University of California
Los Angeles

An Overview of Non-Linear Kernel Functions for Solving the Human Face Recognition Problem

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Luis Antonio Sosa

2016

Abstract of the Thesis

An Overview of Non-Linear Kernel Functions for Solving the Human Face Recognition Problem

by

Luis Antonio Sosa

Master of Science in Statistics

University of California, Los Angeles, 2016

Professor Ying Nian Wu, Chair

Principal Component Analysis has been extensively used in the computer vision field as a method of capturing orthogonal axes of large variability in high-dimensional data sets. Computer vision scientists have come up with reconstructive models which capture the most distinguished features of a human face using Principal Component Analysis, known as Eigenfaces. Several papers have approached the problem of facial recognition using standard PCA, however very few provide a detailed comparison on the different non-linear kernels which can be used in place of the traditional linear approach. The aim of this paper is to introduce several non-linear kernel functions to the human recognition problem, by working with a set of radial basis kernels, a logarithmic kernel, a Cauchy kernel, and a polynomial kernel. We perform a model assessment for each kernel using a parameter tuning method which minimizes reconstruction error, and display reconstruction plots for each kernel method. We also capture influential physical features of the images in the high-dimensional space (the Eigenface) for each kernel and compare reconstructed and original images, by capturing the Frebenius (L2) norm between test and original image data.

The thesis of Luis Antonio Sosa is approved.

Nicolas Christou

Mark Handcock

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2016

To my father . . .

who—among so many other things

saw to it that we had a bed

and shelter to sleep to at night

To my sister . . .

who—protected me in childhood

and saw to it that I was never

bullied in secondary school

And last but not least the person

who influenced my life significantly

My mother . . .

who—among so many other things—

saw to it that I reach for the stars

Table of Contents

List of Figures

List of Tables

Acknowledgments

There are many people in the department of Statistics that I would like to acknowledge and thank personally. If I don't mention your name I apologize in advance because there really is a lot I'd like to thank for.

First and foremost, I want to thank the UCLA Statistics Department for giving me the opportunity to learn from the brightest minds in the field of Statistics. I would like to thank Mark Handcock for his comments and tips on academic writing– his feedback extremely helped with the structure of this paper. I'd like to thank Ying Nian Wu for his support of my topic, and for his courses in Applied Probability and Large Sampling Theory which really inspired and taught me a lot. I would like to thank Nicolas Christou for getting to know him while I was grading homework for his Intro to Mathematical Statistics class. I appreciated his realism, humor, and his support of my thesis topic as well.

I'd like to thank Akram Almohalwas for letting me teach his course Introduction to Statistics for Life Science Majors for two quarters, and also for speaking with him and getting to know him personally. He's a good man and sombody I have upmost respect for. I want to thank Robert Gould for being a friendly face I encountered on random occasions, Arash Amini for his friendly nature, and Chad Hazlett, Jessica Jingyi, Mahtash Esfandiari, Nikkhyl (Bryon) Aragam, Hongquan Xu, Qing Zhou, and Alan Yuille for their sincere, friendly and honest interactions with them whenever I encountered them. One professor I need to show my grattitude for after taking his course, is Song Chun Zhu. His Pattern, Recognition, and Machine Learning course was what gave me the idea for this topic. Most of the ideas about eigenfaces I learned from Song Chun Zhu.

I'd also like to thank the guys in the office, Chie Ryu, Jason Mesa, Verghese Nallengara, Enrique Reyes, Nannette Callo, and Glenda Jones for being people I can say hi to whenever I'm walking around the department. Jason was the man who knew all the upcoming rock groups, Chie was the guy who knew about every Korean BBQ spot that needed to get checked out, Verghese and Enrique were those dudes who could fix a printing/network issue

in the Master's Lounge or the Boelter lab in like 5 minutes, Nannette was the nice lady who you can go to for a wide variety of questions and she would help you in any creative way which makes her awesome and last but not least Glenda Jones, who was like that mother or aunt who'd crack down on me when I wasn't getting my act together.

The last two years have been a blessing and I would like to thank the department and the people I've met for the opportunity to write this thesis and learn graduate level Statistics. Thank you for everything.

CHAPTER 1

Introduction

The Face Recognition problem has been an area of fascination for many researchers in the past few decades. During this time, advances in high-level computing has allowed thorough and extensive contribution to the assortment of face recognition literature [1]. With a wide-array of applications such as biometric authentication, security, surveillance, and face detection, the promises of what Face Recognition can provide in our day to day lives introduces a remarkable benefit and interest for future computer vision collaborators, whether it be in the research realm, government, or private sector [6]. The face recognition problem can be defined as the following: Given an input face image and a database of known individuals and their face images, how can we verify the identity of the person in the input image [1]?

This question poses some underlying difficulties which arise when coming up with an accurate and precise face recognition algorithm. For instance, how can we most effectively capture a familiar face with least amount of noise and variability? To solve this problem, researchers in computer vision have categorized the variability of face images into two known distinct classes– intrinsic factors and extrinsic factors [7]. Intrinsic factors are defined as the physical characteristics and independent nature of the individual face. Gender, age, and physical characteristics of race, can be considered intrinsic factors. The other set of factors, extrinsic factors, cause the appearance of the face to alter via interaction with lighting. Examples of extrinsic factors are illumination, scale and imaging parameters (resolution, noise, focus, etc.). Both intrinsic and extrinsic factors can cause the image of the human face to vary greatly, which makes the human face recognition problem quite a task to achieve efficiently [1].

With this problem in mind, two researchers (Sirovich and Kirby) published a paper in 1987 discussing a new dimensionality reduction technique for human face images, using principal component analysis on image data [3]. The term Eigenpictures (or more commonly Eigenfaces) was termed for the first time, and gave concept to an image representing the most influential features of a human face. Principal components work by finding the most influential orthogonal components of a covariance matrix, representing a set of training images, and using these components to represent the image in a lower-dimensional subspace by a linear transformation [8]. Eigenfaces have shown much success in capturing influential areas of human faces, and have been a focus for many researchers looking to optimize and study the benefits of principal components to distinguish human faces.

In this thesis, I propose an alternative approach to the traditional principal components analysis by using a set of non-linear kernel methods. Kernel methods work by embedding data into a vector space by using a feature map (Gaussian, Laplacian, etc.) [9]. If the map is chosen suitably, complex relations can be simplified and easily detected, making the kernel approach a useful statistical learning method. There have been papers in the past discussing the use of kernel methods in principal component analysis of face data (Yang. M, 2002) [4]. But, in my paper I demonstrate a larger range of several non-linear kernel techniques. The kernel techniques I use are the exponential radial basis function, logarithmic, Cauchy, laplacian, Gaussian and polynomial (of degree 2).

The order of the paper is as follows: I first begin with a review of Principal Component Analysis and the theoretical framework behind Kernel Principal Component Analysis. Then I will display the notation and characteristics for the several feature spaces used in Kernel Principal Component Analysis. The following chapter will discuss my dataset, and for my analysis I will display the reconstruction errors for the different types of kernel methods. Reconstruction errors plots that I create show the L2-norm between the original test image and the reconstructed test image using my proposed algorithm. I also show images of the Eigenfaces for each kernel method and capture the features which are heavily influenced among the different feature spaces.

CHAPTER 2

Principal Component Analysis: An Eigenvalue Problem

2.1 Background

Principal Component Analysis is a multivariate statistical technique widely used in signal processing, exploratory data analysis, dimension reduction, clustering, etc [12]. The origins of Principal Component Analysis (or PCA) can be traced back to Karl Pearson in 1901 [13], and modernly instantiated by Hotelling in 1933 [10]. The idea behind PCA is to project a dataset into a new coordinate system by an orthogonal linear transformation. The transform is chosen in a manner to maximize the variance among several linearly orthogonal axes, known as principal components. The axes can be discovered by an eigen-decomposition of a data matrix, using the covariance matrix of a standardized data set [8]. Figure 2.1 shows a set of two correlated variables X and Y, and displays the axes of highest variance X1 and Y1 known as principal components. X1 is the first principal component, containing the largest variation of the data and Y1 is the second principal component containing the second-largest variation of the data. X1 and Y1 are both linearly uncorrelated, that is, they are orthogonal to each other.

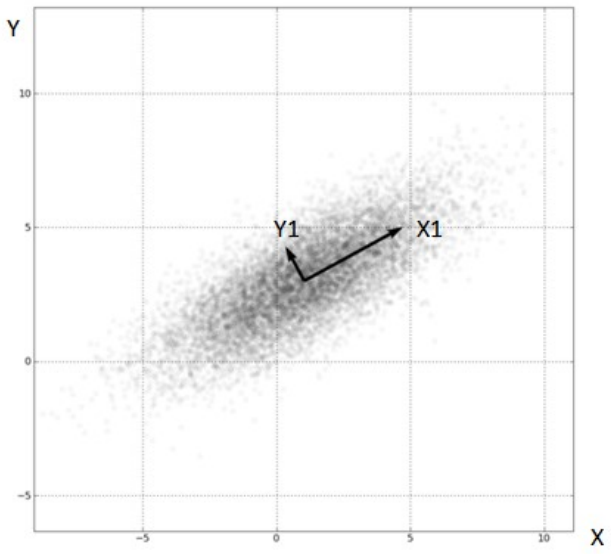


Figure 2.1: The two principal components of a set of observations in a 2 dimensional plane. X1 is largest principal component, and Y2 is second largest. Both axes are linearly orthogonal.

2.2 Methods and Algorithm

A summary of the PCA Approach is the following: First standardize the data matrix. Then obtain the eigenvectors and eigenvalues of your covariance matrix. Choose the k eigenvectors which correspond to the largest eigenvalues so that k is the new dimension of the feature space (where $k << m$, m being the dimension of the original feature space). Construct the projection matrix $\mathbf{W}$ using the best k eigenvectors. Finally transform the original dataset via $\mathbf{W}$ to create a new k dimensional feature subspace $\mathbf{Y}$ [10].

Let us first define the data matrix $\mathbf{X}$, containing m parameters and n observations:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix}$$

Then construct the mean vector $\vec{\boldsymbol{\mu}}$ by adding every column of $\mathbf{X}$ and dividing by number of observations n.

$$\vec{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x_i} = \frac{1}{n}(\boldsymbol{x_1} + \boldsymbol{x_2} + \cdots + \boldsymbol{x_n})$$

where $\boldsymbol{x_i}$ represents a vector containing the m features of observation i

$$\boldsymbol{x_i} = \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{mi} \end{bmatrix}$$

The mean vector $\vec{\boldsymbol{\mu}}$ is important for standardizing our data matrix $\mathbf{X}$. We subtract $\vec{\boldsymbol{\mu}}$ from $\mathbf{X}$ in order to ensure the columns in the data matrix are centered around $\vec{\boldsymbol{\mu}}$. Let $\mathbf{W}$ represent our data matrix with the mean $\vec{\boldsymbol{\mu}}$ subtracted.

$$\mathbf{W} = \begin{bmatrix} x_{11} - \mu_1 & x_{12} - \mu_1 & x_{13} - \mu_1 & \dots & x_{1n} - \mu_1 \\ x_{21} - \mu_2 & x_{22} - \mu_2 & x_{23} - \mu_2 & \dots & x_{2n} - \mu_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} - \mu_m & x_{m2} - \mu_m & x_{m3} - \mu_m & \dots & x_{mn} - \mu_m \end{bmatrix}$$

Solve for the covariance matrix $\mathbf{S}$, where n is the number of observations and $\mathbf{W}$ is our standardized data matrix.

