# Extended Yale Faces B Database – Eigenface

*Author: Seoyoung Park*

## Abstract

Linear algebra plays an important role of application in several areas such as physics, engineering, and biological sciences. It is also helpful in data analysis and computation. In linear algebra, the idea of matrix transferring a vector through multiplication is defined. A new vector y, for instance, is set when you multiply a vector x by a matrix A, which produces a new direction with a new length. (y = Ax) There is a concept called singular value decomposition (SVD), and it is one of most useful methods for analyzing multiple applications.

## Sec. I. Introduction and Overview

In this paper, the goal is to perform an SVD analysis of real data sets, cropped images from Extended Yale Faces B Database. The matrix multiplication, as shown earlier, does two main things to a given vector, rotation and stretching, and they are controlled by a given matrix.

## Sec. II. Theoretical Background

A singular value decomposition (SVD) is a matrix factorization into some constitutive components that all have a meaning in applications. The SVD provides a transformation of the given set of vectors to stretch or compress and rotate. A hyperellipse in $\mathbb{R}^m$ is the surface resulted from stretching a unit sphere in $\mathbb{R}^m$ by some factors $\sigma_1, \sigma_2, \dots, \sigma_m$ in the orthogonal directions $u_1, u_2, \dots, u_m \in \mathbb{R}^m$. If the matrix A is an m × n matrix (m < n),

$$Av_j = \sigma_j u_j \; 1 \le j \le n$$

The form of SVD decomposition is

$$AV = U\Sigma$$

$$A = U\Sigma V^*$$

with the following matrices

$$U \in \mathbb{C}^{m\times m} \text{ (unitary)}$$

$$V \in \mathbb{C}^{n\times n} \text{ (unitary)}$$

$$\Sigma \in \mathbb{R}^{m\times n} \text{ (diagonal)}$$

Firstly, the matrix A applies a unitary transformation by $V^*$ maintaining the unit sphere. It's followed by $\Sigma$ that stretches and creates an ellipse with principal semi axes. Then, U rotates the created hyperellipse. To compute the SVD, the following matrix products result in eigenvalue problems.

$$A^T A = (U\Sigma V^*)^T (U\Sigma V^*) \quad (1)$$

$$= V\Sigma U^* U\Sigma V^*$$

$$= V\Sigma^2 V^*$$

$$A^T AV = V\Sigma^2$$

$$AA^T = (U\Sigma V^*)(U\Sigma V^*) \quad (2)$$

$$= U\Sigma V^* V\Sigma U^*$$

$$= U\Sigma^2 U^*$$

$$AA^T U = U\Sigma^2$$

We can find the orthonormal basis vectors for U and V when we find the normalized eigenvectors for these two equations. The SVD produces a sort of least-square fitting algorithm that let us to make the matrix onto low-dimensional representations in a type of algorithm.

Now lets consider there is a simple mass-spring system, and we are trying to record the motion of the mass with three cameras. Assume the mass is moving by a small perturbation in the z-direction. Since we have 3 cameras to observe the motion, the data sets would have redundancy and noise. However, with principal component analysis (PCA), we can come up with an simple version of the data sets. Let's say each camera gives a two-dimensional data such as camera 1 : ($x_a$, $y_a$), camera 2: ($x_b$, $y_b$), camera 3: ($x_c$, $y_c$). These can be expressed in a single matrix

$$X = \begin{bmatrix} x_a \\ y_a \\ x_b \\ y_b \\ x_c \\ y_c \end{bmatrix}$$

$X \in \mathbb{R}^{m \times n}$ where m is the number of measurement types, n is the number of data points the camera take over time. To get the idea of redundancy, we need to get the covariance in data sets. Let a = $[a_1 \quad a_2 \quad \dots \quad a_n]$ and b = $[b_1 \quad b_2 \quad \dots \quad b_n]$.

