A Face in the Crowd
a quantitative understanding

Dhasharath Shrivathsa, Kaitlyn Keil

***Abstract*—This paper will present the issue of facial recognition in computers and analyze two algorithms–Eigenfaces and Fisherfaces–that have been developed to solve it. We then show that Fisherfaces is more accurate, particularly in varying lighting.**

I. Introduction

WHILE humans are able to recognize faces at a glance, despite variations in angle, distance, lighting, and expression, it remains difficult for computers, despite the development of numerous algorithms in recent years. One of the most successful can be seen in Facebook's photo tagging, where faces are quickly found and names suggested. However, even here it is not uncommon for a statue or oddly-wrinkled knee to be suggested as your best friend.

If we look at images as matrices of pixels and pixel values, we can see the base of the issue. Small changes, such as shadows, a large smile, or turning the head, will change where the values of an image are high and where they are low, with high values corresponding to shades close to white and low values to close to black.

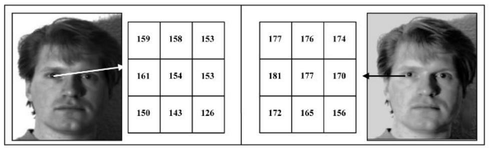

Fig. 1. A photograph of a man under different lighting conditions, with the pixel values within his eye displayed for both. Despite being easily recognizable by sight, the pixel values are significantly different, which keeps the computer from registering them as the same.

Fig. 1 shows how the change in lighting creates a large difference in the pixel values of the man's eyes. While looking at correlation (see Glossary) would still suggest they are similar, a simple value comparison would deny the similarities. If he turned his head, changing the location of his eye within the image, correlation would no longer be able to identify this man.

In response to this problem, several algorithms have developed. Using a database of images created from 344 pictures of the QEA class (eight pictures with varying expressions from 43 people, see Fig. 2 for a few examples), we will examine Eigenfaces and Fisherfaces and compare how well they identify faces, both from the same photoshoot and from different situations, where the lighting and framing are different.

Eigenfaces are a facial recognition algorithm derived through principal component analysis on a covariance matrix of a set of training faces. Once we have these most statistically significant traits, we can process a new face and compare it to those within the training set. We can then identify the face comprised of most similar elements to be the same person as the new image.

Fisherfaces are an alternate facial recognition algorithm that addresses several of the problems that exist in the Eigenfaces implementation. Instead of employing only PCA, Fisherfaces takes the vector set and reduces it further with LDA, which locates the classes (each person) as far away in vector space from all the others as possible.

Both are trained on a database of images from the Quantitative Engineering Analysis class and attempt to match an imput image to a face and a name from within the classroom dataset.

Algorithms, the next section, is broken into two subsections. It begins by deconstructing and examining the steps involved with calculating Eigenfaces and identifying a new image based upon the results. The second subsection follows a similar process for Fisherfaces, including a brief introduction to linear discriminant analysis.

The Comparison of Performance reviews the results of accuracy testing with both programs and gives explanation. It will make an argument for Fisherfaces as the more accurate algorithm, particularly when difference in lighting is involved and justify the results.

The conclusion will review the main points and provide insight into where we would continue to explore in the future, both in changes to our existing programs and in new features we would add.

The glossary, found at the end, will provide the definition and explanation of various terms used throughout the paper.

II. Algorithms

WE compared two algorithms for facial recognition. First, we used Eigenfaces with principle component analysis to match faces. These results were then compared against those of the newer algorithm Fisherfaces, which uses linear discriminant analysis to classify faces. Both used Euclidean distance to place the input face.

