gallery images for identification. The database is broken into four categories:

FB: Two images were taken of an individual, one after the other. One image is of the individual with a neutral facial expression, while the other is of the individual with a different expression. One of the images is placed into the gallery file while the other is used as a probe. In this category, the gallery contains 1196 images, and the probe set has 1195 images.

Duplicate I: The only restriction of this category is that the gallery and probe images are different. The images could have been taken on the same day or 1 ½ years apart. In this category, the gallery consists of the same 1196 images as the FB gallery while the probe set contains 722 images.

fc: Images in the probe set are taken with a different camera and under different lighting than the images in the gallery set. The gallery contains the same 1196 images as the FB & Duplicate I galleries, while the probe set contains 194 images.

Duplicate II: Images in the probe set were taken at least 1 year after the images in the gallery. The gallery contains 864 images, while the probe set has 234 images.

Figure 11 shows example images from the FERET dataset.

5.1.3 The Restructured FERET Dataset

I restructured a portion of the FERET dataset so that there are four images for each of 160 individuals. Two of the pictures are taken on the same day, where one picture is of the individual with a neutral facial expression and the other is with a different expression. The other two pictures are taken on a different day with the same characteristics. The purpose of this restructuring is to create a dataset with more than one training image of each individual to allow testing of Fisher discriminants.

5.2 Bagging and Combining Similarity Measures

Different similarity measures have been discussed, but up until this point they have been examined separately. I will now examine combining some of the similarity measures

together in the hopes of improving performance. The following four similarity measures are examined: L_1 norm (29), the L_2 norm (30), covariance (31), and the Mahalanobis distance (32). I test both simple combinations of the distance measures and "bagging" the results of two or more measures using a voting scheme [2,3,9].

1 5.2.1 Adding Distance Measures

A simple way to combine distance measures is to add them. In other words, the distance between two images is defined as the sum *S* of the distances according to two or more traditional measures:

$$S(a_1,...,a_h) = a_1 + ... + a_h \qquad (41)$$

Using *S*, all combinations of base metrics (L_1, L_2 , covariance, Mahalonobis) are used to select the nearest gallery image to each probe image. The percentage of images correctly recognized using each combination on the Duplicate I probe set, is shown in Table 5, along with the recognition rates for the base measures themselves. Of the four base measures, there appears to be a significant improvement with the Mahalanobis distance. On the surface, 42% seems much better than 33%, 34% or 35%. The best performance of any combined measure is 43% for the *S*(L_1, Mahalanobis) and *S*(L_1,covariance, Mahalanobis) combinations. While higher, the difference does not appear significant.

I used McNemar's test, which simplifies to the sign test [22,24], to calculate the significance of differences in these results. The McNemar's test calculates how often one algorithm succeeds while the other algorithm fails. I formulated the following hypotheses to test significant difference in the previous results.

Table 3. Results of McNemar's test among base measures.

Algorithms 1	Algorithm 2	Success/Success	Success/Failure	Failure/Success	P<
L1	L2	219	34	20	0.038
L1	Angle	214	39	32	0.238
L1	Mahalanobis	220	33	85	0.00001
L2	Angle	224	15	22	0.162
L2	Mahalanobis	214	25	91	0.00001
Angle	Mahalanobis	225	21	80	0.00001

Table 4. Results of McNemar's test of the *S*(L_1, Mahalanobis) and *S*(L_1,covariance, Mahalanobis) combinations compared to the Mahalanobis distance

Algorithms 1	Algorithm 2	Success/Success	Success/Failure	Failure/Success	P<
Mahalanobis	S(L_1+Mahalanobis)	282	23	26	0.388
Mahalanobis	*S* (L1, Ang, Mah)	280	25	29	0.342

1. Of the four base measures, Mahalanobis distance outperforms all others, 42% versus 33%, 34% or 35%.
2. The performance of any combined measures is not statistically better than the performance of the base measures. 43% for the *S*(L_1, Mahalanobis) and *S*(L_1,covariance, Mahalanobis) combinations versus 42% for Mahalanobis distance.

Table 3 shows the results of McNemar's test performed for each pair of base measures. Note that Mahalanobis distance always fails less often than the other similarity measures, indicated by $P < 0.00001$. Yet, no other measure is found to be different.

Table 4 shows the results of McNemar's test of the *S*(L_1, Mahalanobis) and *S*(L_1,covariance, Mahalanobis) combinations compared to the Mahalanobis distance. Here no significant difference is found between these algorithms, indicating that when they do differ on a particular image each is equally likely to identify the image correctly.

