Principal component analysis - a tutorial

Article · January 2016

DOI: 10.1504/IJAPR.2016.079733

1 author:

Alaa Tharwat
Frankfurt University of Applied Sciences
81 PUBLICATIONS **638** CITATIONS

SEE PROFILE

Principal Component Analysis - A Tutorial

Alaa Tharwat

Electrical Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt

E-mail: emgalaatharwat@hotmail.com

Abstract: Dimensionality reduction is one of the preprocessing steps in many machine learning applications and it is used to transform the features into a lower dimension space. *Principal Component Analysis* (PCA) technique is one of the most famous unsupervised dimensionality reduction techniques. The goal of the PCA is to find the space, which represents the direction of the maximum variance of the given data. This paper highlights the basic background needed to understand and implement the PCA technique. This paper starts with basic definitions of the PCA technique and the algorithms of two methods of calculating PCA, namely, the covariance matrix and Singular Value Decomposition (SVD) methods. Moreover, a number of numerical examples are illustrated to show how the PCA space is calculated in easy steps. Three experiments are conducted to show how to apply PCA in the real applications including biometrics, image compression, and visualization of high-dimensional datasets.

Keywords: Principal Component Analysis (PCA); Dimensionality Reduction; Feature Extraction; Covariance Matrix; Singular Value Decomposition (SVD); PCA Space; Biometrics; Image Compression.

Biographical notes: Alaa Tharwat received his BSc in 2002 and MSc in 2008, from Faculty of Engineering, Computer and Control Systems Department, Mansoura University, Egypt. He is an Assistant Lecturer at Electrical Department, Faculty of Engineering, Suez Canal University, Egypt. He was a researcher at Gent University, within the framework of the Welcome project - Erasmus Mundus Action 2 - with a title "Novel approach of multi-modal biometrics for animal identification". He is an author of many research studies published at national and international journals, conference proceedings. His major research interests include pattern recognition, machine learning, digital image processing, biometric authentication, and bio-inspired optimization

1 Introduction

Dimensionality reduction techniques are important in many applications related to data mining (Tharwat et al., 2012; Bramer, 2013; Larose, 2014), Bioinformatics (Saeys et al., 2007), information retrieval (Venna et al., 2010), machine learning (Duda et al., 2012), and chemistry (Chiang et al., 2000). The main goal of the dimensionality reduction techniques is to transform the data or features from a higher dimensional space to a lower dimensional space. There are two major approaches of the dimensionality reduction techniques, namely, *unsupervised* and *supervised* approaches (Tenenbaum et al., 2000; Kirby, 2000; Duda et al., 2012).

In the supervised approach, the class labels are used to find the lower dimensional space. Supervised approaches have been used in many applications such as Biometrics (Lu et al., 2003; Cui, 2012) and Bioinformatics (Wu et al., 2009). The supervised approach has many techniques such as Mixture Discriminant Analysis (MDA) (Hastie and Tibshirani, 1996), Neural Networks (NN) (Hinton and Salakhutdinov, 2006), and Linear Discriminant Analysis (LDA) (Scholkopft and Mullert, 1999). In the unsupervised approach, the lower dimensional space is found without using the class labels and it is suitable for more applications such as visualization (Müller and Schumann, 2006; Barshan et al., 2011), data reduction (Kambhatla and Leen, 1997; Jolliffe, 2002), and noise removal (Thomas et al., 2006). There are many unsupervised dimensionality reduction techniques such as Independent Component Analysis (ICA) (Hyvärinen et al., 2004), Locally Linear Embedding (LLE) (Roweis and Saul, 2000), and Principal Component Analysis (PCA) (Dash et al., 1997; Belkin and Niyogi, 2003; Tharwat et al., 2015; Gaber et al., 2015), which is the most common dimensionality reduction technique.

PCA technique has many goals including finding relationships between observations, extracting the most important information from the data, outlier detection and removal, and reducing the dimension of the data by keeping only the important information. All these goals are achieved by finding the PCA space, which represents the direction of the maximum variance of the given data (Turk and Pentland, 1991). The PCA space consists of orthogonal principal components, i.e. axes or vectors. The principal components (PCs) are calculated by solving the *covariance matrix* or using *Singular Value Decomposition* (SVD).

This paper gives a detailed tutorial about the PCA technique and it is divided into four sections. In Section 2, a clear definition of the basic idea of the PCA and its background are highlighted. This section begins by explaining how to calculate the PCs using the covariance matrix and SVD methods, how to construct the PCA space from the calculated PCs, projecting the data on the PCA space, and reconstruct the original data again from the PCA space. Moreover, the steps of calculating the PCS, PCA space, and projecting the data to reduce its dimension are summarized and visualized in detail. Section 3 illustrates numerical examples to show how to calculate the PCA space and how to select the most robust eigenvectors to build the PCA space. Moreover, the PCs are calculated using the two methods, i.e. covariance matrix and SVD. In Section 4, three experiments are conducted to show: (1) How the PCA technique is used in the real applications such as biometrics, image

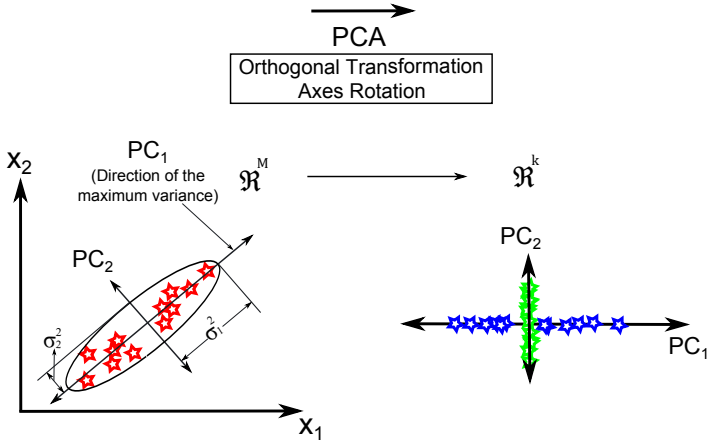


Figure 1: Example of the two-dimensional data (x_1, x_2). The original data are on the left with the original coordinate, i.e. x_1 and x_2, the variance of each variable is graphically represented and the direction of the maximum variance, i.e. the principal component PC_1, is shown; on the right the original data are projected on the first (blue stars) and second (green stars) principal components.

compression, and visualization; (2) The influence of the number of the selected eigenvectors on the amount of the preserved data. Finally, concluding remarks will be given in Section 5.

2 Principal Component Analysis (PCA)

2.1 Definition of PCA

The goal of the PCA technique is to find a lower dimensional space or PCA space (W) that is used to transform the data $(X = \{x_1, x_2, \ldots, x_N\})$ from a higher dimensional space $(\mathcal{R}^M)$ to a lower dimensional space $(\mathcal{R}^k)$, where N represents the total number of samples or observations and x_i represents i^{th} sample, pattern, or observation. All samples have the same dimension $(x_i \in \mathcal{R}^M)$. In other words, each sample is represented by M variables, i.e. each sample is represented as a point in M-dimensional space (Wold et al., 1987). The direction of the PCA space represents the direction of the maximum variance of the given data as shown in Figure 1. As shown in the figure, the PCA space is consists of a number of PCs. Each principal component has a different robustness according to the amount of variance in its direction.

2.2 Principal Components (PCs)

The PCA space consists of k principal components. The principal components are orthonormal[a], uncorrelated[b], and it represents the direction of the maximum variance.

The first principal component ((PC_1 or v_1) $\in \mathcal{R}^{M\times 1}$) of the PCA space represents the direction of the maximum variance of the data, the second principal component has the second largest variance, and so on. Figure 1 shows how the original data are transformed from the original space ($\mathcal{R}^M$) to the PCA space ($\mathcal{R}^k$). Thus, the PCA technique is considered an orthogonal transformation due to its orthogonal principal components or axes rotation due to the rotation of the original axes (Wold et al., 1987; Shlens, 2014). There are two methods to calculate the principal components. The first method depends on calculating the covariance matrix, while, the second one uses the SVD method.

2.3 Covariance Matrix Method

In this method, there are two main steps to calculate the PCs of the PCA space. First, the covariance matrix of the data matrix (X) is calculated. Second, the eigenvalues and eigenvectors of the covariance matrix are calculated. Figure 2 illustrates the visualized steps of calculating the PCs using the covariance matrix method.

2.3.1 Calculating Covariance Matrix (Σ):

The variance of any variable measures the deviation of that variable from its mean value and it is defined as follows, $\sigma^2(x) = Var(x) = E((x-\mu)^2) = E\{x^2\} - (E\{x\})^2$, where μ represents the mean of the variable x, and $E(x)$ represents the expected value of x. The covariance matrix is used when the number of variables more than one and it is defined as follows, $\Sigma_{ij} = E\{x_i x_j\} - E\{x_i\}E\{x_j\} = E[(x_i-\mu_i)(x_j-\mu_j)]$. As shown in Figure 2, step(**A**), after calculating the mean of each variable in the data matrix, the mean-centring data are calculated by subtracting the mean ($\mu \in \mathcal{R}^{(M\times 1)}$) from each sample as follows, $D = \{d_1, d_2, \ldots, d_N\} = \{x_1 - \mu, x_2 - \mu, \ldots, x_N - \mu\}$ (Turk and Pentland, 1991; Bishop, 2006; Shlens, 2014). The covariance matrix is then calculated as follows, $\Sigma = DD^T$ (see Figure 2, step (**B**)).

Covariance matrix is a symmetric matrix (i.e. $X = X^T$) and always positive semi-definite matrix [c]. The diagonal values of the covariance matrix represent the variance of the variable

[a] Orthonormal vectors have a unit length and orthogonal as follows, $v_i^T v_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$

[b] v_i and v_j are uncorrelated if $Cov(v_i, v_j) = 0,\ i \neq j$, where $Cov(v_i, v_j)$ represents the covariance between the i^{th} and j^{th} vectors.

[c] X is positive semi-definite if $v^T X v \geq 0$ for all $v \neq 0$. In other words, all eigenvalues of X are ≥ 0.

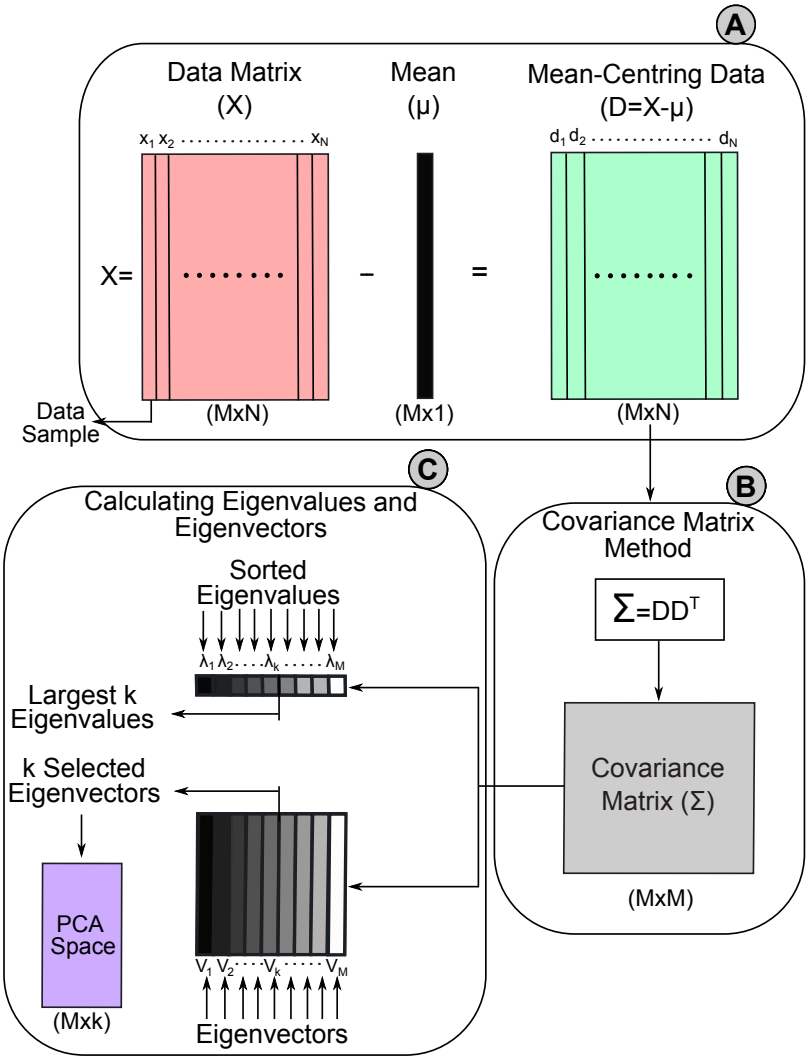


Figure 2: Visualized steps to calculate the PCA space using the covariance matrix method.