A. Eigenfaces

1) Intermediary data representation: For Eigenfaces to work properly, we need a large set of training images. Ours included several expressions from each person, to try to

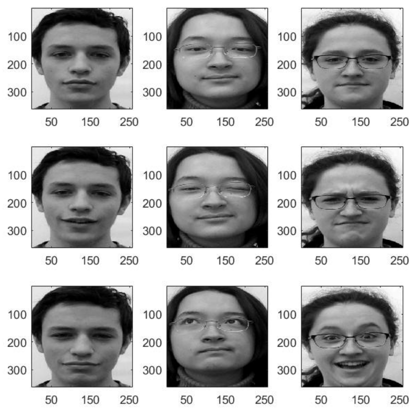

Fig. 2. Nine training faces from the QEA faces dataset. Each columns represents a person and each row, a different expression. This was our training set.

capture the variety of emotions a person might be feeling at the moment of photography.

Each of these grey-scale images is a matrix P_{hw}, where h is the height of the image in pixels, w is the width, and each pixel is represented by a value representing the shade. Because $h \times w$ is generally a very large number (in our case, with 360×256 pixel images, 92160), we downsampled the images by averaging the value of a number of pixels and reducing the image. For example, if we want the image to be one fifth of the original size, we would average the values of a 5×5 box of pixels and reduce it to a single value in a new matrix, then move to the next group.

Once the image has been resampled to a more workable size, we reshape it into a column matrix by stacking the rows on top of each other and combine them into a single $hw \times N$ matrix, where hw is the reduced height times the reduced width, or the total number of pixels in an image. This is matrix V, and each image is represented by V_i. From here, we find the mean image by adding all of the column vectors and dividing by the number of samples, N.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} V_i \tag{1}$$

This mean is used to normalize the images. Each column vector has the mean image subtracted off, centering all the images around 0 in a new matrix following the same shape, A.

$$A = \sum_{i=1}^{N} V_i - \mu \tag{2}$$

The eigenfaces of these images are the eigenvectors of the covariance matrix of A. However, even after downsampling the images by a factor of 15, this still would result in a 432×432 matrix to parse. Instead, we can use principal component analysis to find the most statistically important eigenvectors. Using PCA, we don't need to find the covariance matrix, as we can operate directly on the image matrix. The covariance matrix of A, X, is found in Eq. 3.

$$X = \frac{AA^T}{N} \tag{3}$$

However, the singular value decomposition of A gives us

$$A = U\Sigma V^T \tag{4}$$

If we then look at the eigenvalue decomposition of AA^T, such as we would need for the covariance matrix, this becomes

$$AA^T = U\Sigma\Sigma^T U^T \tag{5}$$

$$AA^T = U\Lambda U^T \tag{6}$$

where Λ is a diagonal matrix of the eigenvalues of AA^T. From this, we can see that the column vectors of U are the eigenvectors of A as well as of AA^T. Because of this, we can find the eigenfaces (U) by using SVD on A. By using PCA (see Glossary) instead, we automatically eliminate all but the significant eigenfaces.

The top 50 eigenvalues correspond with the 50 eigenfaces which capture the majority of the variation between faces, and can be used to process data much more quickly than trying to analyze every image by 340 different eigenfaces. Once we have these fifty eigenfaces, we can find the projection of each eigenvector on the images,

$$W = U'A \tag{7}$$

where each column of W stores the 'weight' of each eigenface that makes up an image, or $W_i = [w_1, w_2, ...w_{50}]$. We can now introduce a new image and find the weights of each eigenvector on the image. This follows much the same process. In the initial processing of the new image, we reduce the size and We reduce the size and adjust the brightness. This is done to help compensate for lighting differences. To do this, we find the mean value of the average face (ma), the mean value of the new face (na), and multiply the new image by $\frac{ma}{na}$. We then transform the image matrix into a column, subtract off the mean, then multiply it by the transpose of the eigenfaces. If we find the Euclidean distance between these weights and those of the original images, we can deduce that the smallest value indicates the matching face and display both. This works best for images with similar lighting and positioning, as even small variations will cause the matrices to be quite different.