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

When dealing with data which contains very large features (m), the matrix $\mathbf{W}\mathbf{W}^T$ ends up becoming very large (m x m). This makes solving for the eigenvalues and eigenvectors of the covariance matrix computationally expensive. A nice workaround can be achieved with the following proposition:

If $\mathbf{A}$ is any m x n matrix of real numbers, then the m x m matrix $\mathbf{A}\mathbf{A}^T$ and the n x n matrix $\mathbf{A}^T\mathbf{A}$ are both symmetric [11].

Now the question to ask is: how do the eigenvalues/eigenvectors for both symmetric matrices $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ relate? If we let $\vec{\boldsymbol{v}}$ be a nonzero eigenvector of $\mathbf{A}^T\mathbf{A}$ with eigenvalue $\lambda \neq 0$, we will have:

$$(\mathbf{A}^T\mathbf{A})\vec{\boldsymbol{v}} = \lambda\vec{\boldsymbol{v}}$$

Multiply both sides of the equation with matrix $\mathbf{A}$ and we obtain:

$$\mathbf{A}\mathbf{A}^T(\mathbf{A}\vec{\boldsymbol{v}}) = \lambda(\mathbf{A}\vec{\boldsymbol{v}})$$

The above equation gives a solid workaround for computing the eigenvectors and eigenvalues of our covariance matrix. If $\mathbf{A}$ is any m x n matrix, the matrices $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ share the same nonzero eigenvalues [11]. Considering that n is the number of observations and m is the number of features in data matrix $\mathbf{A}$ (where $n << m$), it would be much less computationally expensive to compute the eigenvectors of $\mathbf{A}^T\mathbf{A}$ (an n x n matrix) than $\mathbf{A}\mathbf{A}^T$ (an m x m).

Now that we have our eigenvectors and eigenvalues for symmetric matrix $\mathbf{A}^T\mathbf{A}$, we can choose the largest k eigenvectors which correspond to the largest eigenvalues of our covariance matrix. These k eigenvectors will act as a basis for the reconstruction of our original standardized data $\mathbf{W}$. In other terms, we project the original standardized data $\mathbf{W}$ into these k eigenvectors and transform them into the eigenspace $\mathbf{\Phi}$.

$$\mathbf{\Phi}_i = \vec{\boldsymbol{v}}_i^T * \mathbf{W}$$

The eigenspace is a linear combination of orthogonal vectors that act as a representation of our original data. This linear combination represents our data in a reduced dimension, making it easier to capture features of most influence. Figure 2.2 demonstrates a set of observations in a 2 dimensional axes being projected into a 1 dimensional linear subspace. This linear subspace represents the largest principal component in this data.

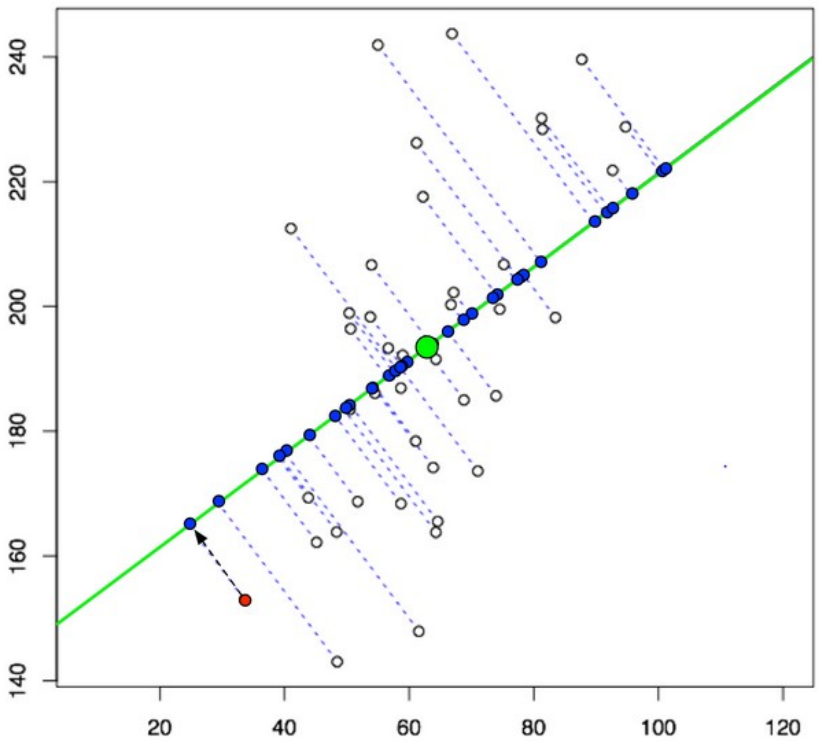


Figure 2.2: A set of observations in a 2 dimensional space being projected onto a linear subspace represented in green. White dots represent the original observation and blue dots represent the linear projection.

2.3 Eigenfaces for Face Recognition

Eigenfaces have proven to be a very effective face recognition technique. The eigenface is an image which displays the principal components of a set of face data, by displaying the highest variation in features (eyes, nose, mouth, eyebrows, etc.) [5]. Figure 2.3 displays an example of 5 eigenfaces, created from a collection of face images.

Figure 2.3: A representation of 5 individual eigenfaces, each representing the most influential features of a human face.

2.3.1 Classification

Mathematically, the eigenface represents the set of eigenvectors which characterizes the variation between a training set of face images, using the covariance or scatter matrix [5]. To perform face recognition, one would calculate the best k eigenfaces of a face dataset (eigenfaces with the largest eigenvalues), project a test image into the face space (the eigenspace for eigenfaces) and find the euclidean norm between an image already in the database and the test image (both images projected onto the face space) [5]. Ω represents the projection of the test image and Ω_i represents the projection of an image belonging to individual i.

$$\epsilon_i = \|(\Omega - \Omega_i)\|_2$$

If this Euclidean norm is underneath a certain threshold T, then the image is correctly classified as belonging to individual i (otherwise, the face does not match).

2.3.2 Image Reconstruction

Any image that is projected into the face space can be reconstructed using the linear combination displayed below, where u_j represents an eigenface and w_i the weight for each eigenface. Figure 2.4 displays an example of a reconstructed test image to the right, and the original test image to the left (taken from the FaceAccess Database produced by Cornell University [21]) . The reconstructed test image has its advantage over the original test image in that its components are shown in a reduced dimension, making calculations much simpler and less computationally expensive in classifying the face.

$$\mathbf{\Phi}_i = \sum_{j=1}^{k} w_i u_j$$

Figure 2.4: Test Image reconstructed using a linear combination of eigenfaces. The reconstructed image displays the features of the original image, in a reduced dimension. Image taken from FaceAccess produced by Cornell University [21].

2.3.3 Reconstruction Error Plots

Reconstruction error plots are an effective tool which measure the difference in pixels (or errors) between the original test image and the reconstructed test image. These errors are plotted as a measure of k, the amount of eigenfaces used to create the reconstructed

test image. Generally, as the number of eigenfaces (k) used in the image reconstruction increases, the more closely the reconstructed test image will resemble the original test image (thus producing smaller errors). The trade-off however is that an increase in the number of eigenfaces increases the amount of computations and calculations in the algorithm, with accuracy and effectiveness being limited. A good analogy to how reconstruction errors act as a function of k would be to that of screwing a nail to a wooden door. You can screw the nail into the door as much as you can, but there is a point where you dont need to screw anymore (the nail is firmly in).

Figure 2.5 displays a reconstruction error plot for the difference between the original image shown, and the reconstructed image at every value of k. Notice the image of the reconstructed image using 10 eigenfaces is not as precise as the reconstruction image using 100 eigenfaces. According to the plot, the reconstruction error levels off at about $k = 30$ and shows a slower rate of decrease in error as k increases (image taken from Indian Institute of Technology Kanpur [22]).

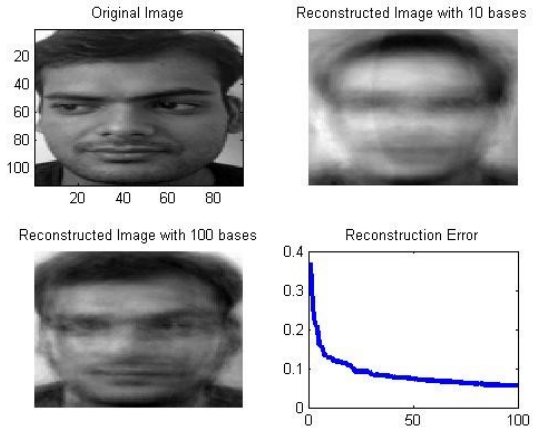


Figure 2.5: A representation of the reconstruction error for every value of k, using the test image shown. As k increases there is a dramatic reduction in reconstruction error, until a certain threshold is reached and k does not reduce error significantly. Image taken from Indian Institute of Technology Kanpur [22].

2.4 Kernel Principal Component Analysis

Kernel functions are a class of algorithms used in pattern recognition and statistical learning. The main idea behind using kernel functions is to find general types of relationships in higher dimensional spaces. Suppose we would like to map data nonlinearly into a feature space F [15]

$$\mathbf{\Phi} : \mathbf{R}^N \rightarrow F \, , \, \mathbf{x} \rightarrow \mathbf{X}$$

For the nonlinear transformation $\mathbf{\Phi}$, we can still perform PCA in feature space F by the use of kernel functions. Let us assume that our original data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ is mapped into feature space $\Phi_{\mathbf{x}_1}, \Phi_{\mathbf{x}_2}, \ldots, \Phi_{\mathbf{x}_n}$, and is centered $\sum_{k=1}^{n}(\Phi_{\mathbf{x}_k}) = 0$. A standard eigendecomposition of the covariance matrix [15]

$$C = \frac{1}{n}\sum_{j=1}^{n} \Phi_{\mathbf{x}_j}\Phi_{\mathbf{x}_j}^T$$

will leave us with the eigenvectors of the transformed data within F. These eigenvectors act as the principal components of our original data, mapped into the high dimensional space. Figure 2.6 represents a 2-Dimensional scatter of observations re-mapped using a Gaussian Radial Basis kernel. The two separate classes in the left image could not be separated linearly, however after applying the Gaussian RBF kernel map we have a linearly separable distinction between the two classes. This is what makes the use of kernel functions so effective, in that there are many classification and regression problems which are not linearly separable in the original space of the inputs $\mathbf{x}$.

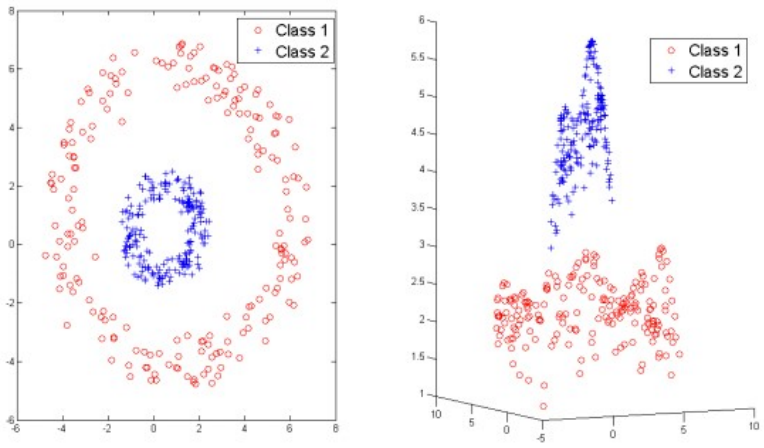


Figure 2.6: Two classes which are linearly inseperable in one feature space, transformed into a new feature space using the Gaussian Radial Basis Function. In the new feature space, the two classes are linearly seperable.