Table 5. Results of adding similarity measures

Classifier	Duplicate I
L1	0.35
L2	0.33
Covariance	0.34
Mahalanobis	0.42
$S(L_1, L_2)$	0.35
$S(L_1$, Covariance)	0.39
$S(L_1$, Mahalanobis)	0.43
$S(L_2$, Covariance)	0.33
$S(L_2$, Mahalanobis)	0.42
S(Angle, Covariance)	0.42
$S(L_1, L_2$, Covariance)	0.35
S(L1, L2, Mahalanobis)	0.42
S(L1, Cov, Mah)	0.43
S(L2, Cov, Mah)	0.42
S(L1, L2,Cov, Mah)	0.42

Table 6. Results of bagging similarity measures

Classifier	Dup I	FB
L1	0.35	0.77
L2	0.33	0.72
Cov	0.34	0.70
Mah	0.42	0.74
Bagging	0.37	0.75
Bagging (best 5)	0.38	0.78
Bagging (Weighted)	0.38	0.77

Interestingly, the performance of the combined measures is never less than the performance of their components evaluated separately. For example, the performance of $S(L_1, L_2)$ is 35%; this is better than the performance of L_2 (33%) and the same as L_1 (35%). These results suggest that L_1 and L_2 are identifying the same images correctly; hence combining measures does not identify any additional images correctly.

5.2.2 Distance Measure Aggregation

The experiment above tested only a simple summation of distance measures; one can imagine many weighting schemes for combining distance measures that might outperform simple summation. Rather than search the space of possible distance measure combinations, however, I took a cue from recent work in machine learning that suggests the best way to combine multiple estimators is to apply each estimator independently and combine the results by voting [2,3,9].

For face recognition, this implies that each distance measure is allowed to vote for the image that it believes is the closest match for a probe. The image with the most votes is chosen as the matching gallery image. Voting is performed three different ways.

Bagging: Each classifier is given one vote as explained above.

Bagging, best of 5: Each classifier votes for the five gallery images that most closely match the probe image.

Bagging, weighted: Classifiers cast five votes for the closest gallery image, four votes for the second closest gallery image, and so on, casting just one vote for the fifth closest image.

Table 6 shows the performance of voting for the Duplicate I and FB probe sets. On the Duplicate I data, Mahalanobis distance alone does better than any of the bagged classifiers: 42% versus 37% and 38%. On the simpler FB probe set, the best performance for a separate classifier is 77% (for L_1), and the best performance for the bagged classifiers is 78%. The McNemar's test confirms that this is not a significant improvement. In the next section, I explore one possible explanation for this lack of improvement when using bagging.

2 5.2.3 Correlating Distance Metrics

As described in [2], the failure of voting to improve performance suggests that the four distance measures share the same bias. To test this theory, I correlate the distances calculated by the four measures over the Duplicate I probe set. Since each measure is

defined over a different range, Spearman rank correlation is used [14]. For each probe image, the gallery images are ranked by increasing distance to the probe. This is done for each pair of distance measures. The result is two rank vectors, one for each distance measure. Spearman's Rank Correlation is the correlation coefficient for these two vectors.

Table 7 shows the average correlation scores. L_2, covariance and Mahalanobis all correlate very closely to each other, although L_1 correlates less well to covariance and Mahalanobis. This suggests that there might be some advantage to combining L_1 with covariance or Mahalanobis, but that no combination of L_2, covariance or Mahalanobis is very promising. This is consistent with the scores in Table 5, which show that the combinations $S(L_1$, covariance) and $S(L_1$, Mahalanobis) outperform these classifiers individually.

I also constructed a list of images in the FB probe set that were grossly misclassified, in the sense that the matching gallery image is not one of the ten closest images according to one or more distance measures. A total of 179 images are poorly identified by at least one distance measure.

Table 8 shows the number of images that are poorly identified by all four distance measures, three distance measures, two distance measures, and just one distance measure. This table shows that there is shared bias among the classifiers, in that they seem to make gross mistakes on the same images. On the other hand, the errors do not overlap

Table 7. Correlation between similarity measures.

	L1	L2	cov	Mah
L1	1	0.46057	0.38873	0.38749
L2	0.46057	1	0.615479	0.498865
cov	0.38873	0.615479	1	0.582534
Mah	0.38749	0.498865	0.582534	1

Table 8. Number of images similarly identified poorly

Images commonly poorly identifed	# of images out of 179	% images
by 4 classifiers	46	25.7
by 3 classifiers	48	26.82
by 2 classifiers	34	18.99
by 1 classifier	51	28.29

completely, suggesting that some improvement might still be achieved by some combination of these distance measures.

5.3 Like-Image Difference on the FERET dataset

In order to test the performance of like-image ordering of eigenvectors compared to ordering by corresponding eigenvalue, I performed two experiments. The first experiment is on the original FERET dataset and compares results to those of Moon & Phillips. The second experiment is on the restructured dataset, where 10 trials are run and training/test data is clearly separated.

