Homework 1: Extended Yale Faces B Database Eigenfaces

Eunji Lindsey Lee
(*e-mail:* elee0025@uw.edu)

Abstract

For this homework, we performed an Singular Value Decomposition (SVD) analyses on the data sets of Yale Faces. First, we started with cropped images of faces, then did further analysis for uncropped images and eventually made comparisons between two analyses.

Contents

1 Introduction and Overview

Linear Algebra plays a significant role in many mathematical applications in numerous different fields such as computer science, biological sciences, economics and psychology. Such applications are widely used in data analysis and computation, including digital image processing. One of the widely known mathematical techniques of image processing is Singular Value Decomposition (SVD). SVD is widely used for image processing such as image compression, noise reduction, and image recognition. The focus of this paper is the applications of SVD on data sets of cropped and uncropped images of popular face data sets called Yale Faces and perform various analyses.

Cropped images of Yale faces are the collection of facial images, which barely shows the whole face. These images focuses only on the facial features such as eyes, nose, mouth,

eyebrow, chin and forehead. They do not reflect other features such as facial shapes, ears, hair styles or facial expressions. Whereas in uncropped images, there are more variations among these pictures in terms of their facial expressions, different lighting and accessories such as glasses, so uncropped images include more feature information than cropped images. Uncropped images also include the whole face, showing all facial features as shown in cropped images, together with the background. Overall, there are a total of 2432 cropped images of 38 individuals and a total of 165 uncropped images of 15 individuals, which might include corrupted images.

In this paper, we will explain the basic theory of Singular Value Decomposition (SVD) and the algorithm implementation and development of the applications of SVD on Yale faces database.

2 Theoretical Background

2.1 Singular Value Decomposition (SVD)

SVD is about obtaining coordinate systems. When it comes to represent the data, the objective of SVD is to represent data matrix into "readable" matrix components so that a data matrix, which was initially "meaningless", now have a specific meaning in applications. Any m x n matrix, say A, can be decomposed by SVD into three matrices:

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

where $U \in \mathbb{C}^{mxm}$ and $V \in \mathbb{C}^{nxn}$ are unitary and $\Sigma \in \mathbb{R}^{mxn}$ is diagonal. The diagonal entries of Σ are called singular values (SV) of A ($\sigma_1, \sigma_2, ..., \sigma_r$, where r =rank of A=number of nonzero singular values). They are assumed to be non-negative and are in decreasing order. ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$). Moreover, singular values $\sigma_1, \sigma_2, ..., \sigma_r$ are unique, whereas matrices U and V are not.

As matrices U and V are unitary, column vectors of these matrices are linearly independent and the order of these vectors correspond to the order of SVs in Σ. Each of these linearly independent components are said to bear its own energy contribution. Then the purpose of SVD is to get a system such that maximal energy is stored in as few SVs as possible. A largest SV is said to store the maximum energy contained in the image.

The geometrical interpretation of SVD of A is that matrix A is rotated by unitary transformation V^*, stretched by Σ and finally, rotated again by unitary transformation U.

2.1.1 An Eigenvalue Decomposition And A Singular Value Decomposition

An eigenvalue decomposition and a singular value decomposition are very closely related. Let's consider a singular matrix A with eigenvalues and eigenvectors (not every matrices have eigenvalues and eigenvectors). If we say S is a matrix whose columns are a collection of eigenvectors and Λ is a matrix whose diagonals are the corresponding eigenvalues, then A can be written as follows:

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

We call this an eigenvalue decomposition of A. From this, we can also conclude the following: $A^m = S\Lambda^m S^{-1}$. Since any matrices can be decomposed by SVD, we can say

that $A = U\Sigma V^*$ as well. Then by computation:

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

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

By comparing these with eigenvalue decomposition, we can conclude that U and V are columns of eigenvectors and square root of eigenvalues are singular values (SV). i.e., $\Lambda = \Sigma^2$, thus square root of $\Lambda = \Sigma$. Furthermore, we can also conclude that V diagonalizes $A^T A$, which means the columns of V are the eigenvectors of $A^T A$. Similarly, U diagonalizes AA^T, which means the columns of U are the eigenvectors of AA^T and form an orthonormal basis for Range(A).