$x_i, i = 1, \ldots, M$, while the off-diagonal entries represent the covariance between two different variables as shown in Equation (1). A positive value in covariance matrix means a positive correlation between the two variables, while the negative value indicates a negative correlation and zero value indicate that the two variables are uncorrelated or statistically independent (Shlens, 2014).

$$\begin{pmatrix} Var(x_1, x_1) & Cov(x_1, x_2) & \ldots & Cov(x_1, x_M) \\ Cov(x_2, x_1) & Var(x_2, x_2) & \ldots & Cov(x_2, x_M) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(x_M, x_1) & Cov(x_M, x_2) & & Var(x_M, x_M) \end{pmatrix} \tag{1}$$

2.3.2 Calculating Eigenvalues (λ) and Eigenvectors (V):

The covariance matrix is solved by calculating the eigenvalues (λ) and eigenvectors (V) as follows:

$$V\Sigma = \lambda V \tag{2}$$

where V and λ represent the eigenvectors and eigenvalues of the covariance matrix, respectively.

The eigenvalues are scalar values, while the eigenvectors are non-zero vectors, which represent the principal components, i.e. each eigenvector represents one principal component. The eigenvectors represent the directions of the PCA space, and the corresponding eigenvalues represent the scaling factor, length, magnitude, or the robustness of the eigenvectors (Hyvärinen, 1970; Strang and Aarikka, 1986). The eigenvector with the highest eigenvalue represents the first principal component and it has the maximum variance as shown in Figure 1 (Hyvärinen, 1970). The eigenvalues may be equal when the PCs have equal variances and hence all the eigenvectors are the same and we cannot decide which eigenvectors are used to construct the PCA space.

2.4 Singular Value Decomposition (SVD) Method

In this method, the principal components are calculated using SVD method. A visualized steps of calculating the principal components using SVD method are illustrated in Figure 3.

2.4.1 Calculating SVD:

Singular value decomposition is one of the most important linear algebra principles. The aim of the SVD method is to diagonalize the data matrix ($X \in \mathcal{R}^{p\times q}$) into three matrices as in Equation (3).

$$X = LSR^T = \begin{bmatrix} \\ l_1 \cdots l_p \\ \\ \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & s_q \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -r_1^T- \\ -r_2^T- \\ \vdots \\ -r_q^T- \end{bmatrix} \tag{3}$$

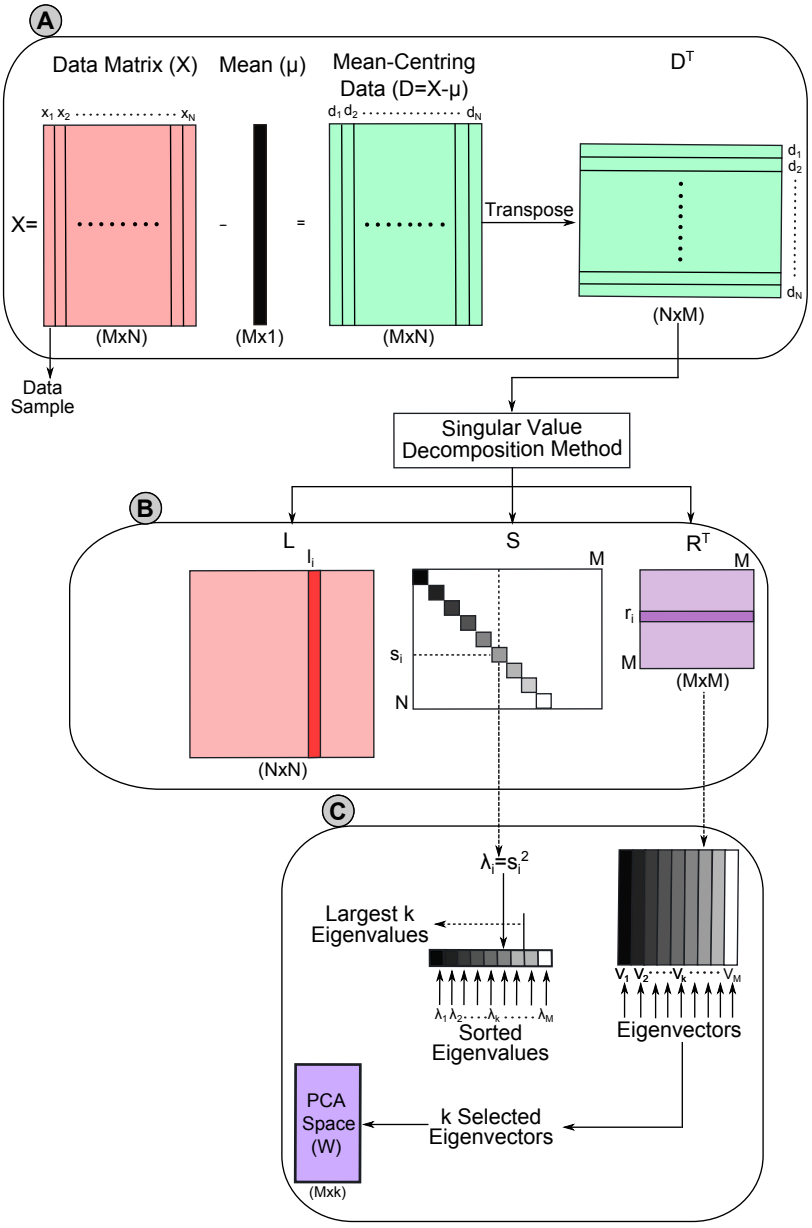


Figure 3: Visualized steps to calculate the PCA space using SVD method.

where $L(p \times p)$ are called left singular vectors, $S(p \times q)$ is a diagonal matrix represents the singular values that are sorted from high-to-low, i.e. the highest singular value in the upper-left index of S, thus, $s_1 \geq s_2 \geq \cdots \geq s_q \geq 0$, and $R(q \times q)$ represents the right singular vectors. The left and right singular matrices, i.e. L and R, are orthonormal bases. To calculate SVD, R^T and S are first calculated by diagonalizing $X^T X$ as follows, $X^T X = (LSR^T)^T(LSR^T) = RS^T L^T LSR^T = RS^2 R^T$, where $L^T L = I$. The

left singular vectors (L) is then calculated as follows, $L = XRS^{-1}$, where Xr_i is in the direction of $s_i l_i$ (Alter et al., 2000; Greenberg, 2001; Wall et al., 2003). The columns of the right singular vectors (R) represent the eigenvectors of $X^T X$ or the principal components of the PCA space, and $s_i^2,\ \forall\ i = 1, 2, \ldots, q$ represent their corresponding eigenvalues as shown in Figure 3, steps (**B & C**). Since, the number of principal components and their eigenvalues are equal to q, thus the dimension of our original data matrix must be reversed to be compatible with SVD method. In other words, the mean-centring matrix is transposed before calculating the SVD method and hence each sample is represented by one row as shown in Figure 3, step (**A**).

2.4.2 SVD vs. Covariance Matrix Methods

To compute the PCA space, the eigenvalues and eigenvectors of the covariance matrix are calculated, where the covariance matrix is the product of DD^T, where $D = \{d_i\}_{i=1}^N, d_i = x_i - \mu$. Using Equation (3) that is used to calculate SVD, the covariance matrix can be calculated as follows:

$$DD^T = (LSR^T)^T(LSR^T) = RS^T L^T LSR^T \tag{4}$$

where $L^T L = I$

$$DD^T = RS^2 R^T = (SVD(D^T))^2 \tag{5}$$

where S^2 represents the eigenvalues of $D^T D$ or DD^T and the columns of the right singular vector (R) represent the eigenvectors of DD^T. To conclude, the square root of the eigenvalues that are calculated using the covariance matrix method are equal to the singular values of SVD method. Moreover, the eigenvectors of Σ are equal to the columns of R. Thus, the eigenvalues and eigenvectors that are calculated using the two methods are equal.

2.5 PCA Space (Lower Dimensional Space)

To construct the lower dimensional space of PCA (W), a linear combination of k selected PCs that have the most k eigenvalues are used to preserve the maximum amount of variance, i.e. preserve the original data, while the other eigenvectors or PCs are neglected as shown in Figures 2(step **C**) and 3(step **C**). The lower dimensional space is denoted by $W = \{v_1, \ldots, v_k\}$. The dimension of the original data is reduced by projecting it after subtracting the mean onto the PCA space as in Equation (6).

$$Y = W^T D = \sum_{i=1}^{N} W^T (x_i - \mu) \tag{6}$$

where $Y \in \mathcal{R}^k$ represents the original data after projecting it onto the PCA space as shown in Figure 4, thus $(M - k)$ features or variables are lost from the original data.

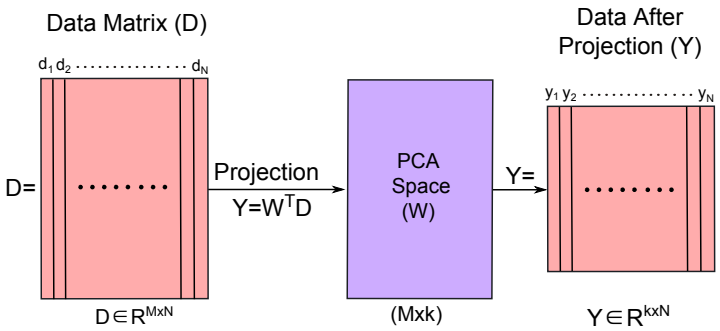


Figure 4: Data projection in PCA as in Equation (6).

2.6 Data Reconstruction

The original data can be reconstructed again as in Equation (7).

$$\hat{X} = WY + \mu = \sum_{i=1}^{N} Wy_i + \mu \tag{7}$$

where $\hat{X}$ represents the reconstructed data. The deviation between the original data and the reconstructed data are called the reconstruction error or residuals as denoted in Equation (8). The reconstruction error represents the square distance between the original data and the reconstructed data, and it is inversely proportional to the total variance of the PCA space. In other words, selecting a large number of PCs, increases the total variance of W and decreases the error between the reconstructed and the original data. Hence, the robustness of the PCA is controlled by the number of selected eigenvectors (k) and it is measured by the sum of the selected eigenvalues, which is called total variance as in Equation (9). For example, the robustness of the lower dimensional space $W = \{v_1 \ldots, v_k\}$ is measured by the ratio between the total variance ($\lambda_i, i = 1, \ldots, k$) of W to the total variance (Abdi and Williams, 2010).

$$Error = X - \hat{X} = \sum_{i=1}^{N} (x_i - \hat{x_i})^2 \tag{8}$$

$$\text{Robustness of the PCA space} = \frac{\text{Total Variance of } W}{\text{Total Variance}} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{M} \lambda_i} \tag{9}$$

Table 1 Notation.