For each of the 1195 probe/gallery matches in the FB probe set of the original FERET dataset, I calculate the difference between the probe and gallery image in eigenspace. These differences are summed together and then divided by the eigenvalue to calculate the like-image difference. The smaller this number is, the better the eigenvector should be at matching images. The top N eigenvalues are selected according to the like-image difference measure, and the FB probe set is reevaluated using the L_1 norm. Figure 12 shows the performance scores of the reordered eigenvalues compared to the performance of the eigenvalues ordered by eigenvalue, as performed by Moon & Phillips. Table 9 shows the number of images correctly identified by each ordering method and the results

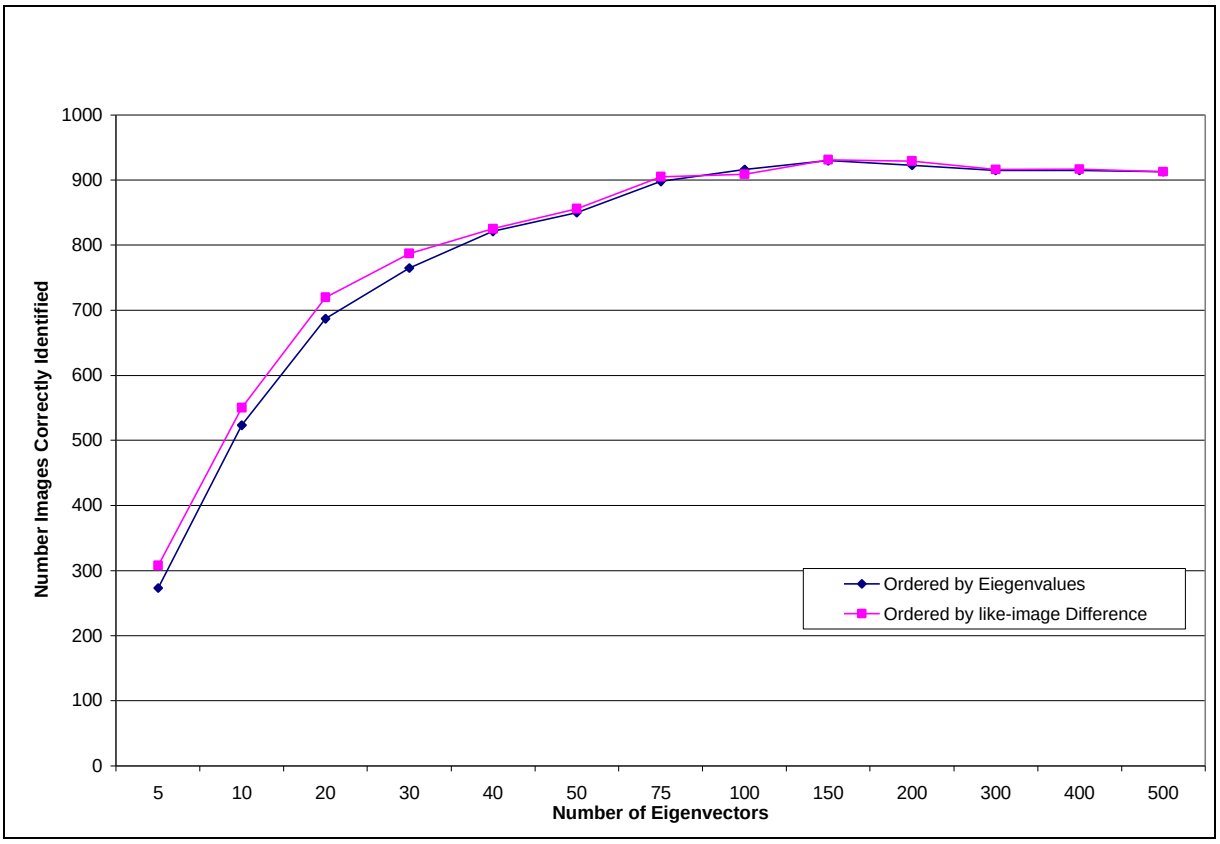

Figure 12. Graph of the performance when ordering by eigenvalues vs. when ordered by like-image difference

Table 9. Correctly identified images by ordering of eigenvectors by eigenvalue and by like-image difference.

# Evs	L1 Sorted by Evs	L1 Sorted by Like-Image Diff.	S/S	S/F	F/S	P <
5	273	308	203	70	105	0.0050
10	523	550	472	51	78	0.0109
20	687	720	659	28	61	0.0003
30	765	787	748	17	39	0.0023
40	821	825	796	25	29	0.3417
50	850	856	839	11	17	0.1725
75	898	905	886	12	19	0.1405
100	916	909	903	13	6	0.0835
150	930	931	925	5	6	0.5000
200	923	929	921	2	8	0.0547
300	915	916	913	2	3	0.5000
400	915	917	912	3	5	0.3633
500	913	913	913	0	0	1.0000

of McNemar's test on the results. Note that for less than 40 eigenvectors McNemar's test indicates that like-image difference ordering of eigenvectors fails less than eigenvectors ordered by corresponding eigenvalue.