The reason for the existence of SVD for any matrices, but not necessarily of eigenvalue decomposition for any matrices lies in the number of bases for decomposition. SVD performs diagonalization with two different orthonormal bases, U and V, while an eigenvalue decomposition performs diagonalization only on one basis, S, which are not generally orthogonal.

2.1.2 Theorems of SVD

There are several key theorems to remember for SVD, which may or may not be relevant for the applications of SVD on our data set.

Theorem. If the rank of A is r, then there are r nonzero singular values.

Theorem. The $\|A\|_2 = \sigma_1$

Theorem. A is the sum of r rank-one matrices

$$A = \sum_{j=1}^{N} \sigma_j u_j v_j^*$$

This is called low-rank approximations.

Theorem. For any N so that $0 \leq N \leq r$, we can define the partial sum

$$A_N = \sum_{j=1}^{N} \sigma_j u_j v_j^*$$

And if $N = \min\{m,n\}$, define $\sigma_{N+1} = 0$. Then,

$$\|A - A_N\|_2 = \sigma_{N+1}$$

This theorem tells us that SVD gives "a type of least-square fitting algorithm". It allows us to "project the matrix onto low-dimensional representations in a formal, algorithmic way".

2.1.3 Comparing SVD with Principle Component Analysis (PCA)

The difference between SVD and PCA essentially lies in the "pre-process" of the dataset. The purpose of PCA is to remove redundancy and noises from the data and to normalize the data before applying the concept, so that PCA would be independent of scaling of variables.

Removing redundancy in the data is critical. In order to remove redundancy and to consider variance and covariance between data sets, we find the covariance matrix of a

data set. The variance and covariance shows a statistical relationship between two vectors. A covariance score shows how much of two vectors are in the same direction (min score of 0 tells us that two vectors are independent, i.e., they are orthogonal, and maximum score of 1 tells us that two vectors are statistically dependent). Variance score(σ), on the other hand, measures how vigorous dynamics are in each directions of vectors.

The covariance matrix of n by n data matrix X is as follows:

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

This matrix is square and symmetric, whose diagonal terms are the variance scores and off-diagonal terms are covariance scores between all pairs of vectors.

Assuming that big diagonal terms are the ones that matter, doing SVD on the covariance would remove all redundancies and tell us which directions/variances matter.

PCA is especially useful when we have two or more of different types and units of data and when we want to find correlations between them. However, the application of PCA is not useful or even worse when specific scaling or variance of variables matters and therefore, do not require standardization.

For the following applications for this paper, PCA on our facial image data set seems unnecessary because our face data set has already been ”normalized” in a sense that most of the images show similar features and are all in grey-scale. However, we will figure out if our assumption would agree with the data analyses.

3 Algorithm Implementation and Development

Pixels or picture elements of image is thought of as a basic unit of pictures. Each pixels of an image represents the color, and this numeric representation of picture is stored as a two-dimensional array or a real number matrix in computer. Transforming an image into a matrix is fairly easy with coding. However, we need a certain mechanism to arrange and order a stack of images to form a single matrix in order to process and thus perform a SVD analysis.

Firstly, we re-sized images to have consistent size of 60 by 40 pixels, and reshaped each images into a 2400 x 1 column vector (where 2400 = 60 x 40 pixels). Then, a total of 38 x 64 = 2432 cropped images, which were transformed to column vectors, now form a 2400 x 2432 matrix.

We then formed another matrix with uncropped images in a similar manner. We re-sized images to 60 by 40 pixels and reshaped each images into a 2400 x 1 column vector as well. In this case, total number of uncropped images is 165. These images were then transformed to a column vectors and now form a 2400 x 165 matrix.

4 Computational Results

By theory, U matrix which is equivalent to a set of eigenvectors, dominates a feature space. In other words, each columns of U are called eigenfaces. A diagonal terms of a diagonal matrix Σ represents the energy contribution of each corresponding eigenface and it’s called singular values (SV), as mentioned in the section above. It also represents the

square root of the eigenvalues. The matrix V tells us how each image itself projects onto the eigenvector space, in other words, how much each image contributes to its eigenface-space. This represent the unique features of each image thus enable us to distinguish pictures eventually.

