FISHERFACES UNDER SIMULATED PERSPECTIVE DISTORTIONS

NATALIA PACHECO-TALLAJ AND TOBY SATTERTHWAITE
DECEMBER 11, 2019

1. Introduction

P.Belhumeur, et al.'s 1997 paper *Eigenfaces vs. Fisherfaces: Recognition Using Class-Specific Linear Projection* proposes an improvement upon previous face-recognition algorithms to better account for variation in photos of the same person in a data set. They consider images as column vectors $\vec{x}_1, \ldots, \vec{x}_N$ in a high-dimensional space $\mathbb{R}^n$ (where n is dependent on number of pixels). The previous face-recognition method that the paper improves upon, called Eigenfaces, uses principal component analysis (PCA) to map the $\vec{x}_k$ to lower-dimensional *feature vectors* $\vec{y}_k = W^T \vec{x}_k$ such that the scatter of the feature vectors is maximized:

$$W_{pca} = \arg\max_W |W^T S_T W| \tag{1}$$

where S_T is the total scatter matrix of the $\vec{x}_k$, $S_T = \sum_{i=1}^{N} (\vec{x}_k - \vec{\mu})(\vec{x}_k - \vec{\mu})^T$, where $\vec{\mu}$ is the image mean.

Usually images are divided into a number of classes $X_1, \ldots, X_c$ (in the case of face recognition, each class is a different person). The issue with Eigenfaces is that by maximizing a total scatter function, it is not only maximizing scatter between image classes but also within image classes. This is particularly harmful if we are working with datasets with variation in lighting conditions within each class, because the Eigenfaces (principal components) retain this variation information. The paper proposes a solution, called Fisherfaces, using the Fisher linear discriminant (FLD) method to extract the feature vectors, which seeks to solve the problem of maximizing between-class scatter of the projected feature vectors $\vec{y}_k$ while minimizing within-class scatter:

$$W_{fld} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{2}$$

where S_W is the within-class scatter matrix and S_B the between-class scatter matrix of the $\vec{x}_k$:$S_W = \sum_{i=1}^{c} N_i(\vec{\mu}_i - \vec{\mu})(\vec{\mu}_i - \vec{\mu})^T$ and $S_B = \sum_{i=1}^{c} \sum_{\vec{x}_k \in X_i} (\vec{x}_k - \vec{\mu}_i)(\vec{x}_k - \vec{\mu}_i)^T$, where $\vec{\mu}_i$ is the i-th class's mean and N_i the number of images in class i.

The authors test the performance of their algorithm against variation in lighting and facial expression, but does not test for other important variation in facial recognition applications such as perspective distortion. Our goal is to test how the Fisherfaces algorithm proposed in the paper fares against different kinds of

increasingly drastic perspective distortion. We apply different kinds of affine transformations with increasing deviation from the identity to our test images and quantify the loss of accuracy with these perspective distortions.

2. Methods

In face recognition applications, there is a problem with solving Equation 2 directly: the rank of S_W is bounded above by the number of N images in the training set, much smaller than the length n of the image vectors which determines the size $n \times n$ of S_W. Then S_W is singular and $|W^T S_W W|$ can be exactly 0. To solve this issue, we must first reduce the dimensions of the data set with PCA so that it equals the number of data points, and then apply FLD to the reduced dataset:

$$W_{pca} = \arg\max_W |W^T S_T W|$$
$$W_{fld} = \arg\max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

Computing scatter matrices is very expensive. With our dataset, images of about $300,000$ pixels each would yield $300,000 \times 300,000$ scatter matrices (or about 90 gigabytes). However, we can avoid computing any of the scatter matrices by using SVD to find the dimension-reduced feature vectors. Below we show how to perform the PCA and FLD steps of the paper's Fisherfaces algorithm using SVD.