Reordering by the like-image difference improves performance for small numbers of eigenvectors (up to the first 40). This suggests that the like-image distance measure should be used when selecting only a few eigenvectors. When more than 40 eigenvectors are selected, there is enough overlap between the two orderings that no significant difference in performance occurs; either method of ordering eigenvectors can be used.

To further test the performance of like-image difference, I ran an experiment on the restructured FERET dataset. For this experiment, 10 trials are run with each of four algorithms (eigenspace projection using L_1 norm, L_2 norm, covariance and Mahalanobis distance), but the training and test images are selected randomly. For each individual, two images taken on the same day are randomly selected for training data. One image of the remaining two images, of the final 140 individuals, is randomly selected as test data. Only 140 images are used as test data because the like-image difference algorithm requires 20 additional training images. Therefore, in each trial 320 training images and 140 test images are used. I tested eigenspace projection by removing the last 40% of the eigenvectors and by reordering the eigenvectors by like-image difference and then removing the last 40%. Table 10 shows the results of these tests. The results of these tests do not show an improvement through the reordering of eigenvectors by like-image difference. Thus, like-image difference ordering of eigenvectors does not improve performance.

Table 10. Results of ordering eigenvectors ordering eigenvectors by eigenvalue vs. by like-image difference.

	Remove Last 40% (127) of the Eigenvectors				Reorder by Significant Difference and remove last 40% (127) of the Eigenvectors			
Trial #	**L1**	**L2**	**Angle**	**Mahalanobis**	**L1**	**L2**	**Angle**	**Mahalanobis**
1	69	59	67	89	68	63	65	85
2	70	61	62	82	68	60	62	83
3	76	54	71	89	76	59	70	86
4	73	53	67	83	71	61	66	84
5	71	62	58	83	70	54	59	84
6	72	50	61	89	67	64	61	79
7	75	55	66	91	72	63	66	85
8	67	61	61	91	69	53	61	83
9	71	59	60	86	72	56	59	79
10	73	63	66	92	73	61	66	86
Average	71.7	57.7	63.9	87.5	70.6	59.4	63.5	83.4

5.4 Cat & Dog Experiments

Two hypotheses prompted experiments on the Cat & Dog dataset. The first hypothesis is that Fisher discriminants will perform better than eigenspace projection. I believed that since the Cat & Dog dataset is a two-class problem that classifies an image as either a cat or a dog, Fisher discriminants will clearly separate the two classes. The second hypothesis is that the combination of eigenspace projection and Fisher discriminants will outperform all other algorithms. By testing multiple variations of the eigenspace projection and combination eigenspace projection/Fisher discriminants algorithms, I introduce a third hypothesis that for each of these algorithms one of the variations will perform better than the others.

The tests on the Cat & Dog data are set up according to an approach described by Salzberg [15]. The 200 cat and dog images are separated into 10 datasets. Each dataset contains 10 cat images and 10 dog images. A single trial is trained on nine of the datasets and tested with the remaining dataset. Therefore, ten trials are run on each algorithm,

using each dataset as the test dataset once. Table 11 shows the number of images correctly identified (out of 20) for all of the algorithms run on the Cat & Dog dataset.

An ANOVA test, performed on the results of the six variations of the eigenspace projection algorithm, indicates with $P < 0.97$, that there is no significant difference between the variations. An ANOVA test is also performed on the results of the six variations of the combination eigenspace projection/Fisher discriminants algorithm. Again, with $P < 0.99$, no significant difference is found between the variations.

To test the two original hypotheses, first a significant difference must be established between the four algorithms. Since a best variation of the eigenspace projection or combination eigenspace projection/Fisher discriminants algorithms could not be determined, I choose the variations with the largest mean number of images correctly identified. An ANOVA test is performed, and with $P < 0.25$ no significant difference is found.

At this time no further analysis can be performed, since no statistically significant difference can be established between any of the algorithms.

Since no statistically significant difference can be found among any of the algorithms on the Cat & Dog data, all three of the hypotheses are rejected. No specific variation of the eigenspace projection is found to perform better than any of the other variations and no variation of the combination eigenspace projection/Fisher discriminants is found to

outperform any of the other variations. Fisher discriminants does not perform better than eigenspace projection, since no difference is found between the two algorithms. The combination of eigenspace projection and Fisher discriminants does not outperform Fisher discriminants. Again, no difference is found between the two algorithms.