4.1 Cropped Images

As we do SVD analysis on the image matrix from cropped images, we got 2400 x 2400 U matrix, 2400 x 2432 Σ matrix and a 2432 x 2432 matrix V^*.

First, to get an idea of the energy distribution of the eigenfaces from SVD, we plotted the singular value spectrum for all eigenvalues as shown in figure 1. The percentage of energy of with the highest SV value was only about 14 percent, which is really small.

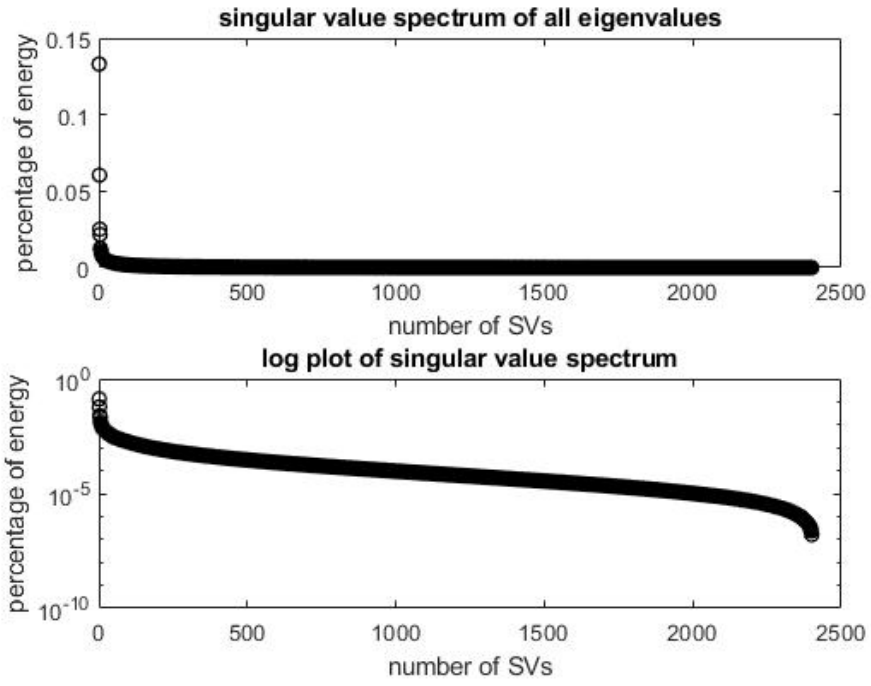


Fig. 1: singular values on a standard plot and a log plot of singular values

Without further ado, we realized that continuing this method of SVD is not useful. From computation, the percentage of energy was just over 95 percent after summing up to 1000 SVs. (Further calculation is provided in the appendix)

For different approach, we did SVD on the covariance matrix of image matrix, say X. We used XX^T as a covariance matrix for simplification. Also, as shown in the previous section, U diagonalizes XX^T so eigenface space would still be U after doing SVD on X. Figure 2 shows the plot of singular values for all SVs and figure 3 shows the plot of singular values for the first 10 largest SVs.