To demonstrate this non-linear transformation, let us consider the following mapping $\mathbf{\Phi}$ for an example $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D$. We would like to convert our original variables into the feature space:

$$\mathbf{\Phi} : \mathbf{x}_1^2, \mathbf{x}_2^2, \ldots, \mathbf{x}_D^2, \mathbf{x}_1\mathbf{x}_2, \mathbf{x}_1\mathbf{x}_3, \ldots, \mathbf{x}_1\mathbf{x}_D, \ldots, \mathbf{x}_{D-1}\mathbf{x}_D$$

The feature space above is an example of a quadratic mapping, which is a taking a set of observations into a higher dimensional space using different combinations of polynomials. Notice the amount of variables in the new feature space is larger than in our original data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D$. This introduces a new set of problems with computing the variables of our new feature space because these spaces have much higher dimensions, and to perform PCA one would have to find the covariance matrix of these newly mapped variables. This can be very computationally expensive, so a workaround is introduced in the following section.

2.4.1 The Kernel Method (or Kernel Trick)

The function $\mathbf{\Phi}$, which is needed to form the covariance matrix, does not need to be calculated explicitly. Since we are mapping $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ to a higher dimension, the actual calcuation of $\mathbf{\Phi_{x_1}}, \mathbf{\Phi_{x_2}}, \ldots, \mathbf{\Phi_{x_n}}$ can be time consuming and computationally expensive. A workaround to avoid explicitly mapping into the higher dimensional space was introduced in 1992 by Vapnik, Guyon and Bosner, and is known as the kernel method, or kernel trick [16] (I like to view this technqiue as a medium or driver as opposed to a trick, simply because there are no tricks involved). The idea is to calculate the inner product between all pairs of data in the feature space, in order to maximize margins in a dual space (this technique is also applied to Support Vector Machines) [16]. The inner product matrix, called the kernel matrix, is shown below.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)^T$$

With this kernel matrix, it is possible to calculate the eigenvectors and eigenvalues under various feature spaces. Let $\boldsymbol{\alpha}$ represent the nonzero eigenvectors of kernel matrix $\mathbf{K}$ and λ its eigenvalues.

$$\mathbf{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$$

An eigendecomposition of the kernel matrix will obtain the principal components of the data in the high-dimensional feature space F. If the data used to construct $\mathbf{K}$ is not centered around 0 (not standardized), the equation below can be used in place of $\mathbf{K}$. This equation calculates the Gramian matrix, where $\mathbf{1_n}$ represents the n x n matrix of $1/n$ and $\mathbf{K}$ represents the kernel matrix above [17]. The Gramian matrix is represented as $\widetilde{\boldsymbol{K}}$.

$$\widetilde{\boldsymbol{K}} = \mathbf{K} - \mathbf{1_n}\mathbf{K} - \mathbf{K}\mathbf{1_n} + \mathbf{1_n}\mathbf{K}\mathbf{1_n}$$

Now this paper will focus on the different kernel functions used in this thesis.

2.4.2 Kernel Functions

2.4.2.1 Polynomial Kernel

The polynomial kernel maps input data into a feature space over polynomials of the original variables. This kernel function is an example of a non-stationary kernel. Parameters which can be adjusted are the slope α, the constant c, and the degree d [9].

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\alpha \mathbf{x}_i^T \mathbf{x}_j + c)^d$$

2.4.2.2 Gaussian Kernel

The Gaussian kernel is a radial basis function kernel. This kernel maps input data into an infinite dimensional Hilbert space. The adjustable σ parameter plays a very important role in the performance of the kernel. If overestimated, the kernel will behave almost linearly and if underestimated the function will lack regularization and be highly sensitive to noise [9].

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$$

2.4.2.3 Exponential Kernel

The Exponential kernel is another radial basis function kernel, very similar to that of the Gaussian RBF. Instead of calculating the squared Euclidean norm however, it calculates the standard Euclidean norm [9].

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma^2})$$

2.4.2.4 Laplacian Kernel

The Laplacian Kernel is less sensitive to changes in the sigma parameter and is another example of a radial basis function kernel. It also uses the Euclidean norm.

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma})$$

2.4.2.5 Logarithmic Kernel

The Logarithmic kernel belongs to a class of conditionally positive definite kernels. This kernel was introduced in 2005 and has proven to work very well with SVM-based image recognition [18]. What makes this kernel unique is how well it performs in comparison to its classical positive definite kernel counterparts, making research in this kernel very opportunistic.

$$K(\mathbf{x}_i, \mathbf{x}_j) = -log(\|\mathbf{x}_i - \mathbf{x}_j\|^d + 1)$$

2.4.2.6 Cauchy Kernel

The Cauchy kernel was introduced in 2010, and worked very well in a research focusing on Spam classification (email filtering). It is a long-tailed kernel and can be used to give long-range influence and sensitivity over the high dimension space. This is another interesting kernel to use because of how recently it has been introduced.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + \frac{\|\mathbf{x}_i - \mathbf{x}_j^2\|}{\sigma^2}}$$

CHAPTER 3

The Chicago Face Database

Now let's dicscuss the image data used in this thesis. The Chicago Face Database was developed by researchers from the University of Chicago in 2015 [20]. These images are high-resolution color, standardized photographs of male and female faces, with varying ethnicity, between the ages of 17 and 65. Figure 3.1 displays three of the images contained in the Chicago Face Database, in this case our focus primarily on Caucasian males.

Figure 3.1: Three random images chosen from the Chicago Face Database [20]. Each image shares similar backgrounds, facial expression, and shirt color for standardization.

Originally, the images are 2444 x 1718 pixels in .jpeg format. All images share a white background for standardization and displays the subjects top of head, and their shoulders. For the purpose of this research, the images used for analysis have been standardized even further by reducing variation and noise. The color pixels have been converted to grey scale using IrfanView, a photo editing software. In addition, the original 2444 x 1718 pixel images have been reduced to a size of 750 x 750 pixels for less expensive computations in our analysis. The original .jpeg images have also been converted to .bitmap as well, in an effort to reduce individual file size of the images (.jpeg files sizes are traditionally larger than .bitmap file

size).

The analysis focuses on the area of the face above the eyebrows, down to the top of the subjects chin. This method reduced the noise in image pixel accounted by hair, shoulder, background, etc. Figure 4.2 displays the cropped, edited image data used in our analysis.

Figure 3.2: Images from the Chicago Face Database that have been cropped even further for the purpose of our analysis. The crop area is greyscaled and focuses on the more intrinsic features of the face (eyebrows, nose, mouth, etc).

The next sections of this paper will discuss the analysis and results of the linear and non-linear Kernel PCA methods.

CHAPTER 4

Assessment of Linear and non-linear Kernel PCA methods on Face Data

Since the Chicago Face Database contains a large variety of different faces, the images were categorized by its intrinsic factors, mainly gender. This is done in order for our models to focus on particular features of a face without extraneous variation.

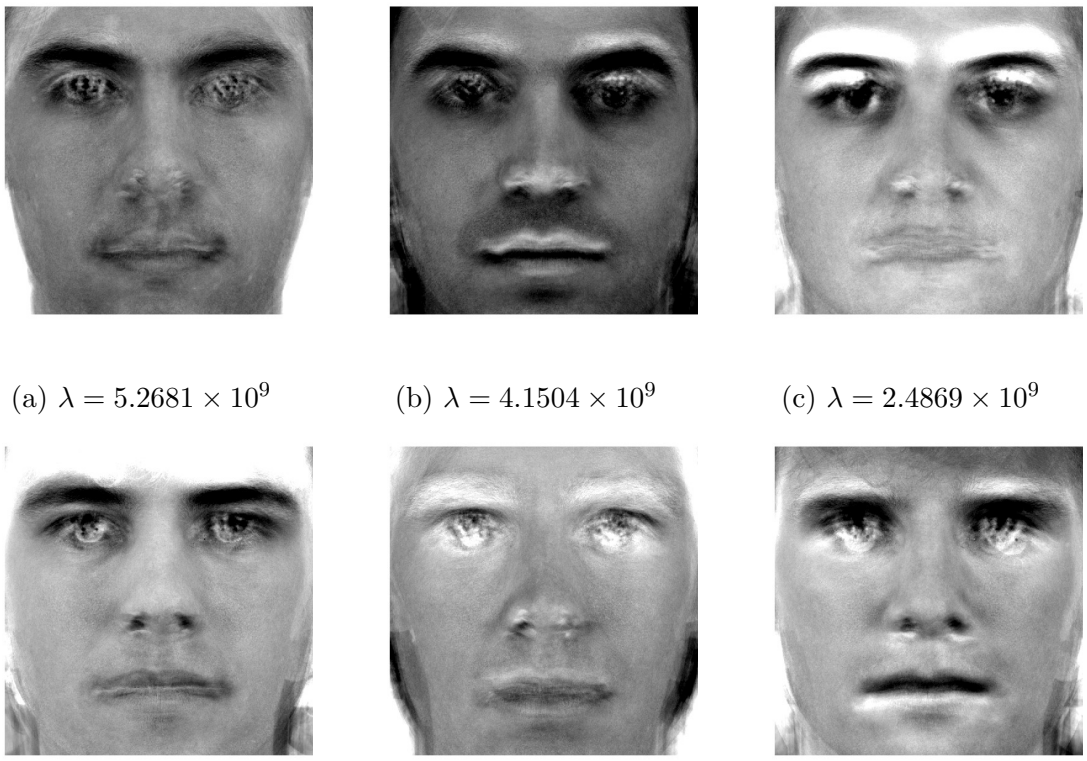
Given the dataset of 93 individual males, 84 images were used for training the model and 9 were used for testing. Each image represents a vector of 562,500 pixels, because every bitmap image is 750 x 750 pixels. The original data matrix $\mathbf{A}$ will therefore be a size of 562,500 x 84 pixels, since there are 84 images for training the model and 562,500 pixels per image. Adding up the 84 columns in data matrix $\mathbf{A}$, and dividing by the number of columns leaves us with an average representation for each of the 562,500 features. Therefore, we are left with a mean vector $\boldsymbol{\mu}$ which represents the average face. The average face of the 84 training images is shown in Figure 4.1.

Figure 4.1: The mean face of the 84 training images used in our analysis. This mean face captures the "average" features among the 84 individual faces.

With this average face, we can subtract this face from every column in data matrix $\mathbf{A}$ in order to standardize our data. Lets call this standardized matrix $\mathbf{B}$. The covariance matrix $\mathbf{B}\mathbf{B}^T$ will be a size of 562,500 x 562,500 pixels, which is far too large to compute (316,406,250,000 pixels of information). Calculating $\mathbf{B}^T\mathbf{B}$ on other hand, leaves us with a covariance matrix of 84 x 84 pixels (which is 7,056 pixels of information, compared to 316,406,250,000).

4.1 Standard PCA - Linear Approach

Performing the eigendecomposition of covariance matrix $\mathbf{B}^T\mathbf{B}$ leaves us with the eigenvectors and eigenvalues associated with our 84 training images, our principal components. Here are the six eigenfaces with the largest eigenvalues (largest principal components).



(a) $\lambda = 5.2681 \times 10^9$ (b) $\lambda = 4.1504 \times 10^9$ (c) $\lambda = 2.4869 \times 10^9$

(d) $\lambda = 1.7473 \times 10^9$ (e) $\lambda = 1.4476 \times 10^9$ (f) $\lambda = 1.2211 \times 10^9$

Figure 4.2: The six eigenvectors with the largest eigenfaces using the standard linear Prinipal Component Analysis.