2.1. PCA step with SVD.

The problem

$$W_{pca} = \arg\max_W |W^T S_T W|$$

is solved by the matrix $W = \begin{pmatrix} \vec{w}_1 & \dots & \vec{w}_m \end{pmatrix}$ where $\vec{w}_k$ are the m eigenvectors corresponding to the m largest eigenvalues of S_T. We can express S_T as $X^T X$ where

$$X = \begin{pmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_N^T \end{pmatrix} - \begin{pmatrix} \vec{\mu}^T \\ \vdots \\ \vec{\mu}^T \end{pmatrix}$$

is the matrix with the vectorized images minus the mean image as its rows. Now if we take the SVD of X we get $X = UDV^T$ with S the diagonal matrix of singular values of X in decreasing order. The singular values of X are by definition the square roots of the eigenvalues of $X^T X = S_T$, so D^2 is the matrix of eigenvalues of S_T. Therefore:

$$S_T = X^T X = VDU^T UDV^T = VD^2 V^T = VD_T V^{-1}$$

(U, V are unitary) so $S_T V = VD_T$, then the columns of V are the eigenvectors. Thus the first m columns of V give us the solution to 1, and the feature vectord y_k are then the rows of XV. If we set $m = N$ to be the amount of images, we can do this with numpy by taking

```
_, _, vh = np.linalg.svd(X, full_matrices=False)
W = vh.transpose()
```

```
# the rows of Y are the feature vectors,
# there are N such vectors of length N each
Y = np.dot(X, V)
```

2.2. **FLD with SVD.** The problem

$$W_{fld} = \arg\max_W \frac{W^T W_{pca}^T S_B W_{pca} W}{W^T W_{pca}^T S_W W_{pca} W} \tag{3}$$

is solved by the matrix $W = \begin{pmatrix} \vec{w}_1 & \dots & \vec{w}_m \end{pmatrix}$ where $\vec{w}_k$ are the generalized eigenvectors with largest generalized eigenvalues (they solve the equation $S_b \vec{w}_k = \lambda_k S_w \vec{w}_k$). To replicate what we did in the previous part we want to express $W_{pca}^T S_B, W_{pca}^T S_W W_{pca}$ as some matrix times its transpose. Let

$$M = \begin{pmatrix} \vec{\mu}_1^T \\ \vdots \\ \vec{\mu}_c^T \end{pmatrix} - \begin{pmatrix} \vec{\mu}^T \\ \vdots \\ \vec{\mu}^T \end{pmatrix}$$

then

$$S_B = M^T M \text{ and } W_{pca}^T S_B W_{pca} = (MW_{pca})^T (MW_{pca})$$

and let

$$K = \begin{pmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_i^T \\ \vec{x}_{i+1}^T \\ \vdots \vec{x}_N^T \end{pmatrix} - \begin{pmatrix} \vec{\mu}_1^T \\ \vdots \\ \vec{\mu}_1^T \\ \vec{\mu}_2^T \\ \vdots \\ \vec{\mu}_c^T \end{pmatrix}$$

where we are assuming the images are ordered by which class they belong to, then

$$S_W = K^T K \text{ and } W_{pca}^T S_W W_{pca} = (KW_{pca})^T (KW_{pca})$$

To simplify notation from now on we let S_B, S_B denote $W_{pca}^T S_B, W_{pca}^T S_W W_{pca}$ and M, K denote MW_{pca}, KW_{pca}. Here's the solution to the generalized eigenvalue problem without computing scatter matrices.

Step 1. Find V_W that diagonalizes S_W by taking the SVD $K = U_W D_W V_W^T$. By what we showed in section 1, $V_W^T S_W V_W = D_W^2$.

Step 2. Let

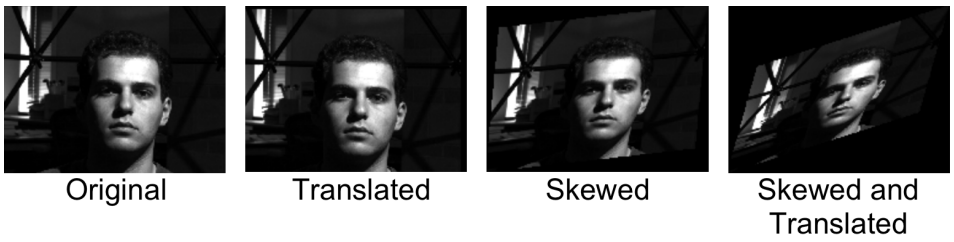
$$\begin{aligned} V'_W &= V_W D_W^{-1} \\ M' &= M V'_W \\ S'_B &= {V'_W}^T S_B V'_W = {M'}^T M' \end{aligned}$$