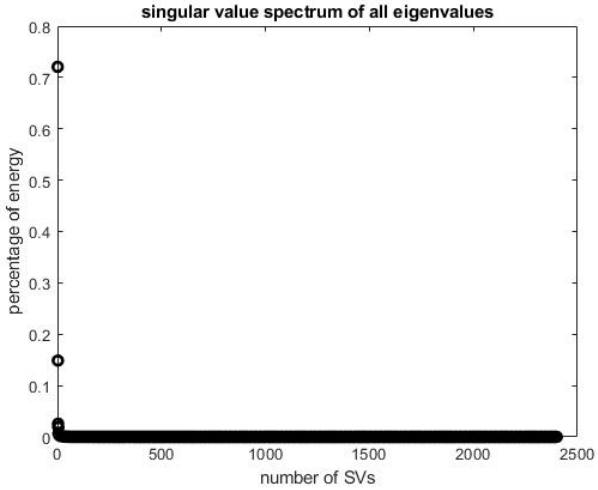


Fig. 2: singular values on a standard plot for covariance matrix

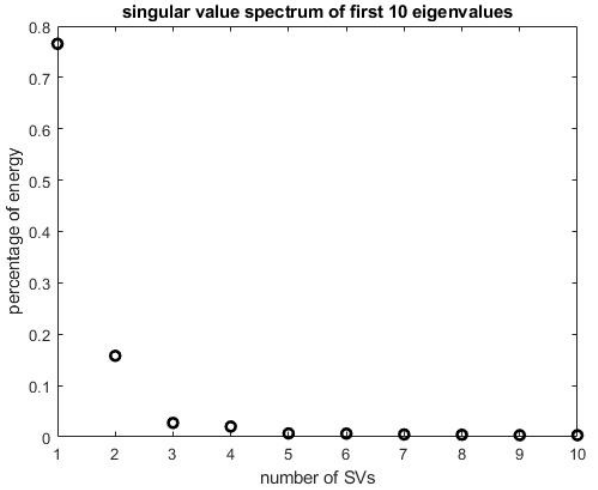


Fig. 3: singular values of first 10 SVs on a standard plot for covariance matrix

As shown in the figures 2 and 3, by using covariance matrix, the redundant data were removed and the figure looked much more efficient than when we had simply done SVD on the image matrix. The mode of data of 4 seem appropriate from figure 3. Also, from computation, total percentage of energy up to 4 SVs are about 91 percent.

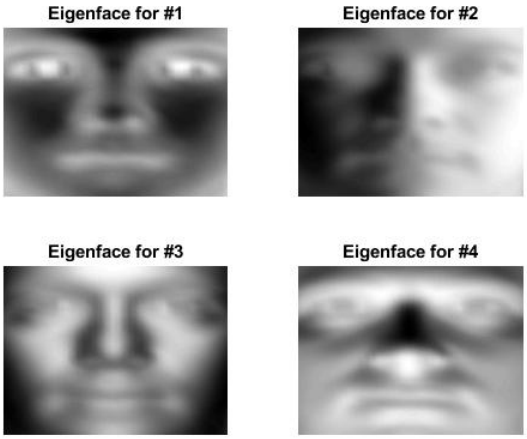


Fig. 4: first 4 eigenfaces

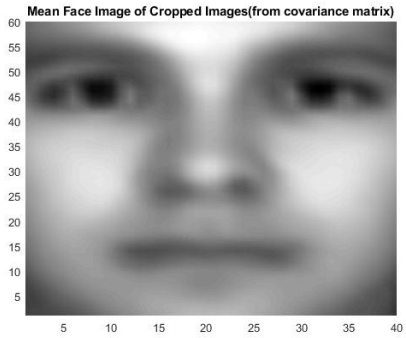
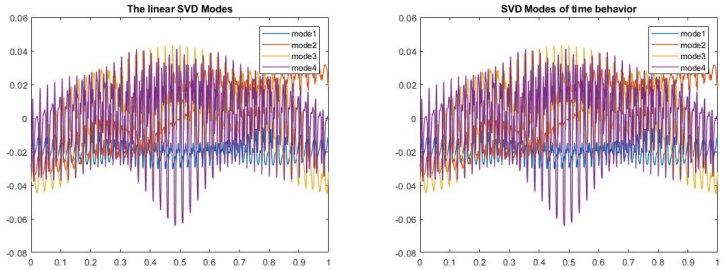


Fig. 5: mean face of all eigenfaces

Figure 4 is the image of first four eigenfaces. This image seems fair as all eigenfaces have required facial features. It's interesting to see that the first eigenface looks rather "abstract" than the database images of cropped faces. Moreover, its mean face of all eigenfaces, from figure 5 is very clear with clear facial features.



A linear SVD modes SVD modes of time

Fig. 7: modes plot of covariance matrix for cropped images

If you look at figure 7 carefully, the blue color graph of mode 1 is most flat, which makes sense because it's the mode with the highest probability of energy. Moreover, the mode plots of U and V are the same because the V part of SVD on covariance matrix, XX^T, is equivalent to U^* and since U is unitary, the SVD mod plot of U and V are identical in this case. ($AA^T = (U\Sigma V^*)(U\Sigma V^*)^T = U\Sigma^2 U^*$ was proved in the previous section)

4.2 Uncropped Images

Now, let's do the analyses on the uncropped images. As described before, there are more variations among uncropped images.