These six eigenfaces contain the largest variation in features among the face training set, each eigenface representing its own linearly orthogonal axis. A linear combination of these faces can be used to reconstruct any of the nine test images we have chosen. In Figure 4.16 (in back of thesis), we show the nine test images and their reconstructed images using 20 of the best eigenfaces.

With the reconstructed images in figure 4.16, we calculate the L2-norm between the reconstructed test image and its original form. Table 4.1 displays the set of L2-norms between all nine test images using linear principal component analysis. Image 8 appears to be the most accurate reconstruction amongst all 9 images, with image 3 and image 5 coming to a

close second and third respectively. Image 7 did not perform as well in reconstruction as did the other images. We will compare the individual image reconstructions using the other kernel methods in the following sections.

L2-Norm: Reconstruction - Original	
Test Image	L2-Norm
1	1.2808×10^4
2	1.3582×10^4
3	1.1342×10^4
4	1.4926×10^4
5	1.1628×10^4
6	1.1819×10^4
7	1.8407×10^4
8	1.1030×10^4
9	1.2749×10^4

Table 4.1: A table displaying the L2-norm between the original test image, and reconstructed image using standard linear principal component analysis.

A reconstruction plot is created which shows the average reconstruction error amongst all 9 test images for every value of k used. The average error displayed in the reconstruction shows the arithmetic mean error for all values of k, which will be used as a measure of how well each kernel model performs. The reconstruction error plot is show in figure 4.3 below. Notice how the error levels off after about $k = 20$ (20 eigenfaces are used in the reconstruction test data).

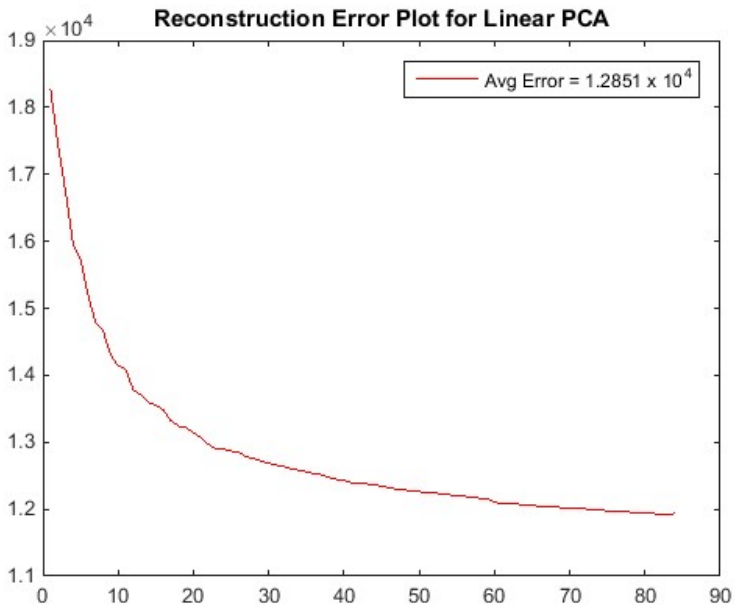


Figure 4.3: Reconstruction error plot using the standard linear PCA.

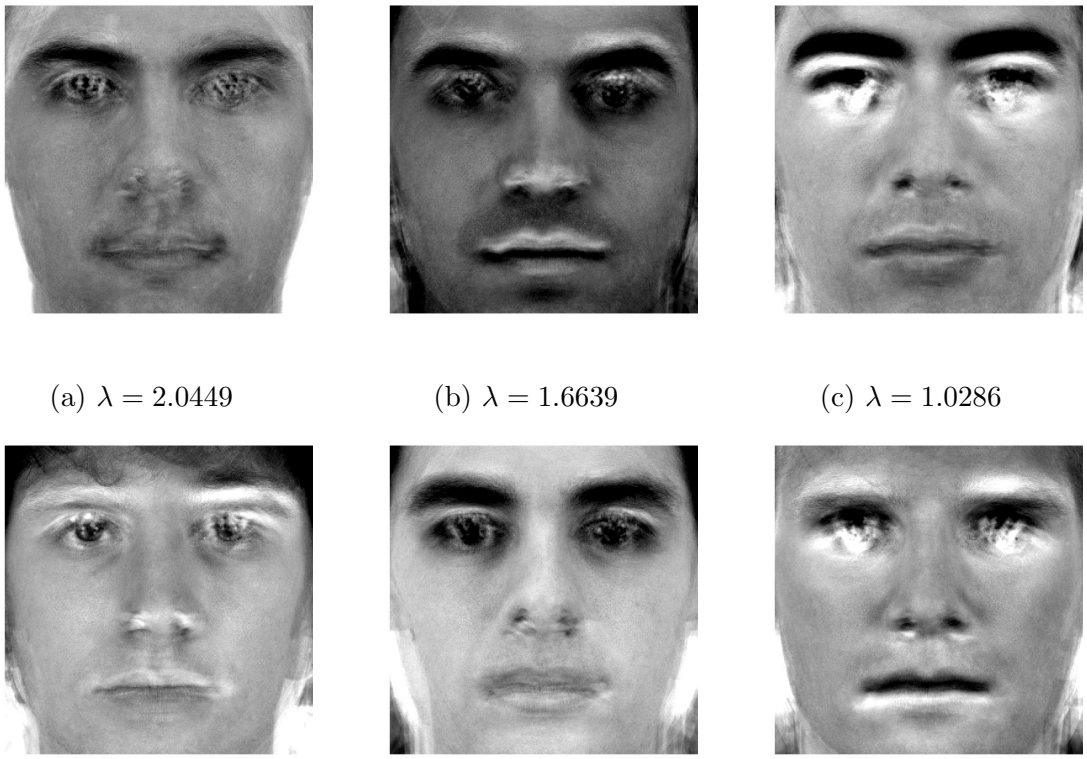
4.2 Gaussian Kernel

The first kernel in our analysis is the Gaussian Radial Basis Function. In order to compute the Squared Euclidean norm between every pair of images, the identity equation:

$$\|\mathbf{X} - \mathbf{Y}\|^2 = \mathbf{X} \cdot \mathbf{X} + \mathbf{Y} \cdot \mathbf{Y} - \mathbf{2}(\mathbf{X} \cdot \mathbf{Y})$$

was used to create the kernel matrix. The next step was tuning the Gaussian kernel, which was determined by performing a cross validation on a set of σ values. The σ value which had the lowest reconstruction error was approximately 2×10^9, at an error level of 1.3172×10^4. The eigenfaces for the Gaussian kernel are shown in figure 4.4. The first two eigenfaces appear similar to that of the linear PCA, but the third and fourth are completely different features. Notice the inclusion of head hair in Gaussian Eigenface 4 as well as a new type of face for Gaussian Eigenface 3. The shift into the higher dimension using the

Gaussian kernel displays some new information that would not have been shown using the Linear PCA.



(a) $\lambda = 2.0449$ (b) $\lambda = 1.6639$ (c) $\lambda = 1.0286$

(d) $\lambda = 0.7126$ (e) $\lambda = 0.6136$ (f) $\lambda = 0.5108$

Figure 4.4: The six eigenfaces with the largest eigenvalues using the Gaussian kernel.

Image reconstructions are shown in the back of this thesis in Figure 4.17. The reconstructed image with least amount of error was again image 8, with 3 and 5 playing a second and third best role again. Image 7 also plays the worst role as well, similar to the linear PCA assessment. Table 4.2 shows the rest of the L2-norms amongst the 9 test images using the Gaussian kernel.

L2-Norm: Reconstruction - Original	
Test Image	L2-Norm
1	1.2874 x 10^4
2	1.3574 x 10^4
3	1.1404 x 10^4
4	1.4877 x 10^4
5	1.1606 x 10^4
6	1.1838 x 10^4
7	1.8493 x 10^4
8	1.1097 x 10^4
9	1.2790 x 10^4

Table 4.2: A table displaying the L2-norm between the original test image, and reconstructed image using the Gaussian kernel.

The reconstruction plot for the Gaussian is very similar to the original linear PCA reconstruction plot. The overall average error on the other hand, appears to be slightly larger than in the linear form. This is very interesting to note, because the σ parameter in the Gaussian kernel plays a role in the flexibility or rigidity of the model. A high σ can make a Gaussian kernel appear linear, so if our lowest reconstruction errors occur with our Gaussian kernel appearing linear, then that can give us some useful information about the structure of our data. Overall, Gaussian performs well as a feature space for image data. In this instance linear PCA performs slightly better.

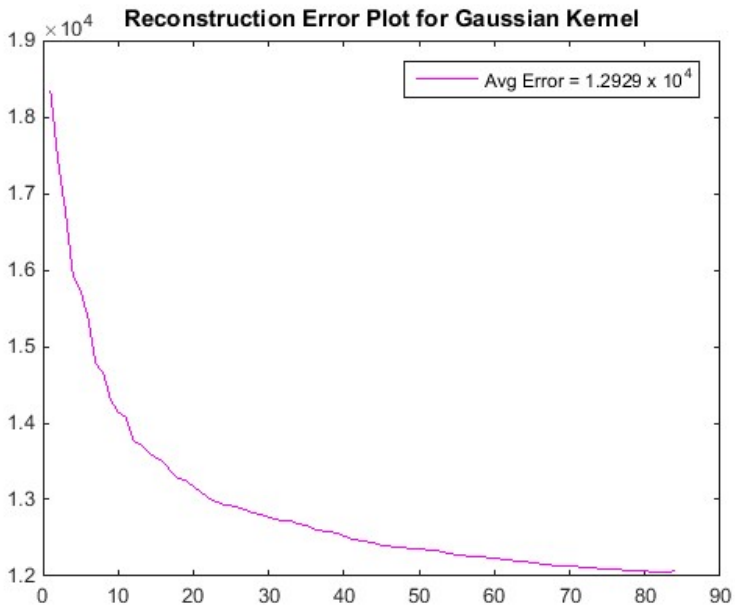
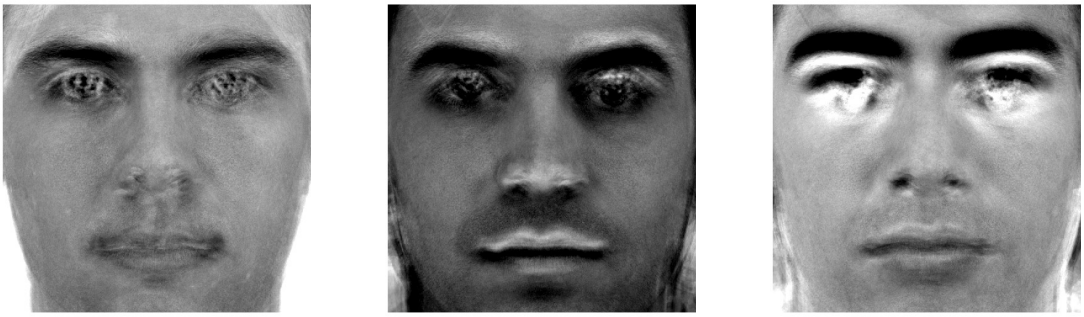


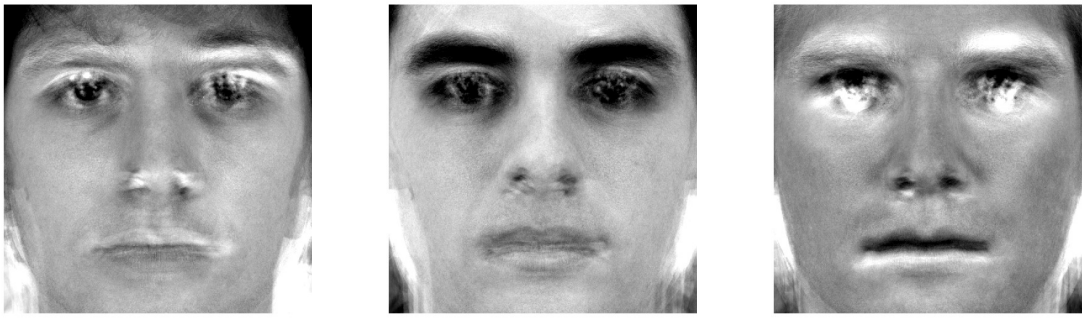
Figure 4.5: Reconstruction error plot using the Gaussian kernel.