The fact that no algorithm outperforms the rest indicates that the cat and dog images cannot be clearly separated. Furthermore, it is interesting to note that on average, many more dogs are incorrectly identified than cats. This could simply be a result of the fact that dogs come in many shapes and sizes, but all cats look generally the same just different colors. This phenomenon can also be a result of biases in the data. Originally, the dataset contained only images of golden retrievers; I added 50 additional images of different types of dogs. Therefore, the dataset may be biased to golden retriever and dogs that look like golden retrievers and more likely to misidentify dogs that do not look like golden retrievers.

5.5 FERET Experiments

Similar to the Cat & Dog experiments, three hypotheses spurred the FERET experiments.

1. Eigenspace projection will outperform Fisher discriminants. Since the FERET dataset contains 160 classes and only two images per class, I believe Fisher discriminants will have difficulty discriminating the data.
2. The combination of eigenspace projection and Fisher discriminants will outperform all other algorithms.

3. One of the variations of eigenspace projection and combination eigenspace projection/Fisher discriminants will perform better than the other variations.

The setup of the FERET experiments differs from the setup of the Cat & Dog experiments. Here 10 trials are run on each algorithm, but the training and test images are selected randomly. For each individual, two images taken on the same day are randomly selected for training data. One image of the remaining two images, of the final 140 individuals, is randomly selected as test data. Only 140 images are used as test data because the like-image difference algorithm requires 20 additional training images. Therefore, for each trial 320 training images and 140 test images are used. Table 12 shows the number of images correctly identified for all of the algorithms run on the FERET dataset.

An ANOVA test is performed on the six variations of eigenspace projection and a significant difference, with a $P < 0.00001$, is found between the variations. To choose a best performer of these variations, I sort the variations by mean number of images correctly identified. A one-tail two-sample t-test assuming unequal variances indicates that eigenspace projection removing the first eigenvector outperforms all other variations of eigenspace projection, with $P < 0.032$.

An ANOVA test is also performed on the six variations of the combination eigenspace projection/Fisher discriminants. The six variations are found to be significantly different with $P < 0.00001$. The variations are sorted by mean number of images correctly

Table 11. Results of Cat & Dog experiments. Number of images correctly identified out of 20.

Trial	Correlation	Eigen Proj (all e-vects)	Eigen Proj (remove last 40% e-vects)	Eigen Proj (Energy Dimension)	Eigen Proj (Stretching Dimension)	Eigen Proj (remove 1st e-vect)	Eigen Proj (Like-Image Difference)	Fisher Disc.	Eigen Proj (all e-vects) & Fisher Disc.	Eigen Proj (remove last 40% e-vects) & Fisher Disc.	Eigen Proj (Energy Dimension) & Fisher Disc.	Eigen Proj (Stretching Dimension) & Fisher Disc.	Eigen Proj (remove 1st e-vect) & Fisher Disc.	Eigen Proj (Like-Image Difference) & Fisher Disc.
1	17	18	18	17	17	17	18	16	16	18	17	16	16	16
2	18	16	17	17	17	15	16	17	17	18	16	17	17	17
3	17	15	17	17	17	18	17	15	15	15	15	15	15	15
4	18	18	18	18	18	17	18	17	17	17	17	17	17	17
5	16	15	14	13	13	17	13	17	17	16	16	17	17	17
6	20	20	20	20	20	20	20	17	17	19	19	19	16	19
7	16	17	17	17	17	17	17	16	16	15	15	14	16	16
8	18	15	15	15	15	16	15	15	15	16	17	17	16	16
9	20	17	18	18	18	19	17	19	19	18	18	18	19	18
10	17	17	17	17	17	18	17	17	17	16	16	16	17	18
Avg	17.7	16.8	17.1	16.9	16.9	17.4	16.8	16.6	16.6	16.8	16.6	16.6	16.6	16.9

Table 12. Results from FERET data. Number of images correctly identified out of 140.

Trial	Correlation	Eigen Proj (all e-vects)	Eigen Proj (remove last 40% e-vects)	Eigen Proj (Energy Dimension)	Eigen Proj (Stretching Dimension)	Eigen Proj (remove 1st e-vect)	Eigen Proj (Like-Image Difference)	Fisher Disc.	Eigen Proj (all e-vects) & Fisher Disc.	Eigen Proj (remove last 40% e-vects) & Fisher Disc.	Eigen Proj (Energy Dimension) & Fisher Disc.	Eigen Proj (Stretching Dimension) & Fisher Disc.	Eigen Proj (remove 1st e-vect) & Fisher Disc.	Eigen Proj (Like-Image Difference) & Fisher Disc.
1	71	71	67	62	66	71	59	77	76	65	66	48	76	81
2	71	63	62	58	64	69	65	73	71	62	65	60	72	80
3	63	72	70	57	57	76	66	80	79	69	66	53	78	84
4	72	67	67	60	57	68	62	75	74	68	70	62	74	85
5	67	62	59	55	63	67	70	73	74	68	70	59	75	86
6	62	61	60	53	58	63	67	66	66	63	62	61	67	75
7	61	68	66	61	56	71	59	74	76	73	66	64	77	80
8	68	62	61	52	66	67	60	77	77	57	59	55	75	76
9	62	62	59	54	66	67	65	84	83	67	65	59	83	88
10	63	67	65	59	61	69	60	79	77	68	65	57	79	81
Avg	66.00	65.50	63.60	57.10	61.40	68.80	63.30	75.80	75.30	66.00	65.40	57.80	75.60	81.60