The variances of a and b are

$$\sigma_a^2 = \frac{1}{n-1} aa^T$$

$$\sigma_b^2 = \frac{1}{n-1} bb^T$$

, and the covariance between a and b is

$$\sigma_{ab}^2 = \frac{1}{n-1} ab^T$$

If the value of covariance is very small, it means the pair is statistically independent, and if it's big, it means statistically dependent which gives redundancy. A covariance matrix can be expressed as

$$C_X = \frac{1}{n-1} XX^T$$

Diagonal entries of the covariance matrix are variance measures, and off-diagonal entries, which are symmetric, self-adjoint, and Hermitian, are covariance between all pairs. The main goals here are to remove redundancy and get signals with maximal variance. If we diagonalize the covariance matrix, we can reach two goals, which the SVD does. If X is a data matrix,

$$XX^T = S\Lambda S^{-1}$$

Where S is a matrix of eigenvectors of $XX^T$(note $S^{-1} = S^T$), and Λ is a diagonal matrix with eigenvalues of $XX^T$. The process of diagonalization of the covariance matrix is following

$$Y = S^T X$$

$$C_Y = \frac{1}{n-1} YY^T$$

$$= \frac{1}{n-1} S^T X (S^T X)^T$$

$$= \frac{1}{n-1} S^T X X^T S$$

$$= \frac{1}{n-1} S^T S \Lambda S^T S$$

$$= \frac{\Lambda}{n-1}$$

Since $\Lambda = \Sigma^2$

$$C_Y = \frac{\Sigma^2}{n-1} \text{ (diagonalized, removing redundancy)}$$

## Sec. III. Algorithm Implementation and Development

For face recognition, PCA is used as a toll to explore the given data and to exhibit the internal structure of the data that illustrate the major variance in the data. (In this paper, PCA transfers a set of face images to a set of uncorrelated variables - eigenfaces)

1. **Convert the face images into the face vectors.**
   (M for cropped images and X for uncropped images)
   Each column vector of the matrix M corresponds to each image from 'Cropped Yale Database', and each column vector of the matrix X corresponds to each image from 'Uncropped Yale Database'
2. **Find common vector (average face vector) every face has average face feature**
   The average face of the set is sum of all faces divided by the number of faces, as shown in figure (A).

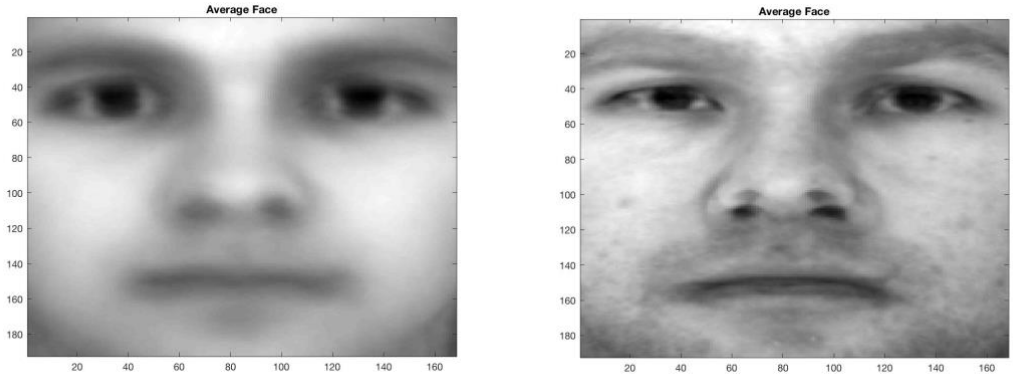


Figure (A) : Left image shows the average face of Cropped Yale database, and right image shows the average face of Uncropped Yale database.