4.3 Exponential Kernel

The Exponential kernel is another radial basis kernel which uses the Euclidean norm, in contrast to the Euclidean squared norm calculated in the Gaussian kernel. For this function, the optimal tuning parameter sigma was set to 1×10^{10}. In the construction of the eigenfaces, the top 6 faces for the Exponential kernel appeared the exact same as the top 6 faces for the Gaussian Kernel (which makes sense since they are both Radial Basis Functions). Below are the Eigenfaces shown:



(a) $\lambda = 0.002483$ (b) $\lambda = 0.002107$ (c) $\lambda = 0.001405$



(d) $\lambda = 0.001034$ (e) $\lambda = 0.000909$ (f) $\lambda = 0.000791$

Figure 4.6: The six eigenfaces with the largest eigenvalues using the exponential kernel.

The reconstruction errors for the test images (Figure 4.18) show a bit of an increase in comparison to the Gaussian Kernel. Not surprisingly, the test images with the lowest reconstruction errors were again images 8, 3 and 5 (first, second, and third lowest respectively), and the worst reconstruction (highest error) was test image 7 again. The outcomes of the test images were similar to the Gaussian kernel, with some interesting changes. For instance, test images 4 and 5 actually obtained lower reconstruction errors for the exponential kernel in comparison to the Gaussian. This makes it apparent that the exponential kernel performs better on some images, than the Gaussian kernel.

L2-Norm: Reconstruction - Original	
Test Image	L2-Norm
1	1.3187 x 10^4
2	1.3631 x 10^4
3	1.1570 x 10^4
4	1.4863 x 10^4
5	1.1604 x 10^4
6	1.1865 x 10^4
7	1.8806 x 10^4
8	1.1198 x 10^4
9	1.2840 x 10^4

Table 4.3: A table displaying the L2-norm between the original test image, and reconstructed image using the exponential kernel.

In the reconstruction plot shown, the trend is similar to that of the Gaussian kernel and has a slightly larger average error of 1.3156×10^4. This leads us to the conclusion that the Gaussian kernel performs better than the Exponential kernel overall, but as we have noticed before, the exponential kernel has the potential to perform better on some images than the Gaussian. Overall it is apparent these two kernels are very similar to one another since they are in the same family of radial basis functions and have their benefits in face reconstruction. The linear PCA algorithm overall, still outperforms both Gaussian and Exponential kernels.

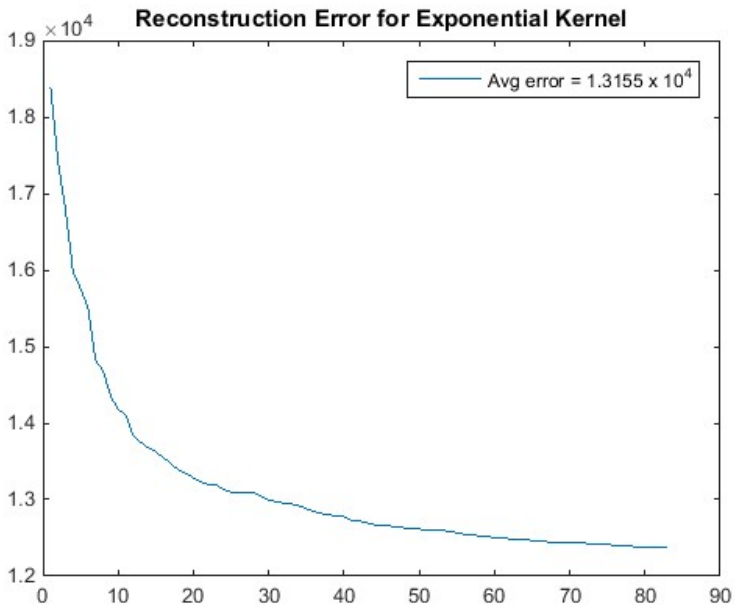


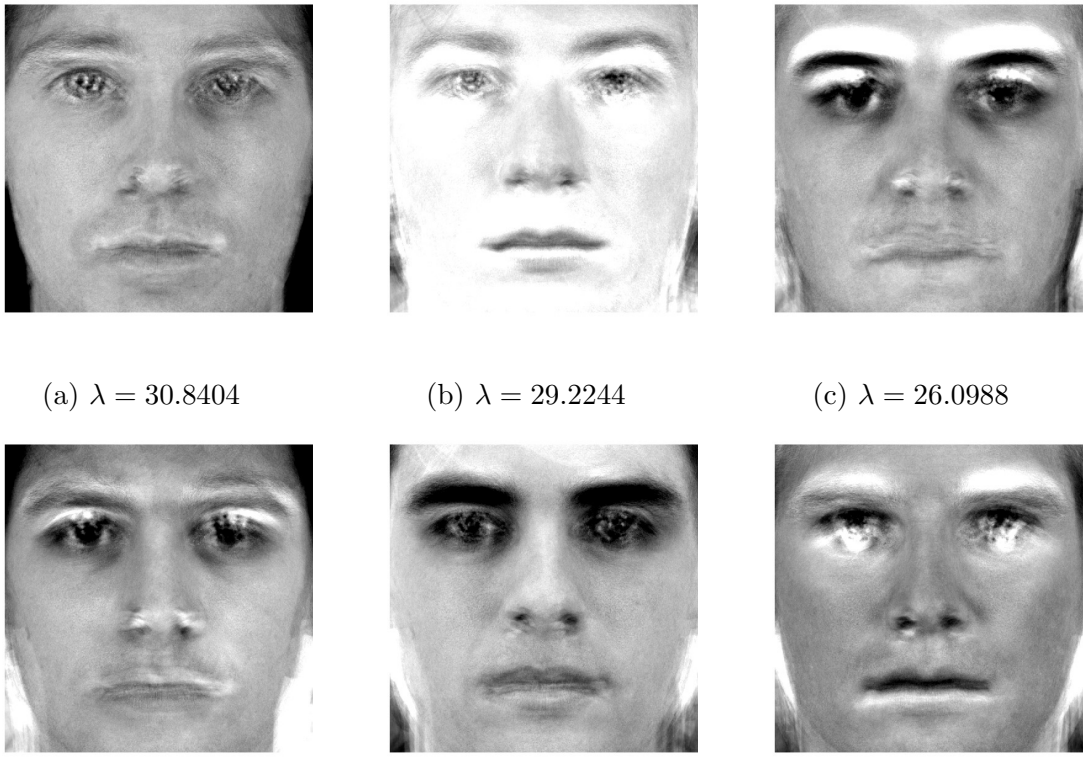
Figure 4.7: Reconstruction error plot using the exponential kernel.

4.4 Logarithmic Kernel

The logarithmic kernel is a function which uses the L2-norm between every pair of image, raises it to the power of d, and adds a constant 1 in the transformation to its feature space. To tune this kernel was a simple matter of choosing which power of d to select. The ranges of d which historically work well with this kernel are $0 < d \leq 2$ (18). The degree which minimizea the reconstruction error occurs at $d = 2$, which means the Euclidean Squared norm is what works best here.

The eigenfaces created under the logarithmic kernel were very different from the ones created using the radial basis functions and linear approach. Eigenfaces 1 and 2 are completely new images tracking more or less the shape of the face in contrast to its features. Notice how the eyebrows in eigenface 1 have been removed in comparison to previous eigenfaces. Eigenfaces 3, 4, 5, and 6 display similar characteristics to that of the previous eigenfaces, with minor details omitted (notice the hair in eigenface 4 is gone, as opposed to the Expo-

nential/Gaussian Kernel).



(a) $\lambda = 30.8404$ (b) $\lambda = 29.2244$ (c) $\lambda = 26.0988$

(d) $\lambda = 24.3896$ (e) $\lambda = 23.2913$ (f) $\lambda = 22.8283$

Figure 4.8: The six eigenfaces with the largest eigenvalues using the logarithmic kernel.

The reconstruction errors among all test images show an increase in magnitude, meaning the logarithmic kernel did not perform as well as the Gaussian, Exponential or Linear PCA. The best image reconstructions occurred with test image 8, 3, and 6 (notice how the logarithmic kernel performs better in image 6 than in image 5, which is different from previous kernels). With that being said, it is clear that the logarithmic kernel also puts emphasis on different features than the ones use in the previous kernels, so it has the potential to work well with very unique images, but overall not as effective generally as the radial basis and linear kernels.

L2-Norm: Reconstruction - Original	
Test Image	L2-Norm
1	1.4654 x 10^4
2	1.4261 x 10^4
3	1.1855 x 10^4
4	1.5204 x 10^4
5	1.2095 x 10^4
6	1.2021 x 10^4
7	2.0458 x 10^4
8	1.1372 x 10^4
9	1.3113 x 10^4

Table 4.4: A table displaying the L2-norm between the original test image, and reconstructed image using the logarithmic kernel.

The reconstruction plot below for the logarithmic kernel shows an error of 1.3938×10^4, which is significantly greater than the previous errors. The shape of the plot is interesting as well, with a quick sharp decrease in early k values but as k increases the error stays consistent at a level of about 1.4×10^4 (starting at $k = 10$). This leads me to believe that the logarithmic function works very well in instances when you want to use low amount of k. In practice, if you are focusing on computations that are less expensive and need a quick solution (with k being small), the logarithmic kernel can provide some great use.

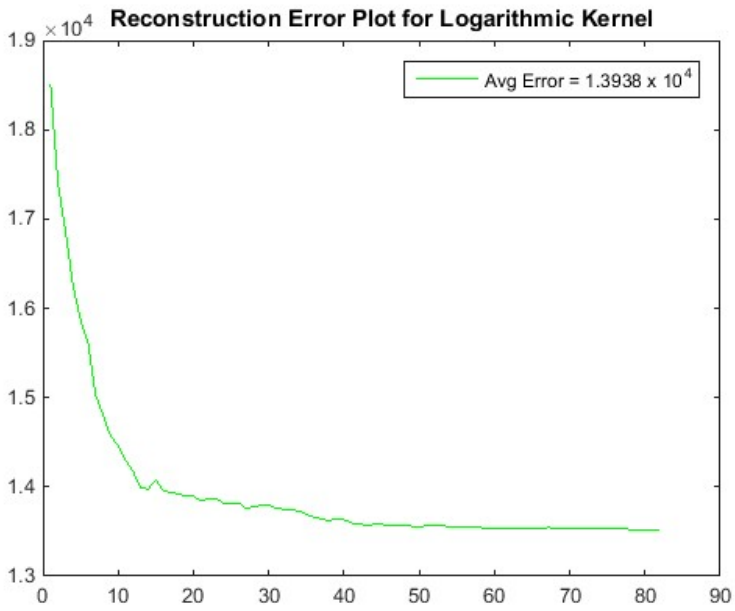
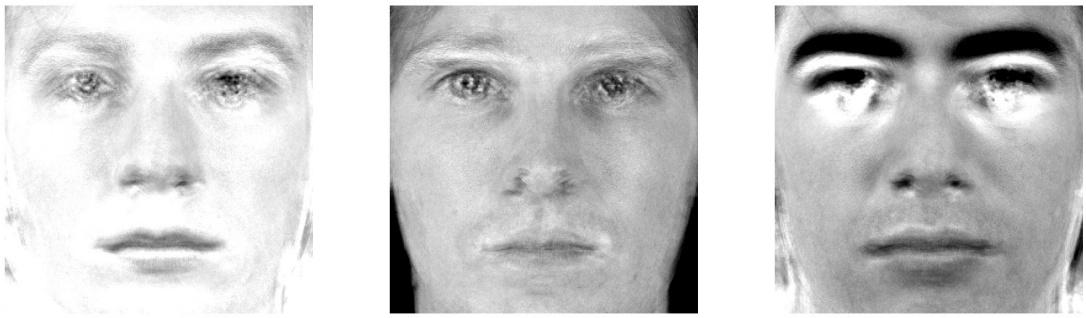


Figure 4.9: Reconstruction error plot using the logarithmic kernel.