identified, and a one-tail two-sample t-test assuming unequal variances indicates that eigenspace projection, resorting eigenvectors by like-image difference and removing the last 40% of the eigenvectors combined with Fisher discriminants outperforms all the other variations, with $P < 0.003$.

Utilizing the two winning variations from above, the four algorithms are compared. First, a single actor ANOVA test is performed to establish significant difference between the algorithms. $P < 0.00001$ indicates a difference among the algorithms. To test the first hypothesis a two-tail two-sample t-test assuming unequal variances is performed on eigenspace projection with the first eigenvector removed and Fisher discriminants. $P < 0.002$ indicates that Fisher discriminants outperformed eigenspace projection. To test my second hypothesis, a one-tail two-sample t-test assuming unequal variances is performed on eigenspace projection with the first eigenvector removed and eigenspace projection resorting eigenvectors by like-image difference and removing the last 40% of the eigenvectors combined with Fisher discriminants. $P < 0.005$ indicates that the combination of eigenspace projection and Fisher discriminants outperforms all other algorithms.

Statistically significant differences are found among the algorithms on the FERET data. Specifically, eigenspace projection removing the first eigenvector outperforms all other eigenspace projection algorithms and eigenspace projection resorting eigenvectors by like-image difference and removing the last 40% of the eigenvectors combined with Fisher discriminants outperforms all other eigenspace projection/Fisher discriminants combination algorithms. Therefore, the third hypothesis is confirmed. My first

hypothesis is rejected because eigenspace projection does not outperform Fisher discriminants, instead the exact opposite is true and Fisher discriminants outperforms eigenspace projection. My second hypothesis is accepted since a variation of the combination of eigenspace projection and Fisher discriminants outperforms all other algorithms.

The fact that Fisher discriminants performs well on the FERET dataset indicates that as few as two images in a class is enough to separate classes. Furthermore, a larger number of classes does not appear to reduce Fisher discriminants ability to separate classes.

6. Conclusion

In reviewing the results of the experiments, three conclusions can be drawn. Combining distance measures in any way does not improve performance on the original FERET dataset. Although like-image difference sorting of eigenvectors seemed like a good idea, it does not improve performance on the FERET dataset. Finally, the most interesting conclusion is that the algorithms and variations of algorithms perform very differently on the Cat & Dog dataset and on the FERET dataset. While no significant difference is found among any of the algorithms on the Cat & Dog dataset, differences are found among the algorithms in the FERET dataset and best performing algorithms can be selected.

One possible reason for the lack of significant difference among algorithms in the Cat & Dog dataset is the size of the test set. For each trial 20 of 200 images are used for testing while 180 images are used for training. Since the size of the test set is small the percentage of correctly identified images drops more quickly than if a larger number of test images are used as in the FERET dataset. Given a chance to run further experiments on the Cat & Dog dataset I would modify the experiment setup to use more test images and fewer training images.

Although no difference is found among the algorithms for the Cat & Dog dataset, in the FERET dataset a best performer is clear. The algorithm that first projects images into

eigenspace and then projects them onto their Fisher basis vectors outperforms all other algorithms. It is interesting to note that these tests are performed over ten trials, where in each trial test and training images are randomly selected.

6.1 Experiment Summary

The experiments performed in this work stand out from similar experiments. They are different because they followed a specific methodology. Before experiments are performed hypotheses are formed. Experiments are performed following know approaches. After experiments are performed statistical tests such as the McNemar's test, ANOVA test, and t-test are used to confirm or reject the hypotheses. Below is a summary of the hypothesis, tests, and conclusions from the experiments.

Hypothesis #1: Adding raw similarity measure scores improves performance.

Tests: In section 5.2.1 raw similarity measure scores are combined and their performance is compared to the performance of the raw similarity measures. McNemar's test is used to identify significant differences among the algorithms.

Conclusion: Adding raw similarity measure scores does not improve performance.

Hypothesis #2: Bagging similarity measures improves performance.

Tests: In section 5.2.2 similarity measures are bagged and their performance is compared to the performance of the raw similarity measures. McNemar's test is used to identify significant differences among the algorithms.