Notation	Description	Notation	Description
X	Data matrix	x_i	i^{th} sample
N	Total number of samples in X	M	Dimension of X or the number of features of X
W	Lower dimensional or PCA space	PC_i	i^{th} principal component
μ	Mean of all samples (Total or global mean)	Σ	Covariance matrix
λ	Eigenvalues of Σ	V	Eigenvectors of Σ
v_i	The i^{th} eigenvector	λ_i	The i^{th} eigenvalue
Y	Projected data	L	Left singular vectors
S	Singular values	R	Right singular vectors
$\hat{X}$	Reconstructed data	k	The dimension of W
D	Mean-Centring data (Data-mean)	ω_i	i^{th} Class

2.7 PCA Algorithms

The first step in the PCA algorithm is to construct a data or feature matrix (X), where each sample is represented as one column and the number of rows represents the dimension, i.e. the number of features, of each sample. The detailed steps of calculating the lower dimensional space of the PCA technique using the covariance and SVD methods are summarized in Algorithm (1) and Algorithm (2), respectively. MATLAB codes for the two methods are illustrated in Appendix A.3.

3 Numerical Examples

In this section, two numerical examples were illustrated to calculate the lower dimensional space. In the first example, the samples were represented by only two features to visualize it and the PCs were calculated using the two methods, i.e. covariance matrix and SVD methods. Moreover, in this example, we show how the eigenvalues and eigenvectors that were calculated using the two methods were equal. Furthermore, the example explains how the data were projected and reconstructed again. In the second example, each sample was represented by four features to show how the steps of PCA were affected by changing the dimension. Moreover, in this example, the influences of a constant variable, i.e. zero variance, were explained. MATLAB codes for all experiments are introduced in Appendix A.1.

Algorithm 1 : Calculating PCs using Covariance Matrix Method.

1: Given a data matrix ($X = [x_1, x_2, \ldots, x_N]$), where N represents the total number of samples and x_i represents the i^{th} sample.
2: Compute the mean of all samples as follows:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{10}$$

3: Subtract the mean from all samples as follows:

$$D = \{d_1, d_2, \ldots, d_N\} = \sum_{i=1}^{N} x_i - \mu \tag{11}$$

4: Compute the covariance matrix as follows:

$$\Sigma = \frac{1}{N-1} D \times D^T \tag{12}$$

5: Compute the eigenvectors V and eigenvalues λ of the covariance matrix (Σ).
6: Sort eigenvectors according to their corresponding eigenvalues.
7: Select the eigenvectors that have the largest eigenvalues $W = \{v_1, \ldots, v_k\}$. The selected eigenvectors (W) represent the projection space of PCA.
8: All samples are projected on the lower dimensional space of PCA (W) as follows, $Y = W^T D$.

Algorithm 2 : Calculating PCs using SVD Method.

1: Given a data matrix ($X = [x_1, x_2, \ldots, x_N]$), where N represents the total number of samples and $x_i(M \times 1)$ represents the i^{th} sample.
2: Compute the mean of all samples as in Equation (10).
3: Subtract the mean from all samples as in Equation (11).
4: Construct a matrix $Z = \frac{1}{\sqrt{N-1}} D^T$, $Z(N \times M)$.
5: Calculate SVD for Z matrix as in Equation (3).
6: The diagonal elements of S represent the square root of the sorted eigenvalues, $\lambda = diag(S^2)$, while the PCs are represented by the columns of R.
7: Select the eigenvectors that have the largest eigenvalues $W = \{R_1, R_2, \ldots, R_k\}$ to construct the PCA space.
8: All samples are projected on the lower dimensional space of PCA (W) as follows, $Y = W^T D$.

3.1 First Example: 2D-Class Example

In this section, the PCA space was calculated using the covariance matrix and SVD methods. All samples were represented using two variables or features.

3.1.1 Calculating Principal Components using Covariance Matrix Method

Given a data matrix $X = \{x_1, x_2, \ldots, x_8\}$, where x_i represents the i^{th} samples as denoted in Equation (13). Each sample of the matrix was represented by one column that consists of two features ($x_i \in \mathcal{R}^2$) to visualize it. The total mean (μ) was then calculated as in Equation (10) and its value was $\mu = \begin{bmatrix} 2.63 \\ 3.63 \end{bmatrix}$.

$$X = \begin{bmatrix} 1.00 & 1.00 & 2.00 & 0.00 & 5.00 & 4.00 & 5.00 & 3.00 \\ 3.00 & 2.00 & 3.00 & 3.00 & 4.00 & 5.00 & 5.00 & 4.00 \end{bmatrix} \tag{13}$$

The data were then subtracted from the mean as in Equation (11) and the values of D will be as follows:

$$D = \begin{bmatrix} -1.63 & -1.63 & -0.63 & -2.63 & 2.38 & 1.38 & 2.38 & 0.38 \\ -0.63 & -1.63 & -0.63 & -0.63 & 0.38 & 1.38 & 1.38 & 0.38 \end{bmatrix} \tag{14}$$

The covariance matrix (Σ) were then calculated as in Equation (12). The eigenvalues (λ) and eigenvectors (V) of the covariance matrix were then calculated. The values of the Σ, λ, and V are shown below.

$$\Sigma = \begin{bmatrix} 3.70 & 1.70 \\ 1.70 & 1.13 \end{bmatrix}, \ \lambda = \begin{bmatrix} 0.28 & 0.00 \\ 0.00 & 4.54 \end{bmatrix}, \text{ and } V = \begin{bmatrix} 0.45 & -0.90 \\ -0.90 & -0.45 \end{bmatrix} \tag{15}$$

From the results, we note that the second eigenvalue (λ_2) was more than the first one (λ_1). Moreover, the second eigenvalue represents $\frac{4.54}{0.28+4.54} \approx 94.19\%$ of the total eigenvalues, i.e. total variance, while the first eigenvalue represents $\frac{0.28}{0.28+4.54} \approx 5.81\%$, which reflects the robustness of the second eigenvector than the first one. Thus, the second eigenvector (i.e. second column of V) points to the direction of the maximum variance and hence it represents the first principal component of the PCA space.

Calculating the Projected Data:

The mean-centring data (i.e. data $-$ total mean) were then projected on each eigenvector as follows, ($Y_{v1} = v_1^T D$ and $Y_{v2} = v_2^T D$), where Y_{v1} and Y_{v2} represent the projection of the D on the first and second eigenvectors, i.e. v_1 and v_2, respectively. The values of Y_{v1} and Y_{v1} are shown below.

$$\begin{aligned} Y_{v1} &= \begin{bmatrix} -0.16 & 0.73 & 0.28 & -0.61 & 0.72 & -0.62 & -0.18 & -0.17 \end{bmatrix} \\ Y_{v2} &= \begin{bmatrix} 1.73 & 2.18 & 0.84 & 2.63 & -2.29 & -1.84 & -2.74 & -0.50 \end{bmatrix} \end{aligned} \tag{16}$$

Calculating the Reconstruction Error:

The reconstruction error between the original data and the reconstructed data using all eigenvectors or all principal components, i.e. there is no information lost, approximately tend to zero. In other words, if the original data are projected on all eigenvectors without neglecting anyone, and then reconstructed again, the error between the original and the reconstructed data will be zero. But, on the other hand, removing one or more eigenvectors to construct a lower dimensional space ($W \in \mathcal{R}^k$), reduces the dimension of the original data to k, thus some data are neglected, i.e. when the dimension of the lower dimensional space is lower than the original dimension, hence, there is a difference between the original data and the reconstructed data. The reconstruction error or residual depends on the number of the selected eigenvectors (k) and the robustness of those eigenvectors, which is measured by their corresponding eigenvalues.

In this example, the reconstruction error was calculated when the original data were first reconstructed as in Equation (7). Moreover, the original data were projected on the two calculated eigenvectors separately, i.e. $Y_{v1} = v_1^T D$ and $Y_{v2} = v_2^T D$. Hence, each eigenvector represents a separate lower dimensional space. The original data were then reconstructed using the same eigenvector that was used in the projection as follows, $\hat{X}_i = v_i Y_{vi} + \mu$. The values of the reconstructed data (i.e. $\hat{X}_1$ and $\hat{X}_2$) are as follows:

$$\begin{aligned} \hat{X_1} = v_1 Y_{v1} + \mu &= \begin{bmatrix} 2.55 & 2.95 & 2.75 & 2.35 & 2.95 & 2.35 & 2.55 & 2.55 \\ 3.77 & 2.97 & 3.37 & 4.17 & 2.98 & 4.18 & 3.78 & 3.78 \end{bmatrix} \\ \hat{X_2} = v_2 Y_{v2} + \mu &= \begin{bmatrix} 1.07 & 0.67 & 1.88 & 0.27 & 4.68 & 4.28 & 5.08 & 3.08 \\ 2.85 & 2.66 & 3.25 & 2.46 & 4.65 & 4.45 & 4.84 & 3.85 \end{bmatrix} \end{aligned} \tag{17}$$

The error between the original data and the reconstructed data that were projected on the first and second eigenvectors are denoted by E_{v1} and E_{v2}, respectively. The values of E_{v1} and E_{v2} are as follows:

$$\begin{aligned} E_{v1} = X - \hat{X_1} &= \begin{bmatrix} -1.55 & -1.95 & -0.75 & -2.35 & 2.05 & 1.65 & 2.45 & 0.45 \\ -0.77 & -0.97 & -0.37 & -1.17 & 1.02 & 0.82 & 1.22 & 0.22 \end{bmatrix} \\ E_{v2} = X - \hat{X_2} &= \begin{bmatrix} -0.07 & 0.33 & 0.12 & -0.27 & 0.32 & -0.28 & -0.08 & -0.08 \\ 0.15 & -0.66 & -0.25 & 0.54 & -0.65 & 0.55 & 0.16 & 0.15 \end{bmatrix} \end{aligned} \tag{18}$$

From the above results it can be noticed that the error between the original data and the reconstructed data that were projected on the second eigenvector, E_{v2}, was much lower than the reconstructed data that were projected on the first eigenvector, E_{v1}. Moreover, the total error between the reconstructed data that were projected on the first eigenvector and

the original data was $3.00 + 4.75 + 0.7 + 6.91 + 5.26 + 3.4 + 7.5 + 0.25 \approx 31.77$, while the error using the second eigenvector was equal to $0.03 + 0.54 + 0.08 + 0.37 + 0.52 + 0.38 + 0.03 + 0.03 \approx 1.98$[d]. Thus, the data lost when the original data were projected on the second eigenvector were much lower than the first eigenvector. Figure 5 shows the error E_{v1} and E_{v2} in green and blue lines. As shown, E_{v2} was much lower than E_{v1}.

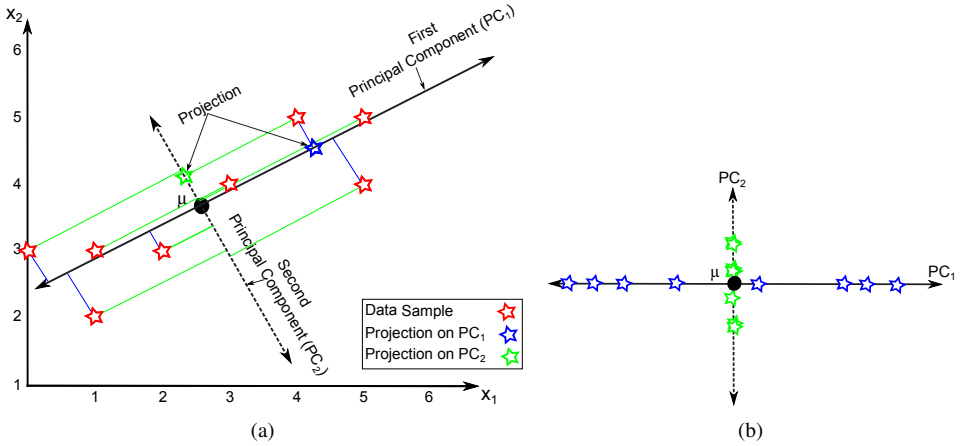


Figure 5: A visualized example of the PCA technique, (a) the dotted line represents the first eigenvector (v_1), while the solid line represents the second eigenvector (v_2) and the blue and green lines represent the reconstruction error using PC_1 and PC_2, respectively; (b) projection of the data on the principal components, the blue and green stars represent the projection onto the first and second principal components, respectively.