As we did for cropped images, we did SVD on the data matrix of uncropped images.

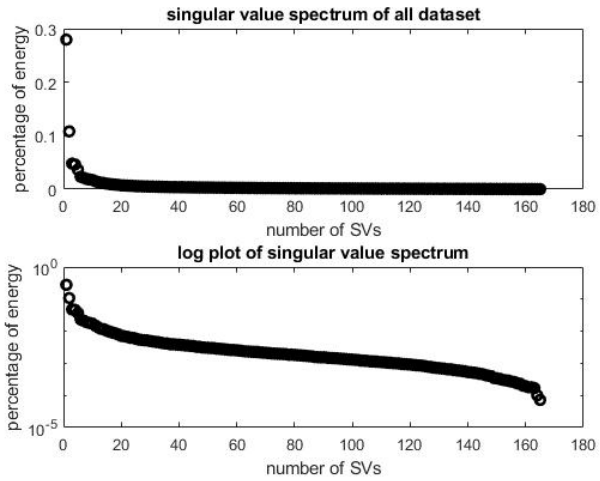


Fig. 8: singular value spectrum of uncropped images

The singular value plots from figure 8tells us that the SVD on the data matrix of uncropped images is inefficient for the same reason as cropped image data matrix. Figure 8 shows that the percentage of energy with the highest SV value is only about 29 percent. And also from computation, total of about 100 SVs are required to reach the sum of over 95 percentage of energy.

Hence, we used a covariance matrix for the same reason.

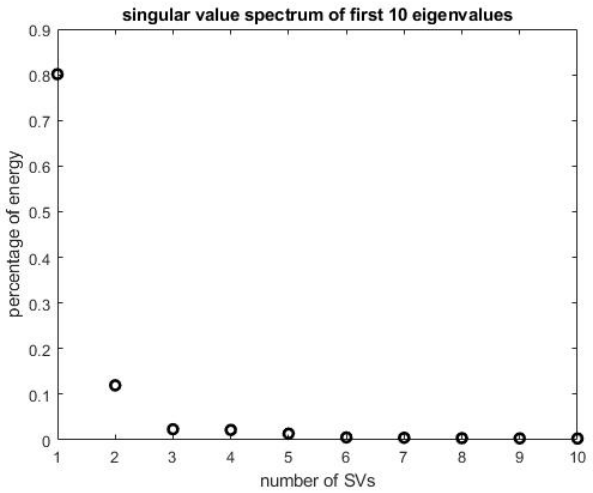


Fig. 9: singular value spectrum of uncropped images for covariance matrix

The figure 9 shows that the percentage of energy with the highest SV is about 80 percent and the number of modes necessary for good image construction is about 5. The total percentage of energy of 5 SVs is about 96 percent from computation (code for computation is shown in appendix)

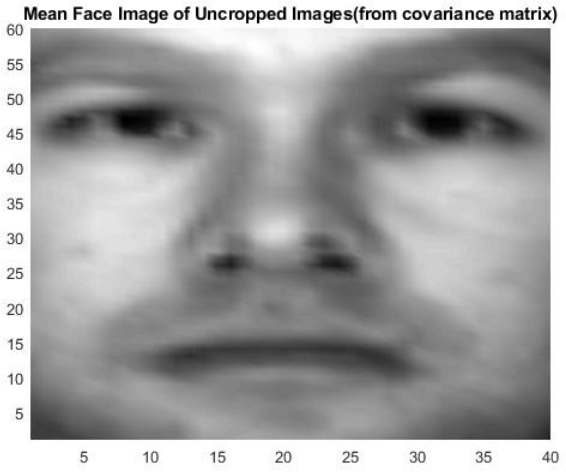


Fig. 10: mean face of all eigenfaces

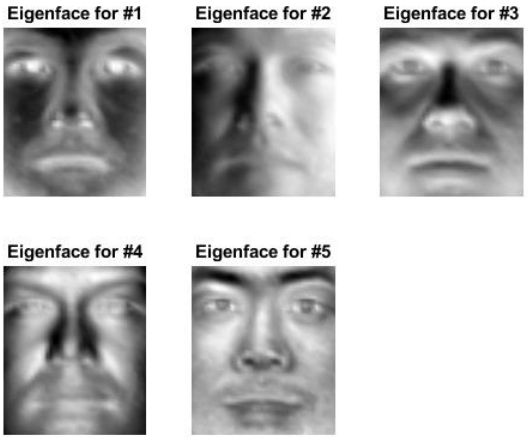


Fig. 11: first five eigenfaces

Figure 10 shows the mean face of all eigenfaces and figure 11 shows the first five eigenfaces from uncropped images. Its eigenfaces and mean face reflect clear facial features and they look very similar to cropped images.