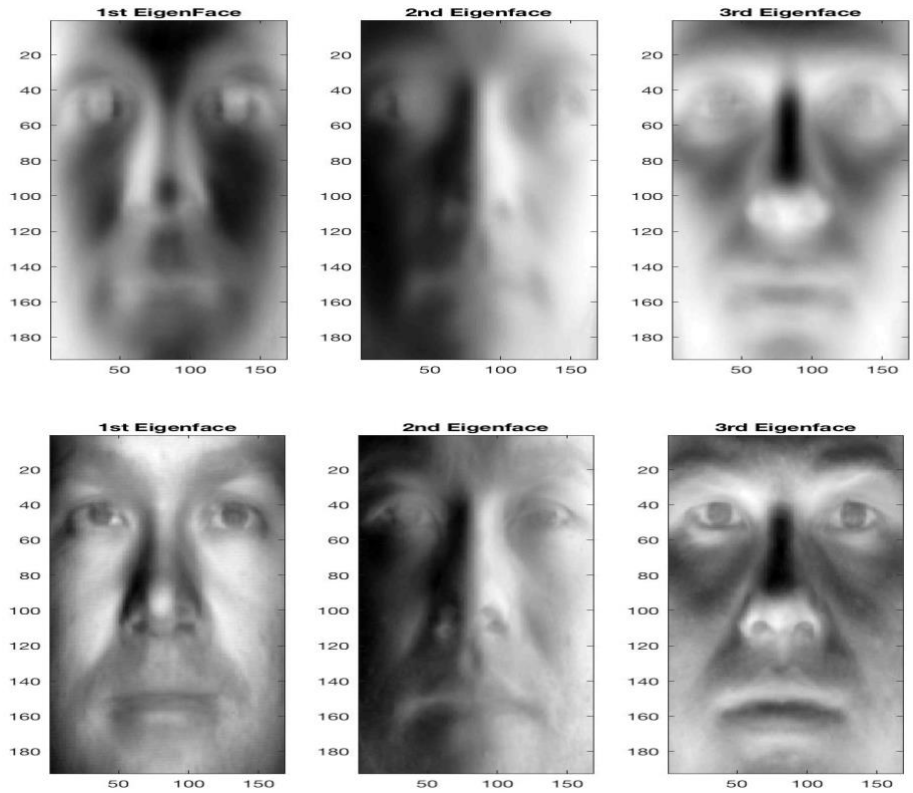


Figure (B): Upper three faces are the first three eigen faces of Cropped Yale database, and Lower three faces are the first three eigen faces of Uncropped Yale database.

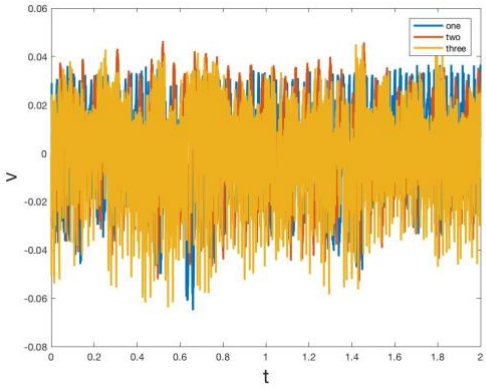


Figure (C): The graph demonstrates what each first three vectors of unitary transformation matrix V of Cropped Yale database do in time for 2 seconds.

3. **Subtract average face from each face from the data, then we can get the normalized face vectors.**
4. **Reduce the dimension of the data sets (SVD), and get eigenvectors, score, and eigen values.**
   As shown in figure (B), the pictures are the examples of eigenfaces of each database. Each picture is resulted from the first three eigenvectors $(u_1, u_2, u_3)$ that are constructed by singular value decomposition.
5. **Select the best eigenfaces that can represent whole data sets.**