3.1.2 Calculating Principal Components using SVD Method

In this section, a lower dimensional space of PCA was calculated using SVD method. The original data (X) that were used in the covariance matrix example were used in this example. The first three steps in SVD method and covariance matrix methods are common. In the fourth step in SVD, the original data were transposed as follows, $Z = \frac{1}{N-1} D^T$. The values of Z are as follows:

$$Z = \begin{bmatrix} -0.61 & -0.24 \\ -0.61 & -0.61 \\ -0.24 & -0.24 \\ -0.99 & -0.24 \\ 0.90 & 0.14 \\ 0.52 & 0.52 \\ 0.90 & 0.52 \\ 0.14 & 0.14 \end{bmatrix} \tag{19}$$

[d]For example, the error between the first sample ($[1 \;\; 3]^T$) and the reconstructed sample using v_1 and v_2 was, $(2.55 - 1)^2 + (3.77 - 3)^2 \approx 3.00$ and $(1.07 - 1)^2 + (2.85 - 3)^2 \approx 0.03$, respectively.

SVD was then used to calculate L, S, and R as in Equation (3) and their values are as follows:

$$L = \begin{bmatrix} -0.31 & 0,12 & -0,07 & -0,60 & 0,58 & 0,15 & 0,41 & 0,04 \\ -0.39 & -0,52 & -0,24 & 0,20 & -0,29 & 0,53 & 0,31 & 0,14 \\ -0.15 & -0,20 & 0,96 & -0,01 & -0,01 & 0,08 & 0,07 & 0,02 \\ -0.47 & 0,43 & 0,02 & 0,69 & 0,32 & -0,05 & 0.12 & -0,01 \\ 0.41 & -0,51 & -0,04 & 0,31 & 0,68 & 0,08 & -0,09 & 0,02 \\ 0.33 & 0.44 & 0,08 & 0,02 & 0,02 & 0,82 & -0,15 & -0,05 \\ 0,49 & 0,12 & 0,05 & 0,17 & -0,15 & -0,12 & 0,83 & -0,03 \\ 0,09 & 0,12 & 0,02 & 0,00 & 0,01 & -0,05 & -0,04 & 0,99 \end{bmatrix}, S = \begin{bmatrix} 2.13 & 0 \\ 0 & 0.53 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 0.90 & -0.45 \\ 0.45 & 0.90 \end{bmatrix} \quad (20)$$

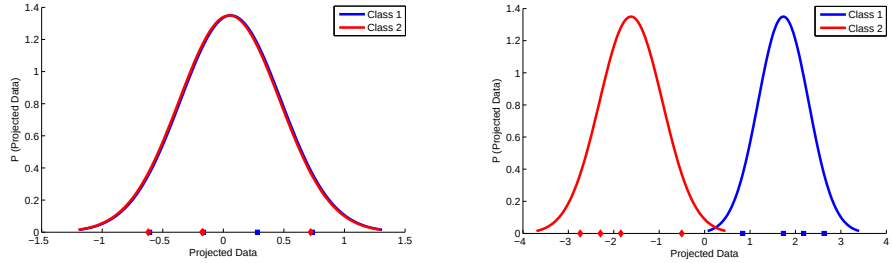
From the above results, we note that the first singular value (s_1) was more than the second one, thus, the first column of R represents the first principal component, and it was in the same direction of the second column of V that was calculated using the covariance matrix method. Moreover, the square roots of the eigenvalues that were calculated using the covariance matrix method were equal to the singular values of SVD. As a result, the eigenvalues and eigenvectors of the covariance matrix and SVD methods were the same.

Figure 5 shows a comparison between the two principal components or eigenvectors. The samples of the original data are represented by red stars (see Figure 5 **(a)**), each sample is represented by two features only ($x_i \in \mathcal{R}^2$) to be visualized. In other words, each sample is represented as a point in the two-dimensional space. Hence, there are two eigenvectors (v_1 and v_2) are calculated using SVD or solving covariance matrix. The solid line represents the second eigenvector (v_2), i.e. first principal component, while the dotted line represents the first eigenvector (v_1), i.e. second principal component. Figure 5 **(b)** shows the projection of the original data on the two principal components.

3.1.3 Classification Example

PCA was used to remove the redundant data or noise that have a good impact on the classification problem. Hence, PCA was used commonly as a feature extraction method. In this section, a comparison between the two eigenvectors was performed to show which one was suitable to construct a sub-space to discriminate between different classes.

Assume that, the original data consists two classes. The first class ($\omega_1 = \{x_1, x_2, x_3, x_4\}$) consists of the first four observations or samples, while the other four samples represent the second class ($\omega_2 = \{x_5, x_6, x_7, x_8\}$). As shown in Figure 6 **(a)**, more important data or information that were used to discriminate between the two classes were lost and discarded when the data were projected on the second principal component. On the other hand, the important information of the original data were preserved when the data were projected on the first principal component, hence, the two classes can be discriminated as shown in Figure 6 **(b)**. These findings may help to understand that, the robust eigenvectors were preserved the important and discriminative information that were used to classify between different classes.



(a) (b)

Figure 6: Probability density function of the projected data of the first example, (a) the projected data on PC_2, (b) the projected data on PC_1.

3.2 Multi-Class Example

In this example, each sample was represented by four variables. In this experiment, the second variable was constant for all observations as shown below.

$$X = \begin{bmatrix} 1.00 & 1.00 & 2.00 & 0.00 & 7.00 & 6.00 & 7.00 & 8.00 \\ 2.00 & 2.00 & 2.00 & 2.00 & 2.00 & 2.00 & 2.00 & 2.00 \\ 5.00 & 6.00 & 5.00 & 9.00 & 1.00 & 2.00 & 1.00 & 4.00 \\ 3.00 & 2.00 & 3.00 & 3.00 & 4.00 & 5.00 & 5.00 & 4.00 \end{bmatrix} \tag{21}$$

The covariance matrix of the given data was calculated and its values are shown below.

$$\Sigma = \begin{bmatrix} 10.86 & 0 & -7.57 & 2.86 \\ 0 & 0 & 0 & 0 \\ -7.57 & 0 & 7.55 & -2.23 \\ 2.86 & 0 & -2.23 & 1.13 \end{bmatrix} \tag{22}$$

As shown from the results above, the values of second row and column were zeros, which reflects that the variance of the second variable was zeros because the second variable was constant.

The eigenvalues and eigenvectors of the covariance matrix of the above data are shown below.

$$\lambda = \begin{bmatrix} 17.75 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.46 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix} \qquad V = \begin{bmatrix} 0.76 & 0.62 & -0.20 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \\ -0.61 & 0.79 & 0.10 & 0.00 \\ 0.21 & 0.05 & 0.98 & 0.00 \end{bmatrix} \tag{23}$$

From these results, many notices can be seen. First, the first eigenvector represents the first principal component because it was equal to $\frac{17.75}{17.75+1.46+0.33+0} \approx 90.84\%$ of the total variance of the data. Second, the first three eigenvectors represent 100% of the total variance of the total data and the fourth eigenvector was redundant. Third, the second variable, i.e. second row, will be neglected completely when the data were projected on any of the best three eigenvectors, i.e. the first three eigenvectors. Fourth, as denoted in Equation (24), the projected data on the fourth eigenvector preserved only the second variable and all the other original data were lost, and the reconstruction error was ≈ 136.75, while the reconstruction error was ≈ 12.53 , 126.54 , 134.43 when the data were projected on the first three eigenvectors, respectively, as shown in Figure 7.

$$\begin{aligned} Y_{v_1} &= \begin{bmatrix} -2.95 & -3.78 & -2.19 & -6.16 & 4.28 & 3.12 & 4.49 & 3.20 \end{bmatrix} \\ Y_{v_2} &= \begin{bmatrix} -1.20 & -0.46 & -0.58 & 1.32 & -0.58 & -0.39 & -0.54 & 2.39 \end{bmatrix} \\ Y_{v_3} &= \begin{bmatrix} 0.06 & -0.82 & -0.14 & 0.64 & -0.52 & 0.75 & 0.45 & -0.43 \end{bmatrix} \\ Y_{v_4} &= \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix} \end{aligned} \tag{24}$$

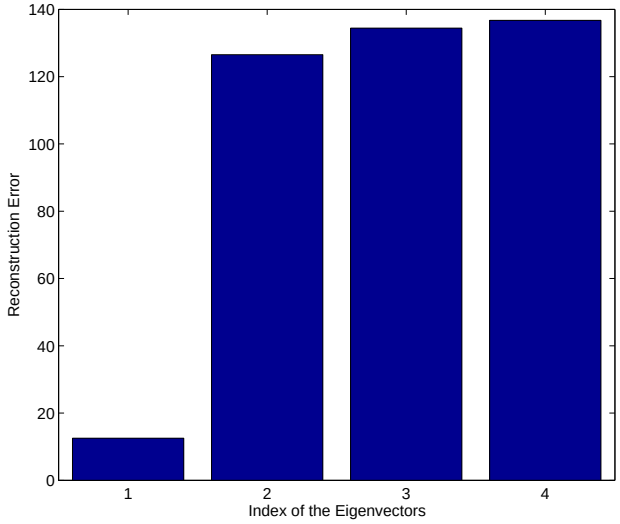


Figure 7: Reconstruction error of the four eigenvectors in the multi-class example.

4 Experimental Results and Discussion

In this section, three different experiments were conducted to understand the main idea of the PCA technique. In the first experiment, the PCA technique was used to identify

individuals, i.e. PCA was used as a feature extraction method. In the second experiment, PCA was used to compress a digital image through removing the eigenvectors that represent the minimum variance from the PCA space and hence some redundant information will be removed from the original image. In this experiment, different numbers of eigenvectors were used to preserve the total information of the original image. In the third experiment, the PCA technique was used to reduce the dimension of the high-dimensional data to be visualized. Different datasets with different dimensions were used in this experiment. In all experiments, the principal components were calculated using the covariance matrix method (see Algorithm (1)). MATLAB codes for all experiments are presented in Appendix A.2.

4.1 Biometric Experiment

The aim of this experiment was to investigate the impact of the number of principal components on the classification accuracy and the classification CPU time using these principal components.

4.1.1 Experimental Setup

Three biometric datasets were used in this experiment. The descriptions of the datasets are as follows:

- Olivetti Research Laboratory, Cambridge (ORL[e]) dataset (Samaria and Harter, 1994), which consists of 40 individuals, each has ten grey scale images. The size of each image was 92×112.
- Ear dataset images[f], which consists of 17 individuals, each has six grey scale images (Carreira, 1995). The images have different dimensions, thus all images were resized to be 64×64.
- Yale[g] face dataset images, which contains 165 grey scale images in GIF format of 15 individuals (Yang et al., 2004). Each individual has 11 images in different expressions and configuration: center-light, happy, left-light, with glasses, normal, right-light, sad, sleepy, surprised, and wink. The size of each image was 320×243.

Figure 8 shows samples from the face and ear datasets.