Conclusion: Bagging similarity measures does not improve performance.

Hypothesis #3: The four distance measures are identifying the same images correctly and the same images incorrectly.

Tests: Two tests are performed in section 5.2.3. The similarity measures are correlated to each other and images that are grossly misclassified by each measure are compared.

Conclusion: Although there is some overlap among similarity measures, they do not overlap completely.

Hypothesis #4: Resorting the original FERET dataset by like-image difference improves performance.

Tests: In section 5.3, the eigenvectors used by Moon & Phillips are resorted by like-image difference and compared to the Moon & Phillips results by the McNemar's test.

Conclusion: Like-image difference improves performance when a small number of eigenvectors are used.

Hypothesis #5: Resorting the restructured FERET dataset by like-image difference and testing on different images improves performance.

Tests: In section 5.3, algorithms are run on the restructured dataset sorting by eigenvalue and by like-image difference. Ten trials are run each randomly selecting test and training images.

Conclusion: Like-image difference does not improve performance when test and training data is separated.

Hypothesis #6: One of the variations of eigenvector selection outperforms the others on the Cat & Dog dataset.

Tests: In section 5.4, two ANOVA tests are performed, one on the six variations of eigenspace projection and one on the six variations of Fisher discriminants where images are first project into eigenspace.

Conclusion: No variation of eigenvector selection outperforms the others on the Cat & Dog dataset.

Hypothesis #7: Fisher discriminants will perform better than eigenspace projection on the Cat & Dog dataset.

Tests: In section 5.4, an ANOVA test is performed on the four algorithms. No significant difference is identified among the algorithms.

Conclusion: Fisher discriminants does not perform better than eigenspace projection on the Cat & Dog dataset.

Hypothesis #8: Fisher discriminants where images are first projected into eigenspace outperforms all other algorithms on the Cat & Dog dataset.

Tests: In section 5.4, an ANOVA test is performed on the four algorithms. No significant difference is identified among the algorithms.

Conclusion: Fisher discriminants where images are first projected into eigenspace does not outperform all other algorithms on the Cat & Dog dataset.

Hypothesis #9: One of the variations of eigenvector selection outperforms the others on the FERET dataset.

Tests: In section 5.5, two ANOVA tests are performed, one on the six variations of eigenspace projection and one on the six variations on Fisher discriminants where images are first project into eigenspace.

Conclusion: In the case of eigenspace projection, removing the first eigenvector outperforms all other variations on the FERET dataset. In the case of combining eigenspace projection and Fisher discriminants, resorting eigenvectors by like-image difference and removing the last 40% of the eigenvectors outperforms all the other variations on the FERET dataset.

Hypothesis #10: Eigenspace projection outperforms Fisher discriminants on the FERET dataset.

Tests: In section 5.5, an ANOVA test is performed on the four algorithms. Significant difference is identified among the algorithms. A two-tailed t-test is performed on eigenspace projection and Fisher discriminants.

Conclusion: Eigenspace projection does not outperform Fisher discriminants on the FERET dataset. In fact the exact opposite is true, Fisher discriminants outperforms eigenspace projection.

Hypothesis #11: Fisher discriminants where images are first projected into eigenspace outperforms all other algorithms on the FERET dataset.

Tests: In section 5.5, an ANOVA test is performed on the four algorithms. Significant difference is identified among the algorithms. A t-test is performed on Fisher discriminants and the other algorithms.

Conclusion: Fisher discriminants where images are first projected into eigenspace outperforms all other algorithms on the FERET dataset.

3 6.2 Future Work

I would like to enlarge the Cat & Dog dataset to contain more than two classes. There are 50 images of horses available with the cat and dog images. I would like to add those images along with another set of images of a different type of animal. The tests performed on the Cat & Dog dataset could then be run on a dataset with three classes and a dataset of four classes and performance could be compared.

I would also like to rerun the tests on the FERET dataset, where only one image of each individual would be in the training data. I would expect performance to drop greatly on this dataset, especially with Fisher discriminants.

Although like-image difference selection of eigenvectors did not improve performance on the FERET dataset, I still believe that some combination of eigenvector selection will improve performance. Therefore, I would like to continue to explore possible algorithms for eigenvector selection that will improve classification performance within subspaces.

Throughout my experiments I try to use statistical tests to support my results. In the future I would like to rerun many of the experiments performed by Moon and Phillips [10] and run statistical tests such as the McNemar's test to examine significant difference among the results.