## Sec. IV. Computational Results

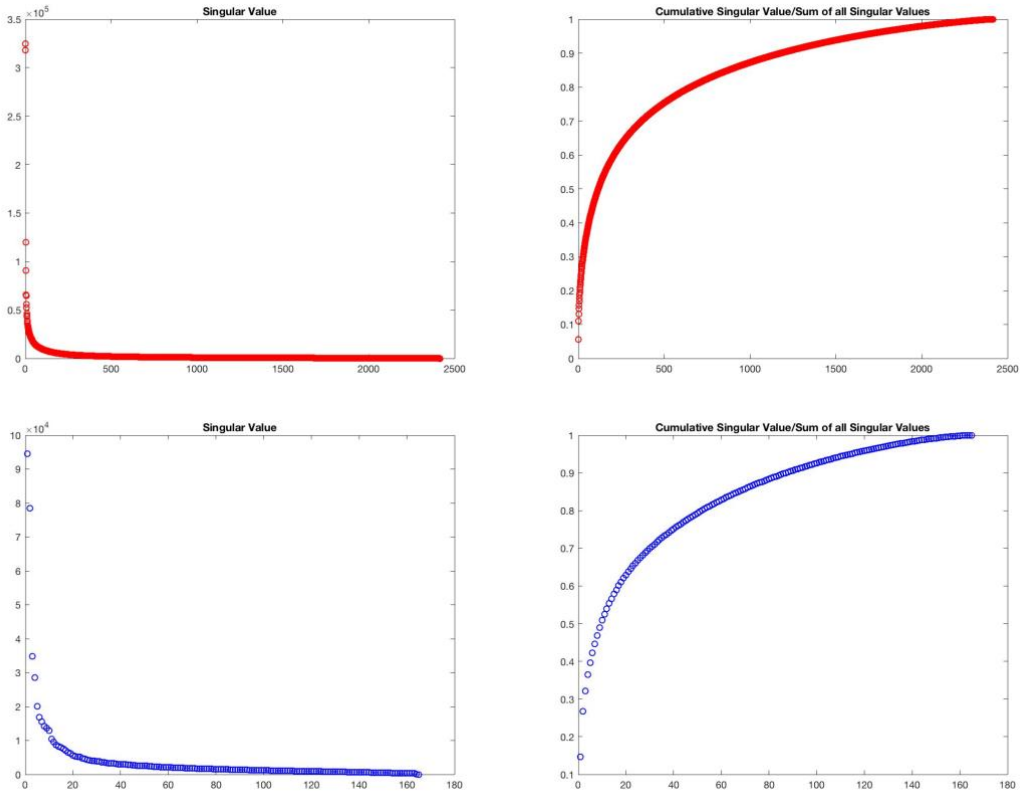


Figure (D): Upper two graphs illustrate the singular value spectrum of Cropped Yale database/ Lower two graphs illustrate the singular value spectrum of Uncropped Yale database.

As shown in figure (D), the graph shows the singular values of the matrix M and X(Dataset of Cropped and Uncropped Yale face images). First some singular values of Cropped pictures are varied more than those of Uncropped picture. They tell us that a 1180 mode approximation of Cropped Yale dataset and a 88 mode approximation of Uncropped Yale dataset produce 90% of the energy. Also, to produce 99% of energy, a 2178 mode approximation of Cropped Yale and a 148 mode approximation of Uncropped Yale are needed. The higher rank it is, the better picture is produced.