5 Summary and Conclusions

The result was really fascinating, especially the result of mean faces and eigenfaces of uncropped images. Since the database images of uncropped images include individuals with background and with glasses, so we expected eigenfaces to reflect such variations. In other words, we expected eigenfaces of uncropped images would be quite different from eigenfaces of cropped images. However, the result was different from what we had expected. The eigenfaces of uncropped image was enlarged from its original images and only focuses on important facial features such as eyes, nose and mouth, just like cropped images. By simply comparing analyses of cropped and uncropped images, it would be difficult for us to differentiate them.

Further analyses can be done on images to do further applications of images such as classifications based on our analyses or also do image compression. We believe the theories and applications we had implemented on this paper had established our knowledge in basic machine learning which would be explored throughout the quarter.

A MATLAB functions used and brief implementation explanation

The following MATLAB code shows how each computations for cropped images and uncropped images are done. No function was used in the following code. The hardest part of coding was to obtain data images from files, especially cropped images where we had to obtain images from sub-files altogether. The following code would generate desired images including figures attached in the previous sections.

B MATLAB codes

B.1 Code Used for Cropped Images

```
% Start with a folder and get a list of all subfolders.
% Finds and prints names of all images in
% that folder and all of its subfolders.
clc;    % Clear the command window.
workspace;  % Make sure the workspace panel is showing.
format long g;
format compact;
% cd('\\tsclient\home\Downloads\Winter2018\AMATH582\HW1')

% Define a starting folder.
start_path = fullfile(matlabroot, '..\HW1\CroppedYale');
% Ask user to confirm or change.
topLevelFolder = uigetdir(start_path);
if topLevelFolder == 0
        return;
end
% Get list of all subfolders.
allSubFolders = genpath(topLevelFolder);
% Parse into a cell array.
remain = allSubFolders;
listOfFolderNames = {};
while true
        [singleSubFolder, remain] = strtok(remain, ';');
        if isempty(singleSubFolder)
                break;
        end
        listOfFolderNames = [listOfFolderNames singleSubFolder];
end
numberOfFolders = length(listOfFolderNames)

% Process all image files in those folders.
% List of image matrices
listOfImMat = {}
for k = 1 : numberOfFolders
        % Get this folder and print it out.
        thisFolder = listOfFolderNames{k};
        fprintf('Processing folder %s\n', thisFolder);

        % Get PMG files.
        filePattern = sprintf('%s/*.pgm', thisFolder);
        baseFileNames = dir(filePattern);
```

```
        numberOfImageFiles = length(baseFileNames);
        % Now we have a list of all files in this folder.

        if numberOfImageFiles >= 1
                % Go through all those image files.
                for f = 1 : numberOfImageFiles
                        fullFileName = fullfile(thisFolder, baseFileNames(f).