4 Appendix I, Symbol Glossary

x^i = Raw training image
X = Data matrix of raw training images
Ω = Covariance matrix
$\bar{x}^i$ = Mean centered training image
$\bar{X}$ = Data matrix of mean centered training images
m = Mean image
P = Number of training images
N = Number of pixel sin the training images
V = Eigenvectors
Λ = Eigenvalues
v_i = i^{th} eigenvector
l_i = i^{th} eigenvalue
$\tilde{x}^i$ = Projected centered training image
y^i = Raw test image
$\bar{y}^i$ = Mean centered test image
$\tilde{y}^i$ = Projected centered test image
Ω' = Modified covariance matrix
V' = Eigenvectors of Ω'
Λ' = Eigenvalues of Ω'
$\hat{V} = \bar{X}V'$
C = Number of classes
S_i = Within class scatter matrix for i^{th} class
S_W = Within class scatter matrix
S_B = Between class scatter matrix
n_i = Number of images in $\boldsymbol{i^{th}}$ class
$\hat{x}^i$ = Class mean centered image
U = Any orthonormal basis
$\hat{m}_i$ = Centered class mean
$\tilde{m}_i$ = Projected centered class mean
e_i = Energy dimension
s_i = Stretching dimension
w = Like-image difference

References

[1] Belhumeur P., Hespanha J., and Kriegman D. (1997), Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, *IEEE Trans. PAMI*, 19(7):711-720.

[2] Breiman L. (1994), Bagging Predictors. *Technical Report No. 421, Dept. of Statistics, University of California (Berkeley).*

[3] Dietterich T. and Bakiri G. (1995), Solving Multiclass Learning Problems via Error-Correction Output Code, *Journal of Artificial Intelligence Research,* 2:263-286.

[4] Duda R. and Hart P., *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[5] Fisher, R.A. (1936), The use of Multiple Measures in Taxonomic Problems, *Ann. Eugenics,* 7:179-188.

[6] Horn R. and Johnson C., *Matrix Analysis,* New York: Cambridge University Press, 1985.

[7] Kirby M. (2000), *Dimensionally of Reduction and Pattern Analysis: an empirical approach.* Under contract with Wiley.

[8] Kirby M. and Sirovich L. (1990), Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces, *IEEE Trans. PAMI*, 12(1):103-108.

[9] Kong E.B. and Dietterich T. (1995), Why error-correcting output coding works with decision trees, *Technical Report, Dept. of Computer Science, Oregon State University (Corvalis).*

[10] Moon H. and Phillips J., Analysis of PCA-based Faced Recognition Algorithms. In Boyer K. and Phillips J., editors, *Empirical Evaluation Techniques In Computer Vision*, IEEE Computer Society Press, Los Alamitos, CA, 1998.

[11] Nayar S., Nene S., and Murasr H. (1996), Real-Time 100 Object Recognition System. *Proceedings of* ARPA *Image Understanding Workshop.*

[12] Phillips J., Moon H., Rizvi S., and Rauss P., The FERET Evaluation. In Wechslet H., Phillips P., Bruse, V., Fogeliman Soulie F., and Hauhg T., editors, *Face Recognition: From Theory to Application*, Springer-Verlag, Berlin, 1998.

[13] Phillips J., Moon H., Rizvi S., and Rauss P. (1999), The FERET Evaluation Methodology for Face-Recognition Algorithms. *NIST Technical Report NISTIR 6264*.

[14] Press W., Teukolsky S., Vetterling W., and Flannery B., *Numerical Recipes in C, The Art of Scientific Computing*, New York: Cambridge University Press, Inc., 1988.

[15] Salzberg S., (1997) On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data mining and Knowledge Discovery*, 1, 317-327.

[16] Sirovich L. and Kirby M. (1987), A low-dimensional procedure for the characterization of human faces, *The Journal of the Optical Society of America*, 4:519-524.

[17] Stevens M. (1999), Reasoning about Object Appearance in terms of a Scene Context. *Ph.D. thesis*.

[18] Swets D. and Weng J. (1996), Using Discriminant Eigenfeatures for Image Retrieval. *IEEE Trans. PAMI*, 18(8):831-836.

[19] Swets D. and Weng J. (1999), Hierarchical Discriminant Analysis for Image Retrieval, *IEEE Trans. PAMI*, 21(5):386-401.

[20] Trucco E. and Verri A., *Introductory Techniques for 3-D Computer Vision*, New Jersey: Prentice-Hall, Inc., 1998.

[21] Turk M. A. and Pentland P., (1991) Face Recognition Using Eigenfaces, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 586-591.

[22] Yambor W., Draper B., and Beveridge J.R., (2000) Analysis of PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures. *Second Workshop on Empirical Evaluation Methods in Computer Vision*.

[23] Zhao W., Krishnaswamy A., Chellappa R., Swets D., and Weng J., (1998) Discriminant Analysis of Principle Components for face Recognition. *3rd International Conference on Automatic Face and Gesture Recognition*, 336-341.

[24] IFA. Statistical tests, http://fonsg3.let.uva.nl:8001/service/statistics.html. Website, 2000.