B. Fisherfaces

C. Intermediary data representation

There exist two matrices that we need to find the Fisherfaces of a given dataset. The between-class matrix (different people's faces) S_B, and the within-class matrix (different faces for the same person) S_W. They are defined as:

$$\mu = \frac{1}{|x|}\sum_{i=1}^{|\mathbf{x}|} x_i \tag{8}$$

$$\mu_c = \frac{1}{N_c}\sum_{i=1}^{N_c} x_{((c-1)\times N_c)+i} \tag{9}$$

$$S_B = \sum_{i=1}^{c} N_i\left(\mu_i - \mu\right)\left(\mu_i - \mu\right)^\tau \tag{10}$$

$$S_W = \sum_{i=1}^{c}\sum_{j=1}^{N_c}\left(x_{((i-1)\times N_c)+j} - \mu_i\right)\left(x_{((i-1)\times N_c)+j} - \mu_i\right)^\tau \tag{11}$$

Where μ in Eq. 8 is the average face in the dataset and x is the face set, μ_c in Eq. 9 is the average face of class c and N_c is the number of images in class c, S_B in Eq. 10 is the between-class matrix of size $\mathbb{R}^{m\times m}$, and lastly S_W in Eq. 11 is the within-class matrix of size $\mathbb{R}^{m\times m}$.

Maximization of the determinant of the between-class matrix with respect to the within-class matrix results in the equation:

$$S_B w_i = \lambda_i S_W w_i \tag{12}$$

However, the matrix S_W is singular, meaning that we can make it exactly zero. The reason S_W is singular is due to it not having full rank, as the dimensions of S_W are $n \times n$, where n is the number of pixels in an image, and generally there are less training images than the number of pixels in an image squared, preventing full rank of the covariance matrix. At most our rank is $N - c$.

Given this, we can see that we need only the $N-c$ principal components for the matrix to become full rank so we can solve Eq. 12. If we *only* take PCA however, we see some actual *smearing of the classes together* to where they are not linearly separable (Fig. 3) We can solve this first by not reducing to less than $N - c$ principal components;

$$S_T = \sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^\tau \tag{13}$$

$$S_T w_i = \lambda_i w_i \tag{14}$$

$$W_{PCA} = w_i^{(N-c)\times|x|} \tag{15}$$

In Eq. 13, we make the correlation matrix for each image against the mean image, in Eq. 14 we get all the corellation matricies eigenvectors and eigenvalues, and in Eq. 15 we choose the $N - c$ eigenvectors with the highest eigenvalues.

Then for classification we can leverage LDA to maximize the between-class distance.

$$W_{FLD} = W_{PCA} S_B W_{PCA} w_i = \lambda_i W_{PCA} S_W W_{PCA} w_i \tag{16}$$

$$W_{OPT} = W_{FLD}^\tau W_{PCA}^\tau \tag{17}$$

In Eq. 16, we do the same as Eq. 12, except applying the most pertinent eigenvectors W_{PCA} to both. Lastly, we find the set of Fisherfaces W_{OPT} in Eq. 17

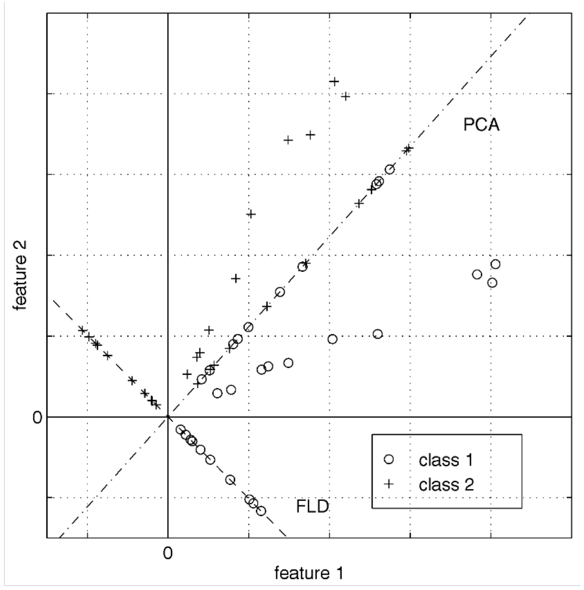

Fig. 3. Given the data points of class 1 and class 2, we can if we compress it to the main axis of variance, we get inter-class interactions. If we want to maximize the inter-class distance, we use FLD to make the two classes linearly separable, *and* and compressed into one dimension