In this experiment, the Nearest Neighbor (NN) classifier was used. The nearest neighbor (minimum distance) classifier (Duda et al., 2012) was used to classify the testing image by

[e] http://www.cam-orl.co.uk
[f] http://faculty.ucmerced.edu/mcarreira-perpinan/software.html
[g] http://vision.ucsd.edu/content/yale-face-database

comparing its position in the PCA space with positions of the training images. The results of this experiment were evaluated using two different assessment methods, namely, accuracy and CPU time. Accuracy or recognition rate represents the percentage of the total number of predictions that were correct as denoted in Equation (25), while the CPU time in this experiment was used to measure the classification time. Finally, the experiment environment includes Window XP operating system, Intel(R) Core(TM) i5-2400 CPU @ 3.10 GHz, 4.00 GB RAM, and Matlab (R2013b).

$$ACC = \frac{\text{Number of Correctly Classified Samples}}{\text{Total Number of Testing Samples}} \tag{25}$$

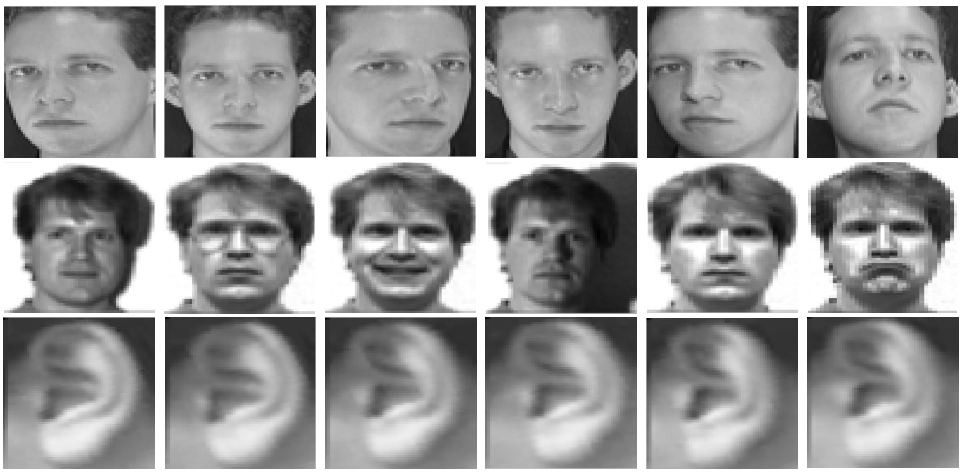


Figure 8: Samples of the first individual in: ORL face dataset (top row); Yale face dataset (middle row); and Ear dataset (bottom row).

4.1.2 Experimental Scenario

In this experiment, different numbers of principal components were used to construct the PCA space. Hence, the dimension of the PCA space and the projected data were changed based on the number of selected principal components (k). In this experiment, three sub-experiments were performed. In the first sub-experiment, ORL face dataset was used. The training images consist of seven images from each subject (i.e. $7 \times 40 = 280$ images), while the remaining images were used for testing (i.e. $3 \times 40 = 120$ images). In the second sub-experiment, the ear dataset was used. The training images consist of three images from each subject, i.e. $3 \times 17 = 51$ images, and the remaining images were used for testing, i.e. $3 \times 17 = 51$ images. In the third sub-experiment, Yale face dataset was used. The training images consist of six images from each subject (i.e. $6 \times 15 = 90$ images), while the remaining images were used for testing (i.e. $5 \times 15 = 75$ images). In the three sub-experiments, the training images were selected randomly. Algorithm (3) summarizes the detailed steps of this experiment. Table 2 lists the results of this experiment.

Algorithm 3 : Steps of Biometric Experiment.

1: Read the training images ($X = \{x_1, x_2, \ldots, x_N\}$), where ($x_i(r \times c)$) represents the i^{th} training image, r represents the rows (height) of x_i, c represents columns (width) of x_i, and N represents the total number of training images and all training images have the same dimensions.
2: Convert all images in vector representation $Z = \{z_1, z_2, \ldots, z_N\}$, where $Z(M \times 1)$, hence each image or sample has $M = r \times c$ dimensions.
3: Follow the same steps of the PCA Algorithm (1) to calculate the mean (μ), subtract the mean from all training images ($d_i = Z_i - \mu, i = 1, 2, \ldots, N$), calculate covariance matrix (Σ), and the eigenvalues ($\lambda(M \times M)$) and eigenvectors ($V(M \times M)$) of Σ.
4: Sort the eigenvectors according to their corresponding eigenvalues.
5: Select the eigenvectors that have the k largest eigenvalues to construct the PCA space ($W = \{v_1, v_2, \ldots, v_k\}$).
6: Project the training images after subtracting the mean from each image as follows, $y_i = W^T d_i, i = 1, 2, \ldots, N$.
7: Given an unknown image $T(r \times c)$. Convert it to vector form, which is denoted as follows, $I(M \times 1)$.
8: Project the unknown image on the PCA space as follows, $U = W^T I$, then classify the unknown image by comparing its position with positions of the training images in the PCA space.

Table 2 A comparison between ORL, Ear, and Yale datasets in terms of accuracy (%), CPU time (sec), and cumulative variance (%) using different number of eigenvectors (biometric experiment).

Number of Eigenvectors	ORL Dataset			Ear Dataset			Yale Dataset		
	Acc. (%)	CPU Time (sec)	Cum. Var. (%)	Acc. (%)	CPU Time (sec)	Cum. Var. (%)	Acc. (%)	CPU Time (sec)	Cum. Var. (%)
1	13.33	0.074	18.88	15.69	0.027	29.06	26.67	0.045	33.93
5	80.83	0.097	50.17	80.39	0.026	66.10	76.00	0.043	72.24
10	94.17	0.115	62.79	90.20	0.024	83.90	81.33	0.042	85.13
15	95.00	0.148	69.16	94.12	0.028	91.89	81.33	0.039	90.18
20	95.83	0.165	73.55	94.12	0.033	91.89	84.00	0.042	93.36
30	95.83	0.231	79.15	94.12	0.033	98.55	85.33	0.061	96.60
40	95.83	0.288	82.99	94.12	0.046	99.60	85.33	0.064	98.22
50	95.83	0.345	85.75	94.12	0.047	100.00	85.33	0.065	99.12
100	95.83	0.814	93.08	94.12	0.061	100.00	85.33	0.091	100.00

Acc. accuracy; Cum. Cumulative; Var. variance.

4.1.3 Discussion

The results in Table 2 show that the accuracy and CPU time of face and ear biometrics using different numbers of principal components.

As shown in Table 2, the accuracy of all types of biometrics was proportional to the number of principal components. However, the accuracy increases until it reaches an extent. As

shown in the table, the accuracy of the ORL face dataset remains constant when the number of principal components increased from 20 to 100. Similarly, in the ear and Yale datasets, the accuracy was constant when the number of principal components was more than 10 and 20, respectively. Further analysis of the table showed that the total variance of the first 100 PCs of all datasets was ranged from 93.08 to 100. Hence, these principal components have the most important and representative information that were used to discriminate between different classes. Consequently, increasing the number of principal components increases the dimension of the projected data, hence increases the classification's CPU time. For example, in ORL dataset, the dimensions of the original image were $M \times 280$, where 280 is the number of training images, while the dimensions of the projected data ranged from (1×280) when one principal component was used to (100×280) when 100 principal components were used. These findings may help to understand that increasing the number of principal components, increases the dimension of the PCA space, which needs more CPU time and preserves more important information, hence increases the accuracy.

(a) (b)

Figure 9: Original images (a) 512×512 8 bit/pixel original image of Lena, (b) 256×256 8 bit/pixel original image of Cameraman.

4.2 Image Compression Experiment

The aim of this experiment was to use the PCA technique as an image compression technique to reduce the size of the image by removing some redundant information.

4.2.1 Experimental Setup

In this experiment, two different images were used, namely Lenna or Lena and Cameraman image (see Figure 9), which are standard test images that widely used in the field of image processing and both two images are included in Matlab (Gonzalez et al., 2004)[h].

Two different assessment methods were used to evaluate this experiment. The first assessment method was the Mean Square Error (MSE). MSE is the difference or error between the original image (I) and the reconstructed image ($\hat{I}$) as shown in Equation (26). The second assessment method was the compression ratio (CR) or the compression power, which is the ratio between the size of original image, i.e. the number of unit memory required to represent the original image, and the size of the compressed image as denoted in Equation (27). In other words, the compression ratio was used to measure the reduction in the size of the reconstructed image that produced by a data compression method.

$$MSE = \frac{1}{rc}\sum_{i=1}^{r}\sum_{j=1}^{c}(I(i,j) - \hat{I}(i,j))^2 \tag{26}$$

$$CR = \frac{\text{Uncompressed Size}}{\text{Compressed Size}} \tag{27}$$

where I represents the original image, $\hat{I}$ is the reconstructed image, r and c represents the number of rows and columns of the image, respectively, i.e. r and c represent the dimension of the image.

4.2.2 Experimental Scenario

The image compression technique is divided into two techniques, namely, *lossy* and *lossless* compression techniques. In the lossless compression technique, all information that was originally in the original file remains after the file is uncompressed, i.e. all information is completely restored. On the other hand, in the lossy compression technique, some redundant information from the original file is lost. In this experiment, PCA was used to reduce the dimension of the original image and the original data was restored again as shown in Algorithm (4). As shown in the algorithm, each column of pixels in an image represents an observation or sample. In this experiment, different numbers of eigenvectors were used to construct the PCA space that was used to project the original image, i.e. compression, and reconstruct it again, i.e. decompression, to show how the number of PCs affects the CR and MSE. Figures 10 and 11 and Table 3 summarizes the results of this experiment.

[h] http://graphics.cs.williams.edu/data/images.xml

Algorithm 4 : Image Compression-Decompression using PCA

1: Read the original image ($I(r \times c)$), where r and c represent the rows (height) and columns (width) of the original image. Assume each column represents one observation or sample as follows, $I = [I_1, I_2, \ldots, I_c]$, where I_i represents the i^{th} column.
2: Follow the same steps of the PCA Algorithm (1) to calculate the mean (μ), subtract the mean from all columns of the original image ($D = I - \mu$), calculate the covariance matrix (Σ), eigenvalues (λ), and eigenvectors (V), where the number of eigenvalues and eigenvectors are equal to the number of rows of the original image.
3: Sort the eigenvectors according to their corresponding eigenvalues.
4: Select the eigenvectors that have the k largest eigenvalues to construct the PCA space ($W = \{v_1, v_2, \ldots, v_k\}$).
5: Project the original image after subtracting the mean from each column as follows, $Y = W^T D$. This step represents the compression step and Y represents the compressed image.
6: Reconstruct the original image as follows:

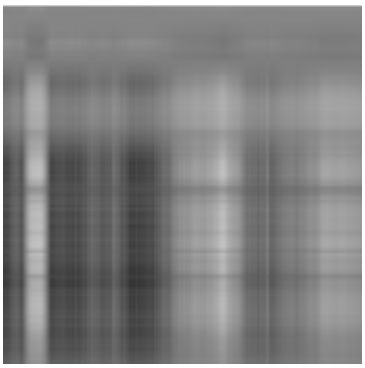
$$\hat{I} = WY + \mu \tag{28}$$

Table 3 Compression ratio and mean square error of the compressed images using different percentages of the eigenvectors (image compression experiment).