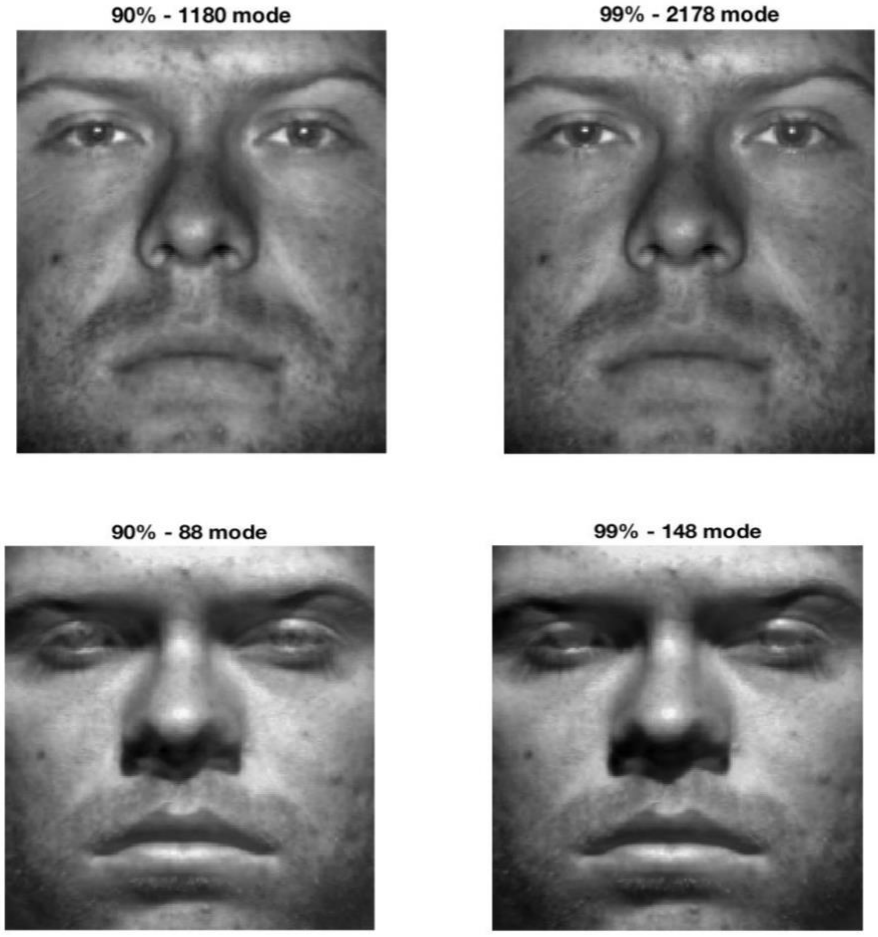


Figure (E): Up: a 1180 mode approximation and a 2178 mode approximation of Cropped Yale
Down: a 88 mode approximation and a 148 mode approximation of Uncropped Yale

## Sec. V. Summary and Conclusions

As understanding singular value decomposition, we can apply it to principal component analysis, and two acknowledge have been used to analyze datasets from both Uncropped Yale images and Cropped Yale images. Datasets, the matrix M and X, can be constructed with two basis U and V and a diagonal matrix Σ that each entry represents the singular value. With three matrices, we can reconstruct the image and get to know what rank is good to reconstruct a good image of dataset.

## Appendix A

**D = dir('NAME')** returns the results in an M-by-1 structure with the fields name, folder, date, bytes, isdir, datenum
**B = repmat(A,M,N)** or **B = repmat(A,[M,N])** creates a large matrix B consisting of an M-by-N tiling of copies of A.
**imagesc** displays image with scaled colors
**reshape(X,M,N)** or **reshape(X,[M,N])** returns the M-by-N matrix whose elements are taken columnwise from X
**I = find(X)** returns the linear indices corresponding to the nonzero entries of the array X
**imshow(I,[])** displays the grayscale image I scaling the display based on the range of pixel values in I
**DIRECTORYNAME = uigetdir(STARTPATH, TITLE)** displays a dialog box for the user to browse through the directory structure and select a directory, and returns the directory name as a string.
**F = fullfile(FOLDERNAME1, FOLDERNAME2, ..., FILENAME)** builds a full file specification F from the folders and file name specified.