W_{OPT} are the Fisherfaces of the training set. We can figure out which face is which by multiplying each classes mean face through the set of Fisherfaces, causing each face to have an associated weight vector W_i.

1) Facial recognition: Image classification can be performed by taking a new face and multiplying it with the set of Fisherfaces to obtain a new weight vector W_r, and taking the L^2 norm (Euclidean distance) from W_r to all W_i, and taking the lowest one as the class the face belongs to.

III. Comparison of Performance

Looking at Fig. 2, we see that the faces used in the training set, while varying expression, don't have much variation in lighting or angle. Unfortunately, this has a large impact on the effectiveness of Eigenfaces.

While within dataset accuracy (removing a picture from the training set to compare to the rest) is 98.26%, if we use an image from a different set of photographs, we get a much lower accuracy of 9.677% accuracy without brightness normalization and 22.58% with. These results were fairly typical; the incorrectly identified image usually bore a fairly strong resemblance to the new picture.

Fisherfaces, however, resulted in a much stronger performance using classification and LDA. Fig. 4 shows that, when framing and angle is kept similar, both Eigenfaces and Fisherfaces are able to identify the person in question. However, when more significant changes in positioning or lighting occur, Fisherfaces is much more accurate. While the within-dataset identification was about equal at 98.63%, when a new photo was compared to the dataset, we found 89.63% accuracy without any brightness or color normalization. When the new image was normalized, this increased to 98.98% within dataset accuracy and 90.84% overall accuracy. Table I displays the accuracy results. However, despite the clearly superior identification, there are more calculations which Fisherfaces must run through before it is ready to produce results, and thus it benefits by having a training set already prepared and saved. Eigenfaces was significantly more efficient than Fisherfaces in computational time.

TABLE I

Algorithm	Within	Outside Image	Normalized
Eigenfaces	98.26%	9.67%	22.58%
Fisherfaces	98.63%	89.63%	90.84%

The accuracy of the algorithms under different conditions. Within refers to identifying an image from within the dataset, thus with identical lighting and framing, but varying expression. Outside image refers to an image taken under different lighting conditions at a later date. Normalized refers to the same set of outside images, but with brightness normalized to the original dataset.

These vastly different accuracy for out-of-dataset identification show Fisherfaces to be a much stronger algorithm when given variations in lighting. Creating vector space through LDA and grouping together classes gave Fisherfaces a more reliable classification and distance method to work with.