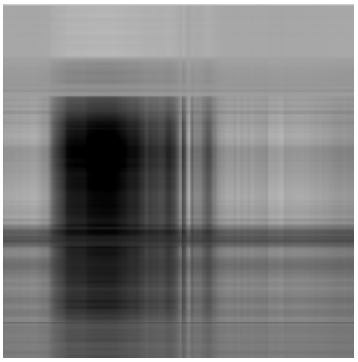
Percentage of the used Eigenvectors	Lena Image			Cameraman Image		
	MSE	CR	Cumulative Variance (%)	MSE	CR	Cumulative Variance (%)
10	5.3100	512:51.2	97.35	8.1057	256:25.6	94.56
20	2.9700	512:102.4	99.25	4.9550	256:51.2	98.14
30	1.8900	512:153.6	99.72	3.3324	256:76.8	99.24
40	1.3000	512:204.8	99.87	2.0781	256:102.4	99.73
50	0.9090	512:256	99.94	1.1926	256:128	99.91
60	0.6020	512:307.2	99.97	0.5588	256:153.6	99.98
70	0.3720	512:358.4	99.99	0.1814	256:179.2	100.00
80	0.1935	512:409.6	100.00	0.0445	256:204.8	100.00
90	0.0636	512:460.8	100.00	0.0096	256:230.4	100.00
100 (All)	0.0000	512:512=1	100.00	0.0000	1	100.00

4.2.3 Discussion

Table 3 shows the compression ratio, mean square error, and cumulative variance of the compressed images using different principal components of the PCA space. From the table, different notices can be seen. First, the minimum compression ratio (i.e. $CR = 1$) achieved when all principal components were used to construct the PCA space and the MSE was equal to zero, hence, the compression is called lossless compression because all data can be restored. On the other hand, decreasing the number of principal components means more data were lost and the compression ratio increased, consequently the MSE increased. For example, the compression ratio reached to $(512 : 1)$ and $(256 : 1)$ while the MSE was 30.07 and 25.97, respectively, when one principal component, i.e. the first principal component, was used as a lower dimensional space as shown in Figure 10**(a,b)**. As shown from the figure,



(a) One principal component, CR=512:1, and MSE=30.7



(b) One principal component, CR=256:1, and MSE=25.97

(c) 10% of the principal components, CR=10:1, and MSE=5.31

(d) 10% of the principal components, CR=10:1, and MSE=8.1057

(e) 50% of the principal components, CR=2:1, and MSE=0.909

(f) 50% of the principal components, CR=2:1, and MSE=1.1926

Figure 10: A comparison between different compressed images of Lena and Cameraman images using different numbers of the eigenvectors.

a large amount of data were lost when one principal component was used and the compressed images much deviated from the original images. On the other hand, increasing the number of principal components to 10% (see Figure 10**(c,d)**) and 50% (see Figure 10**(e,f)**) reduced the MSE and CR. Further analysis of the robustness of the principal components showed that the most important information were concentrated in the eigenvectors that have the highest eigenvalues as shown in Figure 11. Moreover, from the table, the robustness of the eigenvectors decreased dramatically, and more than 98% of the total variance were concentrated in only 20% of the total eigenvectors of the PCA space. To conclude, the MSE and CR were inversely proportional to the number of principal components that were used to construct the PCA space, i.e. projection space, because decreasing the number of principal components means more reduction for the original data. Another interesting finding was that the PCA technique can be used as a lossy/lossless image compression method.

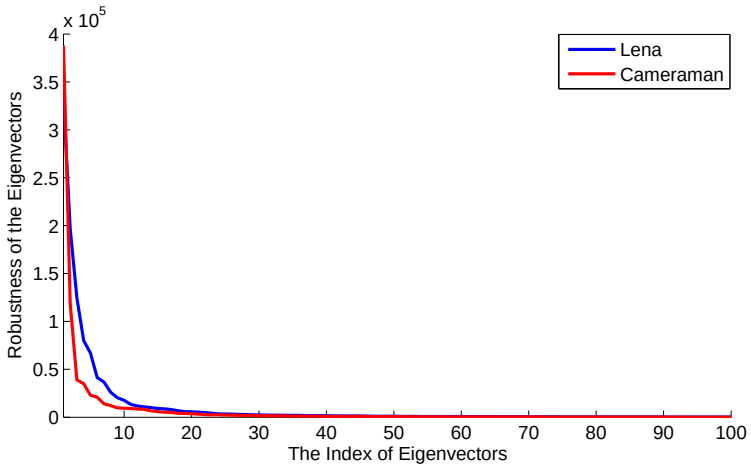


Figure 11: The robustness, i.e. total variance (see Equation (9)), of the first 100 eigenvectors using Lena and Cameraman images.

4.3 Data Visualization Experiment

The aim of this experiment was to use the PCA technique to reduce the dimension of the high dimensional datasets to be visualized. In this experiment, the PCA space was constructed using two or three principal components to visualize the datasets in 2D or 3D, respectively.

4.3.1 Experimental Setup

Six standard datasets were used in this experiment as shown in Table 4. Each dataset has different numbers of attributes, classes, and samples. Table 4 shows the number of classes

Table 4 Datasets descriptions.

Dataset	Number of Classes	Number of Features	Number of Samples
Iris	3	4	150
Iono	2	34	351
Ovarian	2	4000	216
ORL	5	10304	50
Ear$_{64\times64}$	5	4096	30
Yale	5	77760	55

was in the range of [2, 5], the dimensionality was in the range of [4, 77760], and the number of samples was in the range of [30, 351].

The first dataset was the iris dataset which consists of three different types of flowers, namely, Setosa, Versicolor, and Versicolor. The second dataset represents the radar data that consists of two classes, namely, Good and Bad. The third dataset was the Ovarian dataset that was collected at the Pacific Northwestern National Lab (PNNL) and Johns Hopkins University. This dataset was considered one of the large-scale or high-dimensional datasets used to investigate tumors through deep proteomic analysis and it consists of two classes, namely Normal and Cancer. ORL, ear, and Yale datasets that were described in the first experiment (see Section 4.1). The images of ear dataset have different dimensions, but in this experiment, all images were resized to be in the same dimension (64×64). In this experiment, five subjects only were used from ORL, ear, and Yale datasets, to be simple in visualization and faster in a computation.

4.3.2 Experimental Scenario

Due to the high dimensionality of some datasets, it is difficult to visualize data when it has more than three variables. To solve this problem, one of the dimensionality reduction methods are used to find the two-dimensional or three-dimensional spaces which are more similar to the original space. This 2D or 3D subspace is considered a window into the original space, which is used to visualize the original data. In this experiment, the PCA technique was used to reduce the number of attributes, features, or dimensions to be visualized. In other words, PCA searches for a lower dimensional space, i.e. 2D or 3D, and project the original dataset on that space. Only the first two principal components were used to construct the 2D-PCA lower dimensional space to visualize the dataset in 2D, while the first three principal components were used to construct the 3D-PCA lower dimensional space to visualize the datasets in 3D. Figures 12 and 13 show the 2D and 3D visualizations of all datasets that were listed in Table 4. Moreover, Table 5 illustrates the total variance, i.e. robustness, of the PCA space and the MSE between the original and reconstructed data.

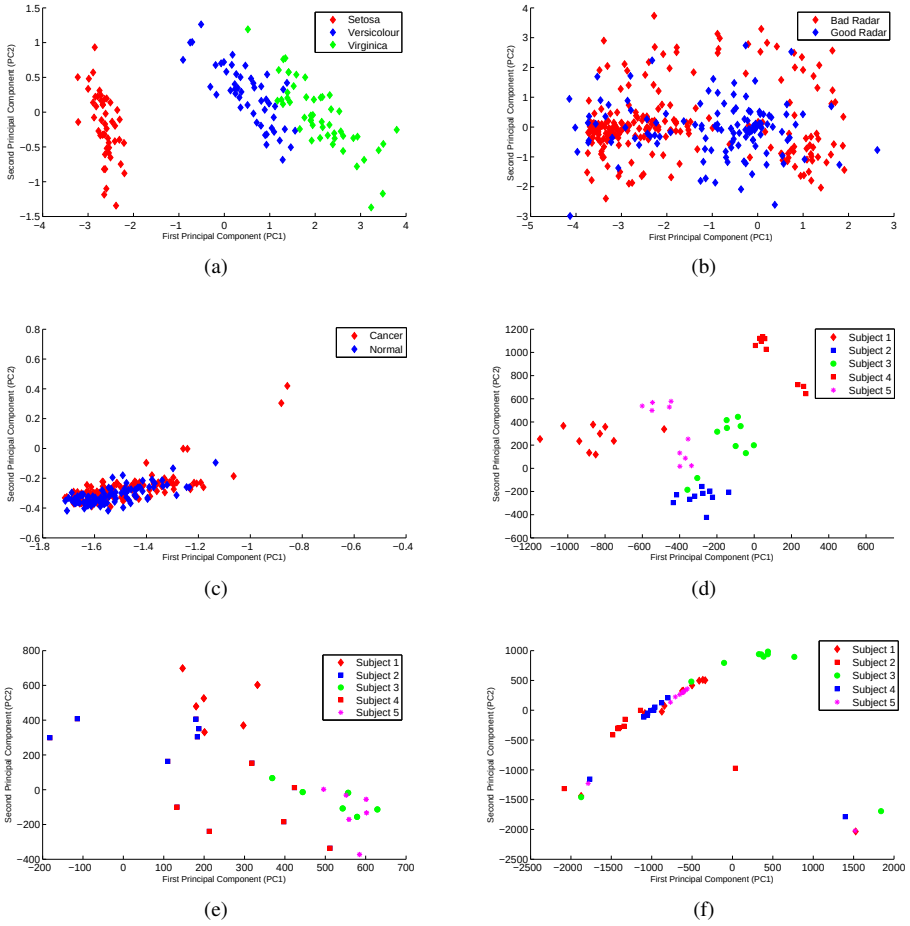


Figure 12: 2D Visualization of the datasets listed in Table 4, (a) Iris dataset, (b) Iono dataset, (c) Ovarian dataset, (d) ORL dataset, (e) Ear dataset, (f) Yale dataset.

Table 5 A comparison between 2D and 3D visualization in terms of MSE and robustness using the datasets that were listed in Table 4.

Dataset	**2D**		**3D**	
	Robustness (in %)	MSE	**Robustness (in %)**	MSE
Iris	97.76	0.12	99.48	0.05
Iono	43.62	0.25	51.09	0.23
Ovarian	98.75	0.04	99.11	0.03
ORL	34.05	24.03	41.64	22.16
Ear$_{64\times64}$	41.17	15.07	50.71	13.73
Yale	48.5	31.86	57.86	28.80

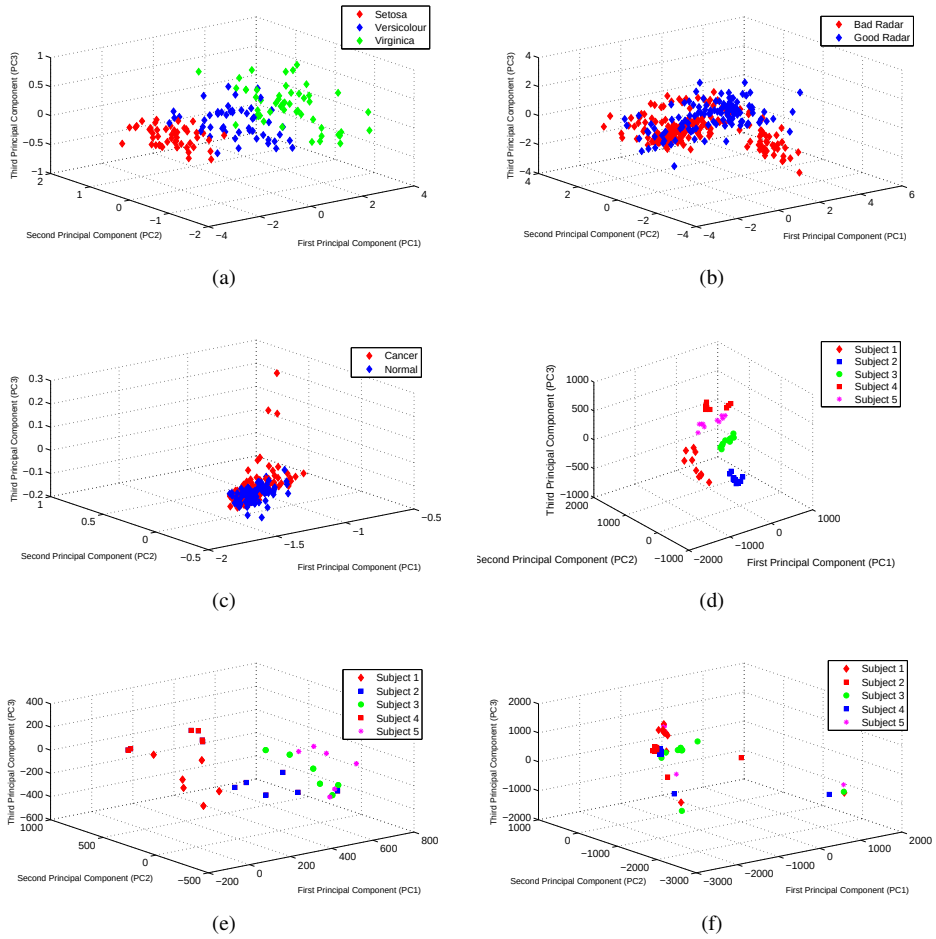


Figure 13: 3D Visualization of the datasets that were listed in Table 4, (a) Iris dataset, (b) Iono dataset, (c) Ovarian dataset, (d) ORL dataset, (e) Ear dataset, (f) Yale dataset.