where D_W^{-1} is obtained by inverting each element in the diagonal. Then $V_W'^T S_W V'_W = D_W^{-1} V_W^T S_W V_W D_W^{-1} = D_W^{-1} D_W^2 D_W^{-1} = I$.

Step 3. Find V_B that diagonalizes S'_B by taking the SVD $M' = U_B D_B V_B^T$, so that

FIGURE 1. One human subject in our data set under different expressions and lighting conditions.

FIGURE 2. One sample from our data set as it is translated, skewed, and both translated and skewed.



$V_B^T S_B' V_B = D_B^2$.

Step 4. Let

$$V = V_W' V_B$$

We can then see that $V^T S_B V = V_B^T V_W'^T S_B V_W' V_B = V_B^T S_B' V_B = D_B^2$ and $V^T S_W V = V_B^T V_W'^T S_W V_W' V_B = V_B^T I V_B = I$, so

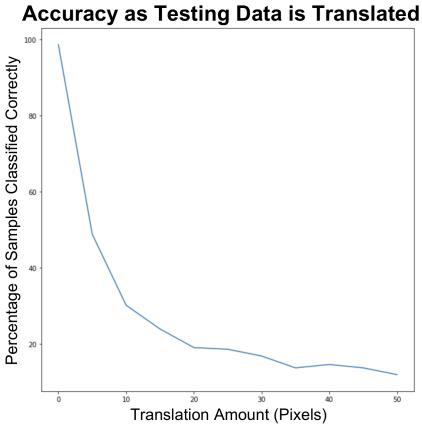
$$S_B V = S_W V D_B^2$$

where D_B is diagonal, from this equation we can see that the i-th column of V is a generalized eigenvector with generalized eigenvalue the i-th diagonal entry of D_B^2. So the first m columns of V give us the solutions to 2.

2.3. **Data Set.** We test this approach on The Extended Yale Face Database B. This data set contains 28 human subjects who are each photographed in nine poses under 64 different lighting conditions for a total of 16128 images. For the purposes of running the simulation on a laptop computer, however, we limited ourselves to 15 classes (people) with 20 randomly selected samples per person for training and 15 randomly selected samples per person for testing.

The two methods of distortion that we use to simulate perspective effects are translation and skew. We translate images in the testing set by moving them by a certain number of pixels in the x or the y dimension, and we quantify the "amount of translation" by an upper bound t in the total translation distance. Zero padding is used to fill in values. In order to skew each image, we create an affine transformation by rotating the image by a random angle between 0 and 2π, scaling it by different factors in both dimensions, and then rotating it back by the same angle. We quantify the "amount of skew" as a number p such that both scaling factors are in $[\frac{1}{p}, p]$. Examples of these effects can be seen in figure 2.

FIGURE 3. Accuracy as testing set is translated by different pixel amounts



3. RESULTS

In order to train and test the model on a laptop computer, we limited our data set to only 15 classes (people), with 20 randomly selected samples from each class used for training, and 15 randomly selected samples from each class used for testing. Without any of our distortions, the model correctly classified 222 out of 225 testing set samples, or 98.7%.

We then translated each of the samples in our testing set by a random number of pixels between 0 and n, with $n \in \{0, 5, 10, ..., 50\}$. This yielded a sharp decrease in the number of samples which were correctly classified, as can be seen in figure 3.

We separately simulated distortion arising from perspective effects by skewing each image. As before, we applied this skew to each of the testing set images, with the skew factor randomly assigned to a number between $1/n$ and n for $n \in \{2, 3, ..., 10\}$. We present our findings in figure 4. As with translation, there is a sharp drop-off in the algorithm's performance, however the accuracy is universally much lower. With a very high skew, the algorithm's performance approaches that of randomly assigning each image to one of the 15 classes (6.67%).