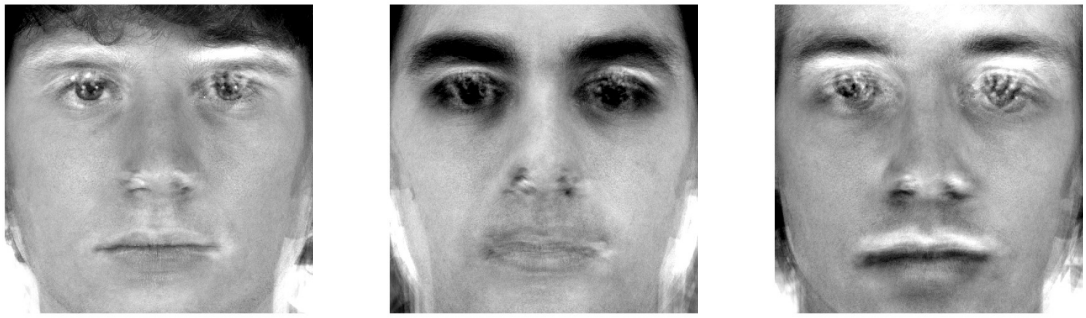
4.5 Cauchy Kernel

The Cauchy Kernel is a kernel which works well on spam filtering, but has not been extensively used in image processing. This makes for a great opportunity in applying the Cauchy kernel within this thesis. With the function, comes a tuning parameter σ and the Euclidean squared norm between pairs of images. The optimal σ which created the least reconstruction error was at $\sigma = 1.3 \times 10^{10}$. The eigenfaces are shown below, which appear to be a combination of the faces from logarithmic and radial basis kernels. Eigenfaces 1 and 2 are similar to that of the logarithmic kernel, but have now been swapped in terms of magnitude of eigenvalue. It is important to note that the eigenvalues for the Cauchy kernel include negative numbers (meaning this kernel matrix is not conditionally positive/positive definite). The sign of the eigenvalue displays the direction of the vector and has nothing to do with strength. So the eigenvalues with the largest absolute value, or magnitude have been chosen as the most influential eigenfaces. Eigenface 4 appears very similar to that of

the Gaussian and Exponential kernel, and there is an inclusion of a new face for Eigenface 6.



(a) $\lambda = 0.0371$ (b) $\lambda = -0.0266$ (c) $\lambda = -0.0132$



(d) $\lambda = -0.0092$ (e) $\lambda = -0.0075$ (f) $\lambda = -0.0067$

Figure 4.10: The six eigenfaces with the largest eigenvalues using the Cauchy kernel. Note: Emphasis was focused on eigenvalues with the largest magnitude, regardless of sign.

The reconstruction errors for the Cauchy Kernel (Figure 4.20) showed some interesting results, with images 8, 3, and 5 again being the top three reconstructed images. Images 7 and 8 actually had a much lower reconstruction error than in the logarithmic and exponential kernels, which shows the usefulness of the Cauchy kernel for certain images. In the overall assessment of reconstruction however, Cauchy does not perform as well as the Gaussian, Exponential, or Logarithmic kernels.

L2-Norm: Reconstruction - Original	
Test Image	L2-Norm
1	1.4210 x 10^4
2	1.3854 x 10^4
3	1.1476 x 10^4
4	1.5028 x 10^4
5	1.1761 x 10^4
6	1.1879 x 10^4
7	1.8675 x 10^4
8	1.1111 x 10^4
9	1.2915 x 10^4

Table 4.5: A table displaying the L2-norm between the original test image, and reconstructed image using the Cauchy kernel.

The reconstruction plot for the Cauchy kernel displays a steady decrease as k increases (as opposed to the logarithmic which shows a fast rate of decrease in early stages of k and then levels off at a consistent error level). The overall average error for this kernel function is 1.42×10^4, which has been the highest so far. It seems though Cauchy has the potential to perform better at reconstruction in individual images, the other kernels have shown a much better performance in the overall image reconstruction process.

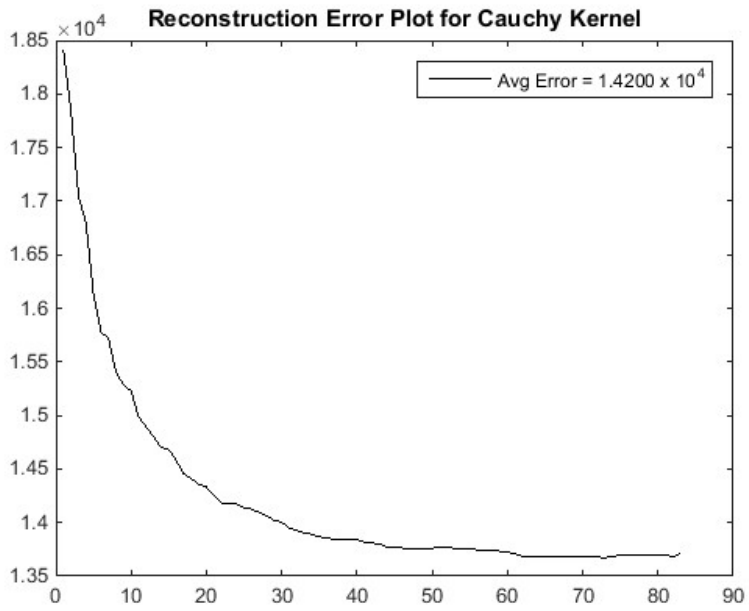
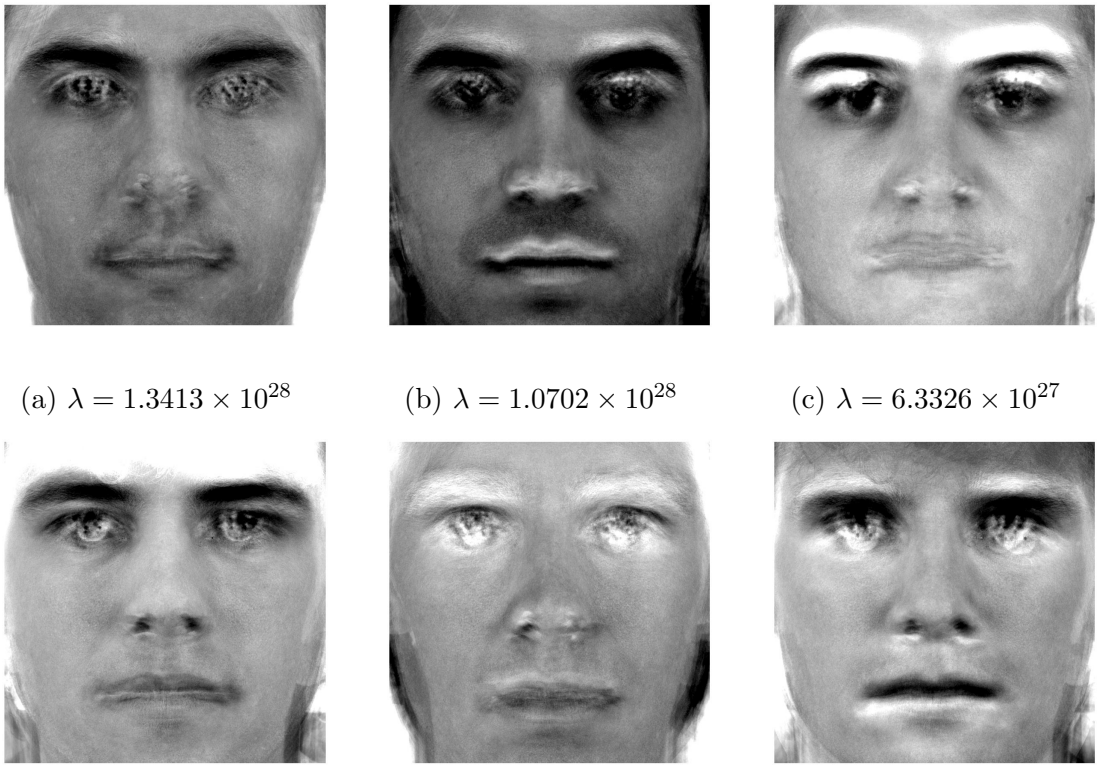


Figure 4.11: Reconstruction error plot using the Cauchy kernel.

4.6 Polynomial of Degree 2

The polynomial kernel is a popular kernel used in many statistical learning applications. For this analysis, the polynomial kernel of degree 2 has been chosen. The parameters which can be modified for this kernel are the scaling constant $\boldsymbol{\alpha}$ (multiplied with the dot product $\mathbf{X} \cdot \mathbf{Y}$) and the additional constant $\mathbf{c}$ (which is added to the product of $\boldsymbol{\alpha}\mathbf{X} \cdot \mathbf{Y}$). In the case of degree 2, the kernel was optimized using $\boldsymbol{\alpha} = 1000$ and $\mathbf{c} = 1.0 \times 10^{15}$ Below are the eigenfaces for the Polynomial kernel, which appear very simlar to that of linear PCA.



(a) $\lambda = 1.3413 \times 10^{28}$ (b) $\lambda = 1.0702 \times 10^{28}$ (c) $\lambda = 6.3326 \times 10^{27}$

(d) $\lambda = 4.4479 \times 10^{27}$ (e) $\lambda = 3.6853 \times 10^{27}$ (f) $\lambda = 3.1073 \times 10^{27}$

Figure 4.12: The six eigenfaces with the largest eigenvalues using the Polynomial Kernel (degree 2).

The L2-norms for the polynomial kernel of degree 2 were some of the best (lowest errors) in this analysis thus far. The polynomial kernel beat the linear PCA model in reconstruction error for test images 1, 2, and 7. The top three test images are the same as in the other kernels (test image 8, 3, and 5), but what really stood out is how similar the reconstructions between the linear PCA and polynomial kernel PCA are. With minor modifications and tweaks, the polynomial performs just as good, and potentially better than the linear PCA. The linear PCA does have better performance in individual test image 3. In general the polynomial kernel of degree 2 tops the other non-linear kernels used in this analysis.

L2-Norm: Original - Reconstruction	
Test Image	L2-Norm
1	1.2807 x 10^4
2	1.3581 x 10^4
3	1.1344 x 10^4
4	1.4926 x 10^4
5	1.1628 x 10^4
6	1.1819 x 10^4
7	1.8404 x 10^4
8	1.1030 x 10^4
9	1.2749 x 10^4

Table 4.6: A table displaying the L2-norm between the original test image, and reconstructed image using the Polynomial kernel (degree 2).