4.3.3 Discussion

As shown in Figure 12, all datasets listed in Table 4 were visualized in 2D. Table 4 shows the robustness (see Equation 9) of the calculated 2D-PCA space. As shown from the table, the robustness was ranged from 34.05% to 98.75%. These variations depend on the dimension of the datasets and the variances of the original datasets' variables. For example, the Ovarian dataset could be represented only by two principal components, i.e. two variables, that have 98.75% of the total variance, although it has originally 4000 variables, which reflects that the amount of important information which were preserved in the 2D-PCA space. Moreover, from the table, we note that the MSE were ranged from 0.04 to 31.86, which reflects that small amount information were lost by projecting the datasets on the 2D-PCA space. Further analysis showed that the MSE was inversely proportional to the robustness of the

2D-space, e.g. the minimum MSE achieved when the robustness reached 98.75% in the Ovarian dataset which represents the maximum robustness.

Figure 13 shows the 3D visualizations of the datasets that were listed in Table 4. In this scenario, the robustness was higher than in 2D scenario, due to the amount of information stored in 3D-PCA space were more than in 2D-PCA space. Therefore, the MSE achieved was lower than in 2D scenario, which reflects that more data were preserved and the visualization in 3D-PCA space was more representative.

From these findings, it was possible to conclude that the PCA technique was feasible to reduce the dimension of the high dimensional datasets to be visualized in 2D or 3D.

5 Conclusions

Due to the problems of high dimensional datasets, dimensionality reduction techniques, e.g. PCA, are important for many applications. Such techniques should achieve the dimensionality reduction by projecting the data on the space which represents the direction of the maximum variance of the given data. This paper explains visually the steps of calculating the principal components using two different methods, i.e. covariance matrix and SVD, and how to select the principal components to construct the PCA space. In addition, a number of numerical examples were given and graphically illustrated to explain how the PCs were calculated and how the PCA space was constructed. In the numerical examples, the mathematical interpretation of the robustness and the selection of the PCs, projecting data, reconstructing data, and measuring the deviation between the original and reconstructed data were discussed. Moreover, using standard datasets, a number of experiments were conducted to (1) Understand the main idea of the PCA and how it is used in different applications, (2) Investigate and explain the relation between the number of PCs and the robustness of the PCA space.

References

Larose, D.T. (2014) 'Discovering knowledge in data: an introduction to data mining', John Wiley Sons, Second Edition.

Bramer, M. (2013) 'Principles of Data Mining', *Springer, Second Edition*.

Saeys, Y., Inza, I., & Larrañaga, P. (2007) 'A review of feature selection techniques in bioinformatics', *bioinformatics*, Vol. 23, No. 19, pp. 2507–2517.

Venna, J., Peltonen, J., Nybo, K., Aidos, H., & Kaski, S. (2010) 'Information retrieval perspective to nonlinear dimensionality reduction for data visualization', *The Journal of Machine Learning Research*, Vol. 11, pp. 451–490.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012) 'Pattern classification', *John Wiley & Sons, Second Edition*.

Chiang, L. H., Russell, E. L., & Braatz, R. D. (2000) 'Fault diagnosis in chemical processes using Fisher discriminant analysis, discriminant partial least squares, and principal component analysis', *Chemometrics and intelligent laboratory systems*, Vol. 50, No. 2, pp. 243–252.

Tharwat, A., Ibrahim, A., Hassanien, A. E., & Schaefer, G. (2015) 'Ear recognition using block-based principal component analysis and decision fusion', *in Proceedings of Pattern Recognition and Machine Intelligence*, Vol. 9124, pp. 246–254.

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000) 'A global geometric framework for nonlinear dimensionality reduction', *Science*, Vol. 290, No. 5500, pp. 2319–2323.

Kirby, M. (2000) 'Geometric data analysis: an empirical approach to dimensionality reduction and the study of patterns', John Wiley & Sons, First Edition.

Lu, J., Plataniotis, K. N., & Venetsanopoulos, A. N. (2003) 'Face recognition using LDA-based algorithms', *IEEE Transactions on Neural Networks*, Vol. 14, No. 1, pp. 195–200.

Cui, J. R. (2012) 'Multispectral palmprint recognition using Image-Based Linear Discriminant Analysis', *International Journal of Biometrics*, Vol. 4, No. 2, pp. 106–115.

Tharwat, A., Ibrahim, A., & Ali, H. A. (2012) 'Personal identification using ear images based on fast and accurate principal component analysis', *In proceedings 8th International Conference on Informatics and Systems (INFOS)*, Vol. 56, pp. 59–65.

Wu, M. C., Zhang, L., Wang, Z., Christiani, D. C., & Lin, X. (2009) 'Sparse linear discriminant analysis for simultaneous testing for the significance of a gene set/pathway and gene selection', *Bioinformatics*, Vol. 25, No. 9, pp. 1145–1151.

Hastie, T., & Tibshirani, R. (1996) 'Discriminant analysis by Gaussian mixtures', *Journal of the Royal Statistical Society Series B (Methodological)*, pp. 155–176.

Hinton, G. E., & Salakhutdinov, R. R. (2006) 'Reducing the dimensionality of data with neural networks' *Science*, Vol. 313, No. 5786, pp. 504–507.

Scholkopft, B., & Mullert, K. R. (1999) 'Fisher discriminant analysis with kernels', *Neural networks for signal processing IX*, Vol. 1, No. 1, pp. 23–25.

Müller, W., Nocke, T., & Schumann, H. (2006, January) 'Enhancing the visualization process with principal component analysis to support the exploration of trends', *In Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation- Australian Computer Society, Inc.*, Vol. 60, pp. 121-130.

Barshan, E., Ghodsi, A., Azimifar, Z., & Jahromi, M. Z. (2011) 'Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds', *Pattern Recognition*, Vol. 44, No. 7, pp. 1357–1371.

Kambhatla, N., & Leen, T. K. (1997) 'Dimension reduction by local principal component analysis', *Neural Computation*, Vol. 9, No. 7, pp. 1493–1516.

Jolliffe, I. (2002) 'Principal component analysis', *John Wiley & Sons*.

Thomas, C. G., Harshman, R. A., & Menon, R. S. (2002) 'Noise reduction in BOLD-based fMRI using component analysis', *Neuroimage*, Vol. 17, No. 3, pp. 1521–1537.

Hyvärinen, A., Karhunen, J., & Oja, E. (2004) 'Independent component analysis', Vol. 46. John Wiley & Sons.

Gaber, T., Tharwat, A., Snasel, V., & Hassanien, A. E. (2015) 'Plant Identification: Two Dimensional-Based Vs. One Dimensional-Based Feature Extraction Methods', *In Proceedings of 10th International Conference on Soft Computing Models in Industrial and Environmental Applications*, Vol. 368, pp. 375–385.

Roweis, S. T., & Saul, L. K. (2000) 'Nonlinear dimensionality reduction by locally linear embedding', *Science*, Vol. 290, No. 5500, pp. 2323–2326.

Dash, M., Liu, H., & Yao, J. (1997, November) 'Dimensionality reduction of unsupervised data', *Proceedings of 9^{th} IEEE International Conference on Tools with Artificial Intelligence*, pp. 532–539.

Belkin, M., & Niyogi, P. (2003) 'Laplacian eigenmaps for dimensionality reduction and data representation', *Neural computation*, Vol. 15, No. 6, pp. 1373–1396.

Turk, M., & Pentland, A. P. (1991, June) 'Face recognition using eigenfaces',*Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'91.*, pp. 586–591.

Turk, M., & Pentland, A. (1991) 'Eigenfaces for recognition', *Journal of cognitive neuroscience*, Vol. 3, No. 1, pp. 71–86.

Wold, S., Esbensen, K., & Geladi, P. (1987) 'Principal component analysis', *Chemometrics and intelligent laboratory systems*, Vol. 2 , No. 1, pp. 37–52.

Shlens, J. (2014) 'A tutorial on principal component analysis', *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*.

Bishop, C. M. (2006) 'Pattern recognition and machine learning', *springer New York*, Vol. 4.

Strang, G., & Aarikka, K. (1986) 'Introduction to applied mathematics', *Wellesley-Cambridge Press, Massachusetts, Fourth Edition*, Vol. 16.

Hyvärinen, L. (1970) 'Principal component analysis', *Mathematical Modeling for Industrial Processes*, pp. 82–104.

Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003) 'Singular value decomposition and principal component analysis', *A practical approach to microarray data analysis, Springer US.*, pp. 91–109.

Greenberg, M. D. (2001) 'Differential equations & Linear algebra', *Prentice Hall*.

Alter, O., Brown, P. O., & Botstein, D. (2000) 'Singular value decomposition for genome-wide expression data processing and modeling', *Proceedings of the National Academy of Sciences*, Vol. 97, No. 18, pp. 10101–10106.

Abdi, H., & Williams, L. J. (2010) 'Principal component analysis', *Wiley Interdisciplinary Reviews: Computational Statistics*, Vol. 2, No. 4, pp. 433–459.

Samaria, F. S., & Harter, A. C. (1994, December) 'Parameterisation of a stochastic model for human face identification', *Proceedings of the Second IEEE Workshop on In Applications of Computer Vision*, pp. 138–142).

Carreira-Perpinan, M. A. (1995) 'Compression neural networks for feature extraction: Application to human recognition from ear images', *MS thesis, Faculty of Informatics, Technical University of Madrid, Spain*.

Yang, J., Zhang, D., Frangi, A. F., & Yang, J. Y. (2004) 'Two-dimensional PCA: a new approach to appearance-based face representation and recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 1, pp. 131–137.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2004) 'Digital image processing using MATLAB', *Pearson Education India.*

Asuncion, Arthur, & David Newman. "UCI machine learning repository." (2007).

Appendix A

In this section, a Matlab codes for the numerical examples, experiments, and the two methods that are used to calculate the PCA are introduced.

A.1: MATLAB Codes for Our Numerical Examples

In this section, the codes the numerical examples in section 3 are introduced.

A.1.1: Covariance Matrix Example

This code follows the steps of the numerical example in section 3.1.1.

```
1
2 clc
3 clear all;
```

```
X = [1      1      2      0      5      4      5      3
     3      2      3      3      4      5      5      4];


[r,c]=size(X);

% Compute the mean of the data matrix "The mean of each ...
    row" (Equation (10))
m=mean(X')';

% Subtract the mean from each image [Centering the data] ...
    (Equation (11))
d=X-repmat(m,1,c);

% Compute the covariance matrix (co) (Equation (12))
co=1 / (c-1)*d*d';

% Compute the eigen values and eigen vectors of the ...
    covariance matrix
[eigvector,eigvl]=eig(co);

% Project the data on the first eigenvector
% The first principal component is the second eigenvector
PC1=eigvector(:,2)
% Project the data on the PCa
Yv2=PC1'*d
% The second principal component is the first eigenvector
PC2=eigvector(:,1)
% Project the data on the PCa
Yv1=PC2'*d

% Reconstruct the data from the projected data on PC1
Xhat2= PC1*Yv2+repmat(m,1,c)
%Reconstruct the data from P2
Xhat1= PC2*Yv1+repmat(m,1,c)

Ev1=X-Xhat1
Ev2=X-Xhat2
```

|

A.1.2: SVD Example

This code follows the steps of the numerical example in section 3.1.2.

```
clc
clear all;
X = [1      1      2      0      5      4      5      3
     3      2      3      3      4      5      5      4];

[r,c]=size(X);

```

```
% Compute the mean of the data matrix "The mean of each ...
    row" (Equation (10))
m=mean(X')';

% Subtract the mean from each image [Centering the data] ...
    (equation (11))
d=X-repmat(m,1,c);

% Construct the matrix Z
Z = d' / sqrt(c-1);

% Calculate SVD
[L,S,R] = svd(Z);

% Calculate the eigenvalues and eigenvectors
S = diag(S);
EigValues = S .* S;
EigenVectors=R;
```

|

A.2: MATLAB Codes for Our Experiments

In this section, the code of the three experiments are introduced.