            currentimage = imresize(double(imread(fullFileName)), [60,40]);
            listOfImMat = [listOfImMat currentimage];
                        fprintf('     Processing image file %s\n', fullFileNar
                end
        else
                fprintf('     Folder %s has no image files in it.\n', thisFol
        end
end

%%
% Get data matrix
Data = [];
for j = 1: length(listOfImMat)
    Data = [Data reshape(listOfImMat{j}, 60*40, 1)];
end

%%
% svd with raw data matrix
[u,s,v] = svd(Data);

figure (1)
% singular value spectrum
subplot(2,1,1), plot(diag(s)/sum(diag(s)),'ko', 'Linewidth', [1])
title('singular value spectrum of all eigenvalues')
xlabel('number of SVs');
ylabel('percentage of energy');
subplot(2,1,2), semilogy(diag(s)/sum(diag(s)),'ko', 'Linewidth', [1])
title('log plot of singular value spectrum')
xlabel('number of SVs')
ylabel('percentage of energy');

figure(2)
trc = s(1:1000,1:1000);
semilogy(diag(trc)/sum(diag(s)),'ko', 'Linewidth', [1]);
title('log plot of singular value spectrum for first 1000 SVs')
xlabel('number of SVs')
ylabel('percentage of energy');
```

```
figure(3)
% compute eigenfaces
for i = 1 : 9
    eigenface = reshape(u(:,i),60,40);
    subplot(3,3,i), pcolor(flipud(eigenface)),shading interp, colormap(gray),set
    title(['Eigenface for #' num2str(i)]);
end

figure (4)
% Compute mean face image of data images
mean_face = mean(Data,2);
pcolor(flipud(reshape(mean_face,60,40))),shading interp, colormap(gray)
title('Mean Face Image of Cropped Images');

% POD modes
x=linspace(0,1,2400);
t=linspace(0,1,2432);
figure (5)
plot(x,u(:,1:4),'Linewidth', [1]);
title('The linear SVD Modes');
legend('mode1','mode2','mode3','mode4');

figure (6)
plot(t,v(:,1:4),'Linewidth', [1]);
title('SVD Modes of time behavior');
legend('mode1','mode2','mode3','mode4');

sig=diag(s);
energy1 = sig(1)/sum(sig);
energy1000 = sum(sig(1:1000))/sum(sig);

%%
% 2. usv on covariance matrix
C = Data*(Data.');
[u1,s1,v1] = svd(C);

figure(1)
plot(diag(s1)/sum(diag(s1)),'ko', 'Linewidth', [2])
title('singular value spectrum of all eigenvalues')
xlabel('number of SVs');
ylabel('percentage of energy');
figure(2)
trc1 = s1(1:10,1:10);
```

```
plot(diag(trc1)/sum(diag(trc1)),'ko', 'Linewidth', [2])
title('singular value spectrum of first 10 eigenvalues')
xlabel('number of SVs')
ylabel('percentage of energy');

figure(3)
% compute eigenfaces
for i = 1 : 4
    eigenface = reshape(u1(:,i),60,40);
    subplot(2,2,i), pcolor(flipud(eigenface)),shading interp, colormap(gray),
    title(['Eigenface for #' num2str(i)]);
end

figure(4)
% Compute mean face image of data images
mean_face1 = mean(C,2);
pcolor(flipud(reshape(mean_face1,60,40))),shading interp, colormap(gray)
title('Mean Face Image of Cropped Images(from covariance matrix)');

% POD modes
x=linspace(0,1,2400);
t=linspace(0,1,2400);
figure(5)
plot(x,u1(:,1:4),'Linewidth', [1]);
title('The linear SVD Modes');
legend('mode1','mode2','mode3','mode4');

figure(6)
plot(t,v1(:,1:4),'Linewidth', [1]);
title('SVD Modes of time behavior');
legend('mode1','mode2','mode3','mode4');

sig1=diag(s1);
energy1 = sig1(1)/sum(sig1)
energy2 = sum(sig1(1:2))/sum(sig1)
energy3 = sum(sig1(1:3))/sum(sig1)
energy4 = sum(sig1(1:4))/sum(sig1)
```