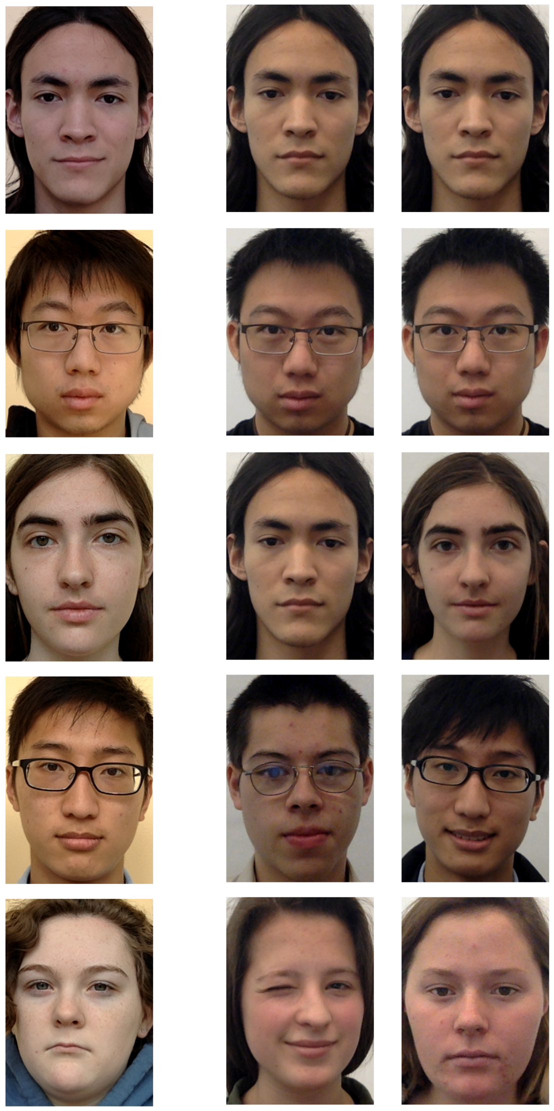

Fig. 4. The lefthand column shows the input faces for our facial recognition software. The right two columns display the results from Eigenfaces (middle) and Fisherfaces (rightmost). The top two rows display images which were correctly identified by both Eigenfaces and Fisherfaces. We can note that both of these faces maintained similar framing and angle. The next two faces show examples of Fisherfaces correctly identifying a face where Eigenfaces was unable. Here, framing, angle, and lighting varied more, but the faces chosen by Eigenfaces were still fairly similar in hair, glasses, or general face shape. The last example is of a face which both algorithms identified incorrectly, likely because of distinct changes in angle.

IV. Conclusion

After testing both Eigenfaces and Fisherfaces in facial recognition, we found Fisherfaces to be a much stronger algorithm. Using LDA rather than PCA provided a more accurate method of classification despite variations in lighting and framing. Brightness and contrast normalization improved both results significantly.

While recognizing portraits is an accomplishment on its own, future steps would include testing on significant expression variation, as most of the new images were neutral, and implementing SIFT, or scale-invariant feature transform, a method by which a computer can choose a face out of an

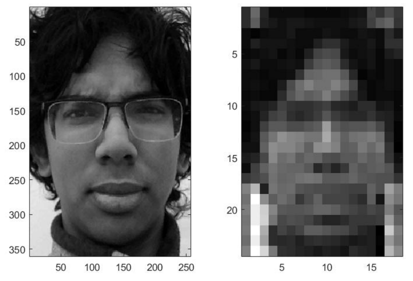

Fig. 5. One of the training set images, resampled by a factor of 15. Each pixel in the downsampled image represents the average of pixel values over the original image.

image despite position, scale, or rotation. Combining this with Fisherfaces would allow us to take almost any picture and mimic the tagging feature of Facebook. We could also develop a way to let the computer build a new class out of faces it does not recognize, thus learning over time.

V. Glossary

a) Normalization: in images is adjusting the range and magnitude of pixel intensities in order to bring it into a more familiar range. Adjusting brightness and contrast are two of the main methods of doing this.

b) Statistically Significant: refers to the amount of influence something has. The statistical significance of eigenvectors is determined by their corresponding eigenvalue. Low values show they are a small component of the full object, whereas eigenvectors with high eigenvalues indicate more weight given to that eigenvector.

c) Brightness: is a term used to compare two images. If they have similar levels of contrast, the brighter image is the one that is overall closer to white.

d) Contrast: refers to the difference between the maximum and minimum pixel values in an image.

e) Correlation: usually refers to the extent to which any two variables have a linear relationship. If they follow similar trends, they are considered positively correlated and gain a value near 1. An inverse relationship gains a value near -1, and no discernable relationship tends towards 0. A correlation matrix is an $n \times n$ symmetric matrix which compares the correlation of the columns of $m \times n$ matrix A.

f) Covariance: is a correlation matrix that is not normalized.

g) Downsampling: or *resizing* is reducing the resolution of an image by taking the average value of an area of pixels and compressing it into a single pixel, as seen in Fig. 5.

h) Eigenfaces: the eigenvectors produced by performing PCA on a covariance or correlation matrix of a dataset of images, as seen in Fig. 6. They represent the areas of greatest variance within the dataset.