A.2.1: Biometric Experiment

This code for the first experiment (Biometric experiment).

```
clc
clear all

% The total number of samples in each class is denoted ...
    by NSamples
NSamples=10;
% The number of classes
NClasses=40;

%Read the data
counter=1;
for i=1:NClasses
   Fold=['s' int2str(i)];
   for j=1:NSamples
        FiN=[Fold '\' int2str(j) '.pgm'];

       % Read the image file
       I=imread(FiN);

       % Resize the image to reduce the computational time
```

```
        I=imresize(I,[50,50]);

        % Each image is represented by one column
        data(counter,:)=I(:);

        % Y is the class label matrix
        Y(counter,1)=i;
        counter=counter+1;
    end
end

% Divide the samples into training and testing samples
% The number of training samples is denoted by ...
    NTrainingSamples
NTrainingSamples=5;

counter=1;
Training=[];   Testing=[];   TrLabels=[];   TestLabels=[];
for i=1:NClasses
    for j=1:NTrainingSamples
        Training(size(Training,1)+1,:)=data(counter,:) ;
        TrLabels(size(TrLabels,1)+1,1)=Y(counter,1);
        counter=counter+1;
    end
    for j=NTrainingSamples+1:NSamples
        Testing(size(Testing,1)+1,:)=data(counter,:) ;
        TestLabels(size(TestLabels,1)+1,:)=Y(counter,1);
        counter=counter+1;
    end
end
clear data Y I;
tic
% Calculate the PCA space, eigenvalues, and eigenvectors
data=Training';

[r,c]=size(data);
% Compute the mean of the data matrix "The mean of each ...
    row" (Equation (10))
m=mean(data')';
% Subtract the mean from each image (Equation (11))
d=data-repmat(m,1,c);

% Compute the covariance matrix (co) (Equation (11))
co=(1/c-1)*d*d';

% Compute the eigen values and eigen vectors of the ...
    covariance matrix
[eigvector,eigvl]=eig(co);

% Sort the eigen vectors according to the eigen values
eigvalue = diag(eigvl);
[junk, index] = sort(-eigvalue);
eigvalue = eigvalue(index);
eigvector = abs(eigvector(:, index));

% EigenvectorPer is the percentage of the selected ...
    eigenvectors
EigenvectorPer=50;
PCASpace=eigvector(:,1:EigenvectorPer*size(eigvector,2)/100);

```

```
% Project the training data on the PCA space.
TriningSpace=PCASpace'*d;
clear data;

% Calculate the lower and upper bounds of the class labels
% For example, the first testing sample is correctly ... classified when its
% label between 1 and 5, because the first testing ... sample belongs to the
% first class and the number of samples of the first ... class is 5. In other
% words, the first five samples of the training data ... belong to the first
% class.
counter=1;
for i=1:NSamples*NClasses-size(Training,1) % number of ... the tseting images
      l(i,:)=1+(counter-1)*(size(Training,1)/NClasses);
      h(i,:)=counter*(size(Training,1)/NClasses);
       if(rem(i,(NSamples-(size(Training,1)/NClasses)))==0)
           counter=counter+1;
       end
end

% Classification phase
% Each image is projected on the PCA space and then ... classified using
% minimum distance classifier.

CorrectyClassified_counter=0;
for i=1:size(Testing,1)
   TestingSample=Testing(i,:);
   TestingSample=TestingSample-m';
   TestingSample=PCASpace'*TestingSample';

   % Apply minimum distance classifier
   rr(i)=mindist_classifier_type_final(TestingSample,...
       TriningSpace,'Euclidean');
   if(rr(i)≥l(i,1) && rr(i)≤h(i,1))
       CorrectyClassified_counter=CorrectyClassified_counter+1;
   end
end

% The accuracy of the testing samples
CorrectyClassified_counter*100/size(Testing,1)
```

|

A.2.2: Image Compression Experiment

This code for the second experiment (image compression experiment).

```
1
```

```
clc
clear all;

data = imread('cameraman.png');

%If the image is RGB convert it to grayscale as follow
% data=rgb2gray(data);

%Each column represents on sample
[r,c]=size(data);

% Compute the mean of the data matrix "The mean of each ...
    row" (Equation (10))
m=mean(data')';

% Subtract the mean from each image [Centering the data] ...
    (Equation (11))
d=double(data)-repmat(m,1,c);

% Compute the covariance matrix (co) (Equation(12))
co=1 / (c-1)*d*d';

% Compute the eigen values and eigen vectors of the ...
    covariance matrix (Equation (2))
[eigvec,eigvl]=eig(co);

% Sort the eigenvectors according to the eigenvalues ...
    (Descending order)
eigvalue = diag(eigvl);
[junk, index] = sort(-eigvalue);
eigvalue = eigvalue(index);
eigvec = eigvec(:, index);



for i=10:10:100
    % Project the original data (the image) onto the PCA ...
        space
    % The whole PCA space or part of it can be used as ...
        folows as in
    % Equation (6)
    Compressed_Image=eigvec(:,1:(i/100)*size(eigvec,2))'*...
    double(d);

    % Reconstruct the image as in Equation (7)
    ReConstructed_Image= ...
        (eigvec(:,1:(i/100)*size(eigvec,2)))...
    *Compressed_Image;
    ReConstructed_Image=ReConstructed_Image+repmat(m,1,c);

    % Show the reconstructed image
    imshow(uint8(ReConstructed_Image))
    pause(0.5)

    % Calculate the error as in Equation (8)
    MSE=(1/(size(data,1)*size(data,2)))*...
    sum(sum(abs(ReConstructed_Image-double(data))))

    % Calculate the Compression Ratio (CR)= number of ...
        elments of the
```

```
    % compressed image / number of elements of the ...
        original image
    CR=numel(Compressed_Image)/numel(data)
end
```

|

A.2.3: Data Visualization Experiment

This code for the third experiment (data visualization experiment).

```
clc
clear all;

load iris_dataset;
data=irisInputs; % Data matrix, four attributes and 150 ...
    samples
[r,c]=size(data);

% Compute the mean of the data matrix "The mean of each ...
    row" (Equation (10))
m=mean(data')';

% Subtract the mean from each image [Centering the data] ...
    (Equation (11))
d=data-repmat(m,1,c);

% Compute the covariance matrix (co) (Equation (12))
co=1 / (c-1)*d*d';

% Compute the eigen values and eigen vectors of the ...
    covariance matrix (Equation (2))
[eigvector,eigvl]=eig(co);

% Project the original data on only two eigenvectors
Data_2D=eigvector(:,1:2)'*d;

% Project the original data on only three eigenvectors
Data_3D=eigvector(:,1:3)'*d;

% Reconstruction of the original data
% Two dimensional case
Res_2D= (eigvector(:,1:2))*Data_2D;
TotRes_2D=Res_2D+repmat(m,1,c);

% Two dimensional case
Res_3D= (eigvector(:,1:3))*Data_3D;
TotRes_3D=Res_3D+repmat(m,1,c);

% Calculate the error between the original data and the ...
    reconstructed data
% (Equation (8))
% (Two dimensional case)
```

```
MSE=(1/(size(data,1)*size(data,2)))*...
sum(sum(abs(TotRes_2D-double(data))))
% (Three dimensional case)
MSE=(1/(size(data,1)*size(data,2)))*...
sum(sum(abs(TotRes_3D-double(data))))

% Calculate the Robustness of the PCA space (Equation (9))
% (Two Dimensional case)
SumEigvale=diag(eigvl);
Weight_2D=sum(SumEigvale(1:2))/sum(SumEigvale)
% (Three Dimensional case)
Weight_3D=sum(SumEigvale(1:3))/sum(SumEigvale)

% Visualize the data in two dimensional space
% The first class (Setosa) in red, the second class ...
    (Versicolour) in blue, and the third class ...
    (Virginica) in
% green
figure(1),
plot(Data_2D(1,1:50),Data_2D(2,1:50),'rd'...
    ,'MarkerFaceColor','r'); hold on
plot(Data_2D(1,51:100),Data_2D(2,51:100),'bd'...
    ,'MarkerFaceColor','b'); hold on
plot(Data_2D(1,101:150),Data_2D(2,101:150),'gd'...
    ,'MarkerFaceColor','g')
xlabel('First Principal Component (PC1)')
ylabel('Second Principal Component (PC2)')
legend('Setosa','Versicolour','Virginica')

% Visualize the data in three dimensional space
% The first class (Setosa) in red, the second class ...
    (Versicolour) in blue, and the third class ...
    (Virginica) in
% green
figure(2),
scatter3(Data_3D(1,1:50),Data_3D(2,1:50),...
    Data_3D(3,1:50),'rd','MarkerFaceColor','r'); hold on
scatter3(Data_3D(1,51:100),Data_3D(2,51:100),...
    Data_3D(3,51:100),'bd','MarkerFaceColor','b'); hold on
scatter3(Data_3D(1,101:150),Data_3D(2,101:150),...
    Data_3D(3,101:150),'gd','MarkerFaceColor','g')
xlabel('First Principal Component (PC1)')
ylabel('Second Principal Component (PC2)')
zlabel('Third Principal Component (PC3)')
legend('Setosa','Versicolour','Virginica')
```

|

A.3: MATLAB Code for Calculating PCA

In this section, the codes for calculating the PCA space using covariance matrix and SVD methods are introduced.

A.3.1: Covariance Matrix Method

This code is used to calculate the PCA space using covariance matrix method and it follows the steps of Algorithm (1).

```
function [Newdata, PCASpace, EigValues]=PCACov(data)

[r,c]=size(data);

% Compute the mean of the data matrix "The mean of each ...
    row" (Equation (10))
m=mean(data')';

% Subtract the mean from each image [Centering the data] ...
    (Equation (11))
d=data-repmat(m,1,c);

% Compute the covariance matrix (co) (Equation (12))
co=1 / (c-1)*d*d';

% Compute the eigen values and eigen vectors of the ...
    covariance matrix
[eigvector,EigValues]=eig(co);

PCASpace=eigvector

% Project the original data on the PCA space
Newdata=PCASpace'*data;
```

|

A.3.2: SVD Method

This code is used to calculate the PCA space using SVD method and it follows the steps of Algorithm (2).

```
function [Newdata,PCASpace,EigValues]=PCASVD(data)
% PCASVD: This function constructs the PCA space using ...
    SVD method
% data - MxN matrix of input data
% M is the dimensions or features
% N is the number of samples or observations
% Newdata is the original data after projection onto the ...
    PCA space
% PCASpace is the space of PCA (i.e. eigenvectors)
% EigValues represent the eigenvalues

[r,c]=size(data);

```

```
% Compute the mean of the data matrix "The mean of each ... row" (Equation (10))
m=mean(data')';

% Subtract the mean from each image [Centering the data] ... (equation (11))
d=data-repmat(m,1,c);

% Construct the matrix Z
Z = d' / sqrt(c-1);

% Calculate SVD
[L,S,R] = svd(Z);

% Calculate the eigenvalues and eigenvectors
S = diag(S);
EigValues = S .* S;
PCASpace=R;

% Project the original data on the PCA space
Newdata=PCASpace'*d;
```