B.2 Code Used for Uncropped Images

```
% Start with a folder and get a list of all subfolders.
% Finds and prints names of all images in
% that folder and all of its subfolders.
clc;    % Clear the command window.
workspace;  % Make sure the workspace panel is showing.
```

```
format long g;
format compact;
% cd('\\tsclient\home\Downloads\Winter2018\AMATH582\HW1')

Directory = '..\HW1\UncroppedYale\yalefaces';
% Read images from Images folder
Imgs = dir(Directory);
listOfImMat2 = {};
for j=1:length(Imgs)
    thisname = Imgs(j).name;
    thisfile = fullfile(Directory, thisname);
    try
      Img = imread(thisfile);  % try to read image
      currentimage = imresize(double(Img), [60,40]);
      listOfImMat2 = [listOfImMat2 currentimage];
      fprintf('     Processing image file %s\n', thisfile);
   catch
   end
end

%%
% Data Matrix
uData = [];
for j = 1: length(listOfImMat2)
    uData = [uData reshape(listOfImMat{j}, 60*40, 1)];
end

%%
% 1. raw data
% Get Eigenfaces by covariance matrix
[u3,s3,v3] = svd(uData);
% features = evectors' * shiftedData;

figure (1)
% singular value spectrum
subplot(2,1,1), plot(diag(s3)/sum(diag(s3)),'ko', 'Linewidth', [2]);
title('singular value spectrum of all dataset')
xlabel('number of SVs');
ylabel('percentage of energy');
subplot(2,1,2), semilogy(diag(s3)/sum(diag(s3)),'ko', 'Linewidth', [2]);
title('log plot of singular value spectrum')
xlabel('number of SVs');
ylabel('percentage of energy');
```

```
figure(2)
trc3 = s3(1:100,1:100);
semilogy(diag(trc3)/sum(diag(s3)),'ko', 'Linewidth', [1]);
title('log plot of singular value spectrum for first 100 SVs')
xlabel('number of SVs')
ylabel('percentage of energy');

figure (3)
% Compute mean face image of data images
mean_face3 = mean(uData,2);
pcolor(flipud(reshape(mean_face3,60,40))),shading interp, colormap(gray);
title('Mean Face Image of Uncropped Images');

figure(4)
% compute eigenfaces
for i = 1 : 9
    eigenface = reshape(u3(:,i),60,40);
    subplot(3,3,i), pcolor(flipud(eigenface)),shading interp, colormap(gray),
    title(['Eigenface for #' num2str(i)]);
end

% POD modes
x=linspace(0,1,2400);
t=linspace(0,1,165);
figure (5)
plot(x,u3(:,1:4),'Linewidth', [1]);
title('The linear SVD Modes');
legend('mode1','mode2','mode3','mode4');

figure (6)
plot(t,v3(:,1:4),'Linewidth', [1]);
title('SVD Modes of time behavior');
legend('mode1','mode2','mode3','mode4');

sig3 = diag(s3)
energy1 = sig3(1)/sum(sig3);
energy100 = sum(sig3(1:100))/sum(sig3);

%%
% 2. usv on covariance matrix
C4 = uData*(uData.');
[u4,s4,v4] = svd(C4);

figure(1)
```

```
plot(diag(s4)/sum(diag(s4)),'ko', 'Linewidth', [2])
title('singular value spectrum of all eigenvalues')
xlabel('number of SVs');
ylabel('percentage of energy');

figure(2)
trc4 = s4(1:10,1:10);
plot(diag(trc4)/sum(diag(trc4)),'ko', 'Linewidth', [2])
title('singular value spectrum of first 10 eigenvalues')
xlabel('number of SVs')
ylabel('percentage of energy');

figure(3)
% compute eigenfaces
for i = 1 : 5
    eigenface = reshape(u4(:,i),60,40);
    subplot(2,3,i), pcolor(flipud(eigenface)),shading interp, colormap(gray),set
    title(['Eigenface for #' num2str(i)]);
end

figure(4)
% Compute mean face image of data images
mean_face4 = mean(C4,2);
pcolor(flipud(reshape(mean_face4,60,40))),shading interp, colormap(gray)
title('Mean Face Image of Uncropped Images(from covariance matrix)');

% POD modes
x=linspace(0,1,2400);
t=linspace(0,1,2400);
figure(5)
plot(x,u4(:,1:5),'Linewidth', [1]);
title('The linear SVD Modes');
legend('mode1','mode2','mode3','mode4','mode5');

figure(6)
plot(t,v4(:,1:5),'Linewidth', [1]);
title('SVD Modes of time behavior');
legend('mode1','mode2','mode3','mode4','mode5');

sig4=diag(s4);
energy1 = sig4(1)/sum(sig4)
energy2 = sum(sig4(1:2))/sum(sig4)
energy3 = sum(sig4(1:3))/sum(sig4)
energy4 = sum(sig4(1:4))/sum(sig4)
```

18

```
energy5 = sum(sig4(1:5))/sum(sig4)
```

References

AMATH 582 Textbook. J. Nathan Kutz. 2013. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data. Oxford University Press, Inc., New York, NY, USA.