The reconstruction plot of the polynomial kernel with degree 2 is shown in Figure 4.13. Interestingly, the average error of the polynormial kernel is 1.2852×10^4, which is almost exactly alike as the linear PCA average error. This makes it clear that the Polynomial kernel, though uses a higher dimensional space, performs in a very similar manner to that of Linear PCA. The benefit of using the Polynomial PCA is because there is the potential to perform better than Linear PCA in individual images. So based on what we've seen thus far, in terms of non-linear kernel methods the polynomial kernel of degree 2 will work the best.

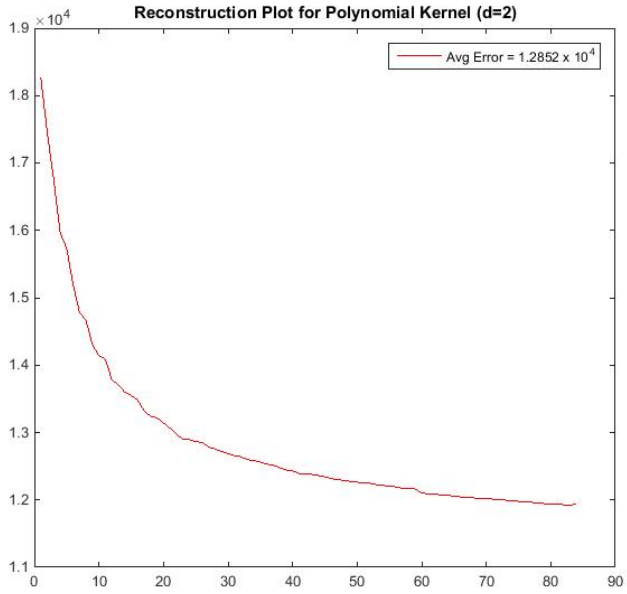


Figure 4.13: Reconstruction error plot using the Polynomial Kernel (degree 2).

4.7 Laplacian Kernel

The last kernel function we will apply to our image data is the Laplacian kernel, which is another radial basis kernel in the same family as the Gaussian and Exponential. Unlike its Exponential and Gaussian counterparts however, the norm of every pair of image is not divided by $\mathbf{2}\boldsymbol{\sigma}^2$, instead the Laplacian kernel is divided by $\boldsymbol{\sigma}$ (refer back to section 2.4.2 for the difference between the kernel functions). Dividing simply by $\boldsymbol{\sigma}$ results in a tuning parameter which is much less sensitive to changes in the $\boldsymbol{\sigma}$ value, which makes this radial basis function an interesting kernel to use for various regression and classification problems.

The eigenfaces for the Laplacian kernel are displayed below, and because of the difference in $\boldsymbol{\sigma}$ versus $\mathbf{2}\boldsymbol{\sigma}^2$, the eigenfaces appear different from the Exponential and Gaussian kernels. The first eigenface looks exactly the same as in the Exponential, linear, polynomial and Gaussian kernels, and interestingly enough the second eigenface appears exactly like the

Cauchy kernel's first eigenface (or the logarithmic kernel's second eigenface). Eigenface 3 is the exact same eigenface as the logarithmic kernel's third eigenface, and eigenfaces 4, 5, and 6 appear exactly the same as the polynomial (with degree 2) and linear eigenfaces.

(a) $\lambda = -14.8308$ (b) $\lambda = -11.7736$ (c) $\lambda = -7.2074$

(d) $\lambda = -5.1744$ (e) $\lambda = -4.3412$ (f) $\lambda = -3.7208$

Figure 4.14: The six eigenfaces with the largest eigenvalues using the Laplacian kernel. Note: Emphasis was focused on eigenvalues with the largest magnitude, regardless of sign.

In general, the Laplacian eigenfaces represent a combination of several of the kernels used in this thesis. The reconstructions errors for each test image, shown in Table 4.7, demonstrate a significant decrease in error similar to that of the linear and polynomial kernel of degree 2. Test images 2, 4, 5, and 7 have lower reconstructions errors than both the linear and polynomial kernel PCA of degree 2. This is signficant because, the polynomial kernel and linear PCA have performed the best in this analysis thus far. Comparing the Laplacian

kernel to the other radial basis functions display an improvement with no question. The Laplacian kernel works better than the Gaussian and the Exponential kernel, making this the best radial basis function applied in this thesis.

L2-Norm: Reconstruction - Original	
Test Image	L2-Norm
1	1.2817 x 10^4
2	1.3567 x 10^4
3	1.1352 x 10^4
4	1.4909 x 10^4
5	1.1612 x 10^4
6	1.1817 x 10^4
7	1.8383 x 10^4
8	1.1043 x 10^4
9	1.2752 x 10^4

Table 4.7: A table displaying the L2-norm between the original test image, and reconstructed image using the Laplacian kernel.

The reconstruction plot for the Laplacian kernel shown in Figure 4.15 displays a smooth improvement as k increases in comparision to the logarithmic and Cauchy kernels. With the average error calculated at 1.2854×10^4, this kernel function performs better than the Gaussian Kernel and the Exponential kernel. The error for the Laplacian Kernel is higher than in the linear and polynomial PCA, but overall the error is extremely close considering the polynomial kernel error was 1.2852×10^4 and the linear error was 1.2851×10^4. With regards to individual test image, it is clear the Laplacian outperforms its radial basis counterparts and can potentially do better than the linear and polynomial kernel, making this an excellent choice for kernel PCA in face reconstruction. Overall, the Laplacian kernel is the best radial basis function used in this thesis.

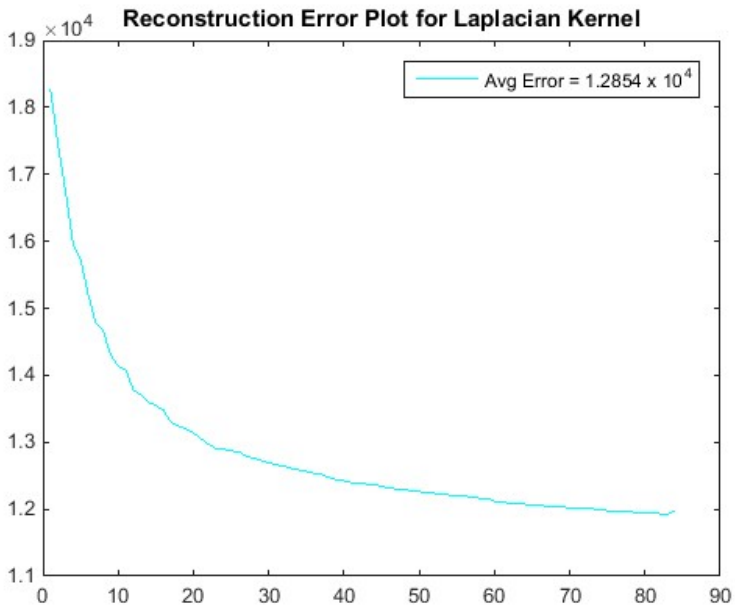


Figure 4.15: Reconstruction error plot using the Laplacian kernel.

In Figure 4.16, we display the reconstruction plots for the Laplacian, Logarithmic, Exponential, Cauchy, and Gaussian Kernels side by side. The radial basis functions (Laplacian, Gaussian, and Exponential) work the best while the Logarithmic and Cauchy kernels have larger errors in comparison. The Logarithmic kernel performs very well in early values of k. The Linear and Polynomial Kernel outperforms all these kernels (the reconstruction error for linear, polynomial, and laplacian appear to be the exact same, denoted as red in the plot below).

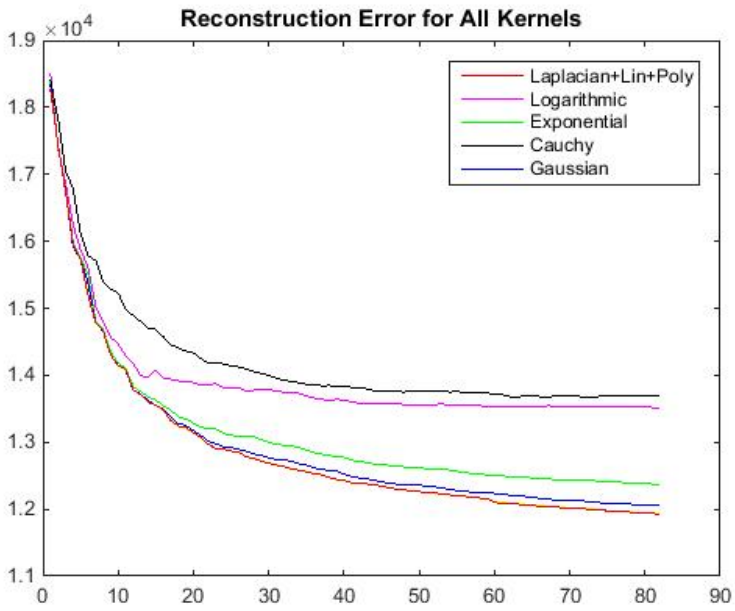


Figure 4.16: Reconstruction plot for every kernel used in this analysis. Note: Linear, Polynomial, and Laplacian Kernel all share similar error values denoted in red.