## Appendix B

```
clear all; close all; clc;
%% Cropped Yale
files = dir('**/*.pgm');
M = zeros(32256,2414); % data matrix
for i = 1:length(files)
   imageArray = imread(strcat(files(i).folder,'/',files(i).name));
   imageVector = imageArray(:);
   M(:,i) = imageVector;
end

% mean face
avgM = mean(M,2);

% substrat the mean from the original dataset matrix
M = M - repmat(avgM, 1, 2414);

% singular value decomposition
% get eigenvectors, score, and eigenvalues
[U,S,V] = svd(M,'econ');

eigVal = diag(S);
for i = 1:2414
s(i) = sum(eigVal(1:i));
end
energy = s./s(end);

figure; plot(diag(S),'ro'); title('Singular Value');
figure; plot(energy,'ro'); title('Cumulative Singular Value/Sum of all Singular Values');

%image of mean face
figure; imagesc(reshape(avgM, 192, 168)); title('Average Face'); colormap(gray);

% image of first three eigen faces
subplot(1, 3, 1); imagesc(reshape(U(:, 1), 192, 168)); colormap(gray); title('1st EigenFace');
```

```
subplot(1, 3, 2); imagesc(reshape(U(:, 2), 192, 168)); colormap(gray); title('2nd Eigenface');
subplot(1, 3, 3); imagesc(reshape(U(:, 3), 192, 168)); colormap(gray); title('3rd Eigenface');

% shows what each line is doing in time
t = linspace(0,2,2414);
figure; plot(t,V(:,1:3),'Linewidth',[2]); title(''); legend('one','two','three');
xlabel('t','Fontsize',[20])
ylabel('v','Fontsize',[20])

% Determine the number of principal components required to model 90% of data variance
count = min(find(energy > 0.9));
S_rank = S;
S_rank(count+1:end,count+1:end) = 0;
A_rank = U*S_rank*V';
subplot(1,2,1); imshow(reshape(A_rank(:,1),[192 168]),[]); title('90% - 1180 mode')

% Determine the number of principal components required to model 99% of data variance
count_ = min(find(energy > 0.99));
S_rank_ = S;
S_rank_(count_+1:end,count_+1:end) = 0;
A_rank_ = U*S_rank_*V';
subplot(1,2,2); imshow(reshape(A_rank_(:,1),[192 168]),[]); title('99% - 2178 mode')
%% Uncropped Yale
myFolder = uigetdir('', 'Select the directory');
% Check to make sure that folder actually exists. Warn user if it doesn't.
if ~isdir(myFolder)
errorMessage = sprintf('Error: The following folder does not exist:\n%s', myFolder);
uiwait(warndlg(errorMessage));
return;
end
% Get a list of all files in the folder with the desired file name pattern.
filePattern = fullfile(myFolder);
theFiles = dir(filePattern);

X = [];
for k = 1 : length(theFiles)
   imageArray = imread(strcat(files(k).folder,'/',files(k).name));
   imageVector = imageArray(:);
   X(:,k) = imageVector;
end
X = X(:,3:end);

% mean face
avgX = mean(X,2);

% substrat the mean from the original dataset matrix
X = X - repmat(avgX, 1, 165);

% singular value decomposition
% get eigenvectors, score, and eigenvalues
[U2,S2,V2] = svd(X,'econ');

eigVal2 = diag(S2);

for i = 1:165
s2(i) = sum(eigVal2(1:i));
end
```

```
energy2 = s2./s2(end);
figure; plot(diag(S2),'bo'); title('Singular Value');
figure; plot(energy2,'bo'); title('Cumulative Singular Value/Sum of all Singular Values');

%image of mean face
figure; imagesc(reshape(avgX, 192, 168)); title('Average Face'); colormap(gray);

% image of first three eigen faces
subplot(1, 3, 1); imagesc(reshape(U2(:, 1), 192, 168)); colormap(gray); title('1st Eigenface');
subplot(1, 3, 2); imagesc(reshape(U2(:, 2), 192, 168)); colormap(gray); title('2nd Eigenface');
subplot(1, 3, 3); imagesc(reshape(U2(:, 3), 192, 168)); colormap(gray); title('3rd Eigenface');

% average face
figure; imagesc(reshape(avgX, 192, 168)); title('Average Face'); colormap(gray);

% Determine the number of principal components required to model 90% of data variance
count2 = min(find(energy2 > 0.9));
S_rank2 = S2;
S_rank2(count2+1:end,count2+1:end) = 0;
A_rank2 = U2*S_rank2*V2';
subplot(1,2,1); imshow(reshape(A_rank2(:,1),[192 168]),[]); title('90% - 88 mode')

% Determine the number of principal components required to model 99% of data variance
count2_ = min(find(energy2 > 0.99));
S_rank2_ = S2;
S_rank2_(count2_+1:end,count2_+1:end) = 0;
A_rank2_ = U2*S_rank2_*V2';
subplot(1,2,2); imshow(reshape(A_rank2_(:,1),[192 168]),[]); title('99% - 148 mode')
```