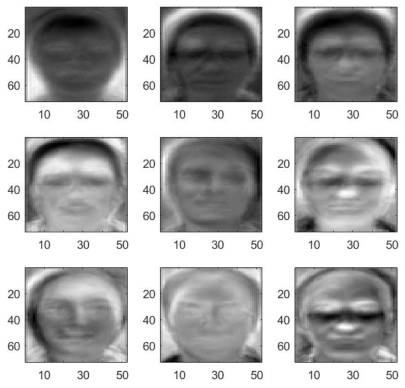

Fig. 6. The nine most statistically significant eigenfaces of our training dataset. These show the areas of greatest variance between faces, with the top left corner being the most significant and the bottom right being the least of the ones shown.

i) Fisherfaces: are the eigenvectors produced by performing LDA on a dataset of images. They are most useful in classifying images and show the areas of greatest variance between classes within the dataset.

j) Vector Space: is a collection of vectors that can be added and multiplied by scalar values in order to represent any value within the space. For example, a graph is generally a vector space created by unit vectors in the x, y, and z directions.

k) PCA: or *principal component analysis* is a dimensionality reduction strategy for eliminating features with low variance. To perform PCA, we first take the correlation matrix for the data, and then get its eigenvectors.

$$AA^{\tau} w_i = \lambda_i w_i$$

Since we're only concerned with features that have high variance, we sort λ_i and w_i, with respect to λ_i descending, and then choose how many features we want to keep.

$$m = |\lambda|$$
$$n = \text{number of principal components} < m$$
$$\lambda_1, \lambda_2, ...\lambda_n \equiv w$$

Which lets us choose features with high variance and exclude those with low variance.
SVD can be utilized for this, as it makes a signular matrix (λ on diagonal) and an AA^{τ} matrix.

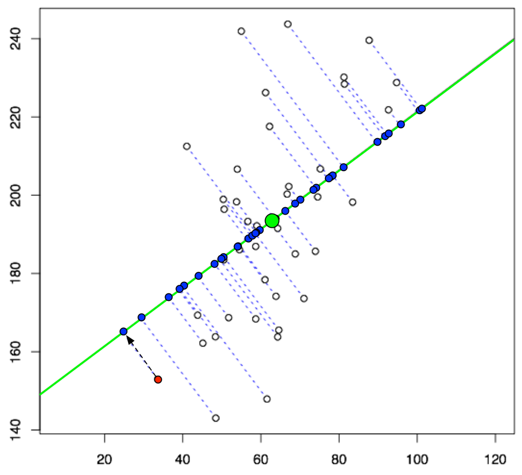

Fig. 7. The white dots represent original data, and since this is a two dimensional plot, we can see the axis with the most variance is the green line, so that is the principal component if we choose $n = 1$.

l) LDA: linear discriminant analysis is a dimensionality reduction strategy generalized from Fisher's linear discriminant. It is used to classify data by minimizing within-class matrix (covariance matrix S_w) and maximizing between-class variation (covariance matrix S_b). This is done by using the largest eigenvector of $S_w^{-1}S_b$ to produce a matrix which contains the first $C-1$ components, where C is the number of classes within the dataset.

m) Dimensionality Reduction: is representing an image as its main components rather than all of the data.

References

[1] Aleix Martinez (2011) Fisherfaces. Scholarpedia, 6(2):4282.

[2] Belhumeur, Peter N., Joao P. Hespanha, and David J. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (1997): 711-14.

[3] Wagner, Philipp. "Fisherfaces." N.p., 3 June 2012. Web. 28 Apr. 2016.