Finally, we simulated both distortion and translation by applying both the skew and the translation effects to each of the images in the testing set. For integer $n \in \{0, 1, ..., 8\}$, we translated each image by a random number of pixels between 0 and $10 + 5n$ and skewed it by skew factor $2 + n$. Our results, seen in figure 5, closely resemble those of skew-only distortion, however at each step, the accuracy is roughly one percentage point lower without the addition of translation.

4. CONCLUSION

While we find that the algorithm performs remarkably well on our unaltered data set, as had been predicted in the original paper, this performance does not extend well to our simulated distortion effects. Under translation only, algorithm performance decreases sharply after each image is translated just a few pixels in the

FIGURE 4. Accuracy as testing set is skewed by different factors

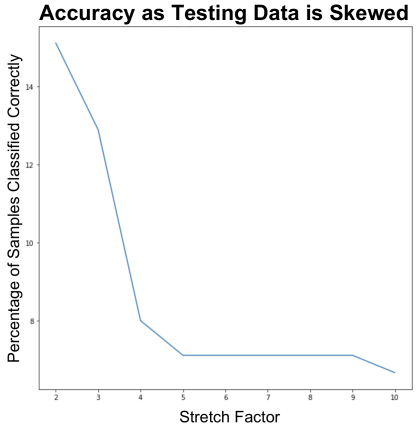
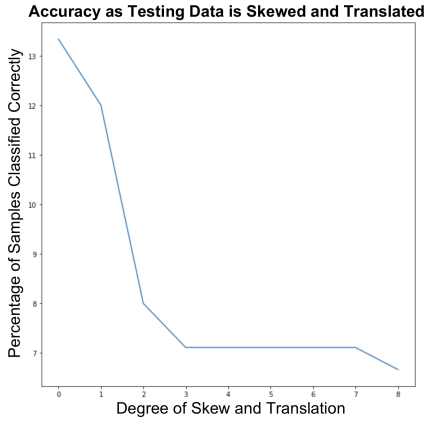


FIGURE 5. Accuracy as testing set is skewed and translated by different factors and pixel values



x or the y directions. Under simulated skew effects, the performance loss is even more drastic as accuracy quickly approaches that of random classification to one of the 15 classes. Finally, composing both skew and rotation creates nearly identical behavior with slightly worse accuracy at each step.

Looking forward, significant changes to the algorithm are needed in order to correctly identify faces under the distortion created by projecting three-dimensional perspective effects onto a two-dimensional image. Though its performance is promising on staged data sets such as The Extended Yale Database B, real world applications of facial recognition require a system that is more robust to perspective alterations.

5. Contribution Statement

I started a first implementation of the paper and realized that the scatter matrices were too big to compute, so going off of Dor's advice, I figured out how to use SVD to bypass computing the scatter matrix when solving the generalized eigenvalue problem necessary for the Fisher linear discriminant method. I wrote some pseudocode for FLD, which Toby then used to finish the implementation of FLD. We added the k means clustering together and ran the first tests together. We went to Prof. Zickler's office hours to ask about meaningful tests to run for our extension and proposed our idea. After getting his aproval, we used my pset 2 code for homography application and I wrote the random distortion and translation generator, and generated the graph for accuracy with increasing translation. Toby then generated the graphs for random distortion and random distortion + translation. We wrote the final report together in a meeting. Throughout, we mostly worked together during meetings so we helped debug and correct errors in each other's code. We both spent the same amount of time outside of meetings working on the project.

6. References

P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 711-720, July 1997.

The Extended Yale Face Database B, Yale. 2001. http://vision.ucsd.edu/ iskwak/ExtYale-Database/ExtYaleB.html.

R. Wang, âĂIJGeneralized Eigenvalue Problem.âĂİ Generalized Eigenvalue Problem, Harvey Mudd College, 27 Apr. 2015, fourier.eng.hmc.edu/e161/lectures/algebra/node7.html.