CHAPTER 5

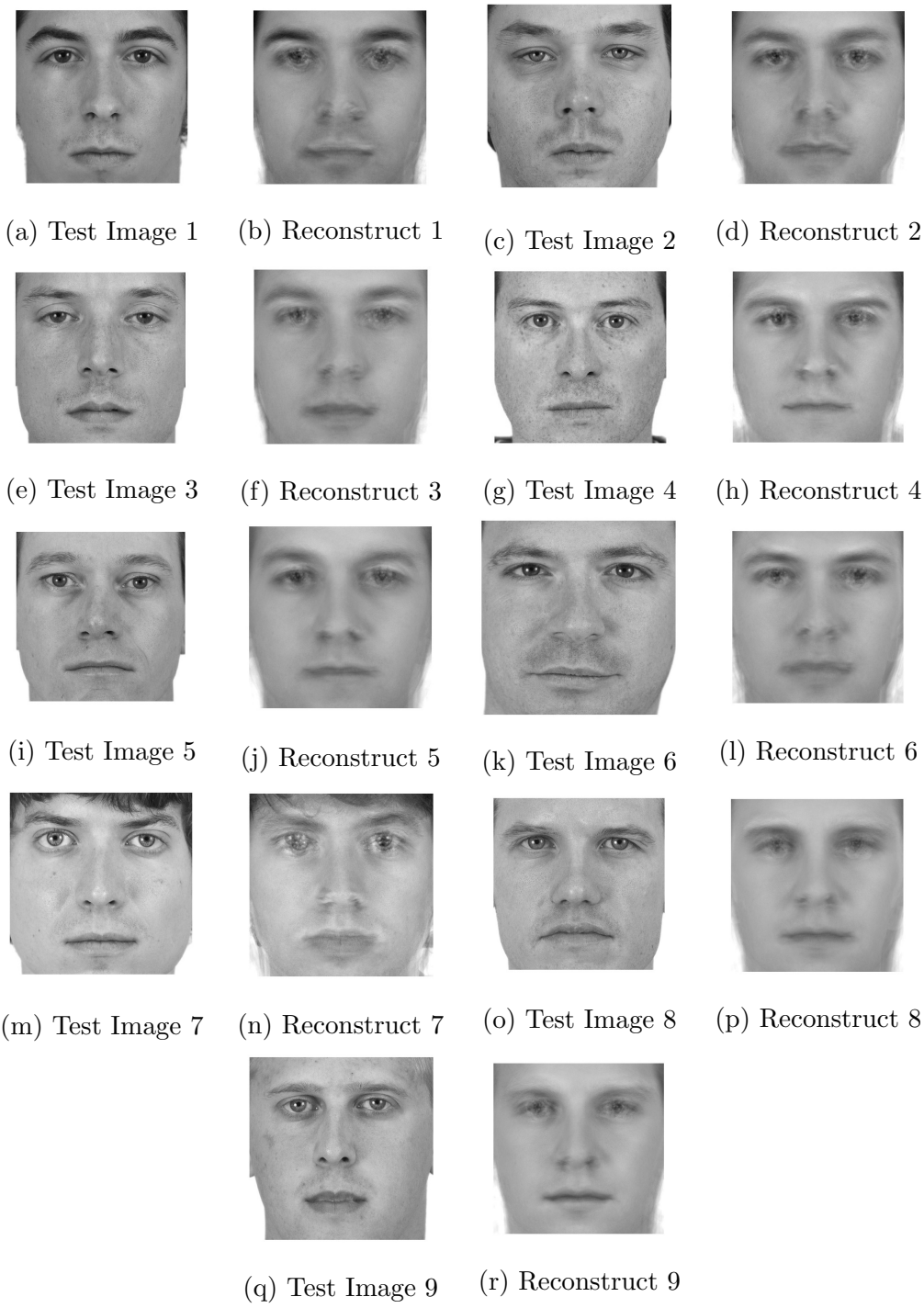
Conclusion and Future Work

The assessment of non-linear kernel techniques which have been applied to face image data has brought some very interesting findings in this thesis. First and foremost, it is clear to say that non-linear kernels have proven to be an effective alternative to linear principal component analysis. The linear PCA has the ability to capture significant features and performed the best in face reconstruction, while the kernel functions provide a much larger flexibility in capturing influential features of variation. Secondly, the type of kernel selected for modeling the face images depends on the type of requirements needed to solve a certain problem. For example, if somone needs to create an algorithm which performs a face reconstruction using only $k = 5$ eigenfaces, the logarithmic function would be of great use (because of how well it works with low levels of k). In general what this thesis has shown is the manner in which each non-linear kernel captures the features of influence, and how each function works in reconstruction for individual images, and also for every value of k.

The results of our analysis is as follows. The linear PCA, Polynomial kernel of degree 2, and the Laplacian kernel have all performed the best in our analysis. In several instances, the polynomial kernel of degree 2 and the Laplacian kernel perform better than linear PCA. The Gaussian and Exponential kernels are runner-ups after the linear PCA, Polynomial kernel, and Laplacian kernel. The Gaussian and Exponential kernel perform better for certain test images than the linear, polynomial, and laplacian kernels, but in the overall assessment does not have as low of reconstruction errors as the linear, laplacian, or polynomial kernels. The Laplacian, Exponential, and Gaussian kernels are all of the same radial basis family, and Laplacian performs the best in this family of kernels. The logarithmic kernel performs

similarly to the other kernels for low levels of k, but does not achieve low reconstruction error rates in the long-term, in comparison to its radial basis and polynomial counterparts. The Cauchy distribution performed the worst in this assessment, making this kernel not a very good choice for image-based data (as far as this thesis is concerned).

These findings provide a basis for future work with regards to kernel PCA and human face reconstruction. Though this thesis has focused on a set of male images, the same assessment can be performed on female faces in order to confirm this paper's finding. Furthermore, the amount of pixel data used in this analysis (750 x 750) can be a contributing factor to the performance of the different kernels, so a modification of pixel size with regard to image data can be looked onto further. Other face databases which exist currently contain images of people expressing different emotions (happiness, anger, etc.), along with images of faces seperated by other intrinsic factors (a person's race can affect intrinsic features of a face). These future analyses can provide some overall conclusions between the performance of these non-linear kernel methods and the large variety of face data that exists in the world today, making the possibilities of future work very opportunistic.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

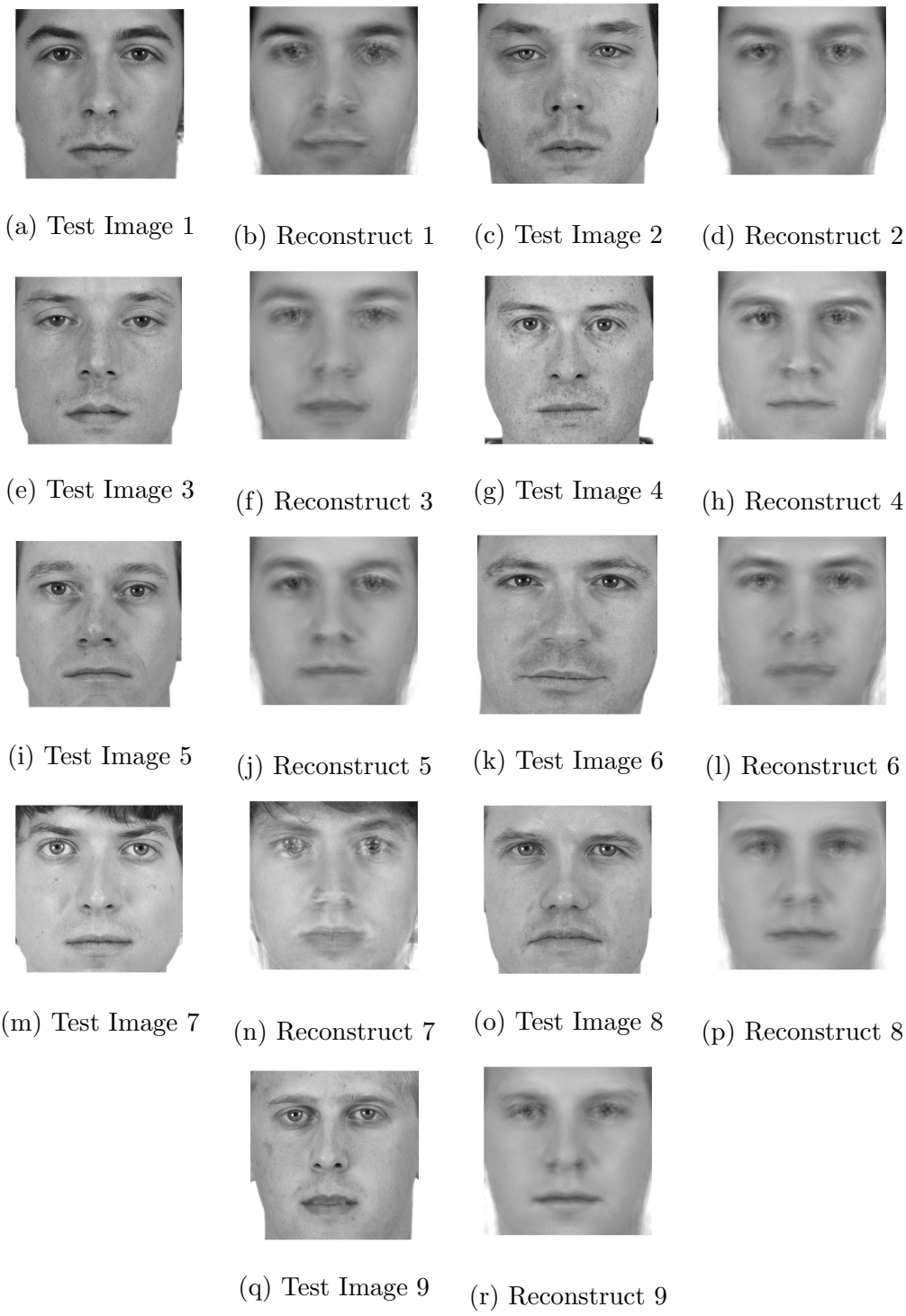
(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.1: Comparison of the nine original test images and reconstruction images using standard linear principal component analysis.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

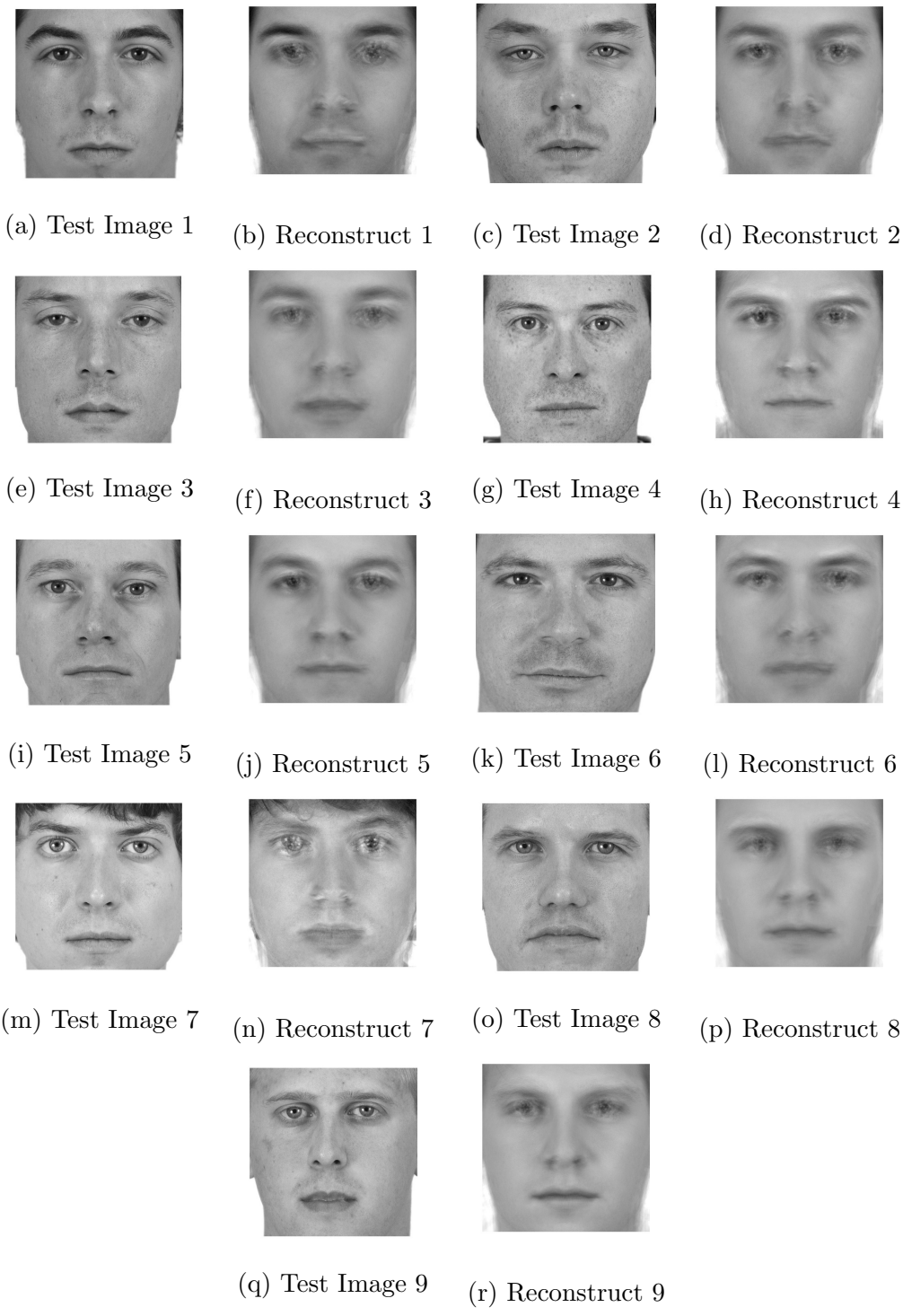
(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.2: Comparison of the nine original test images and reconstruction images using the Gaussian kernel.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

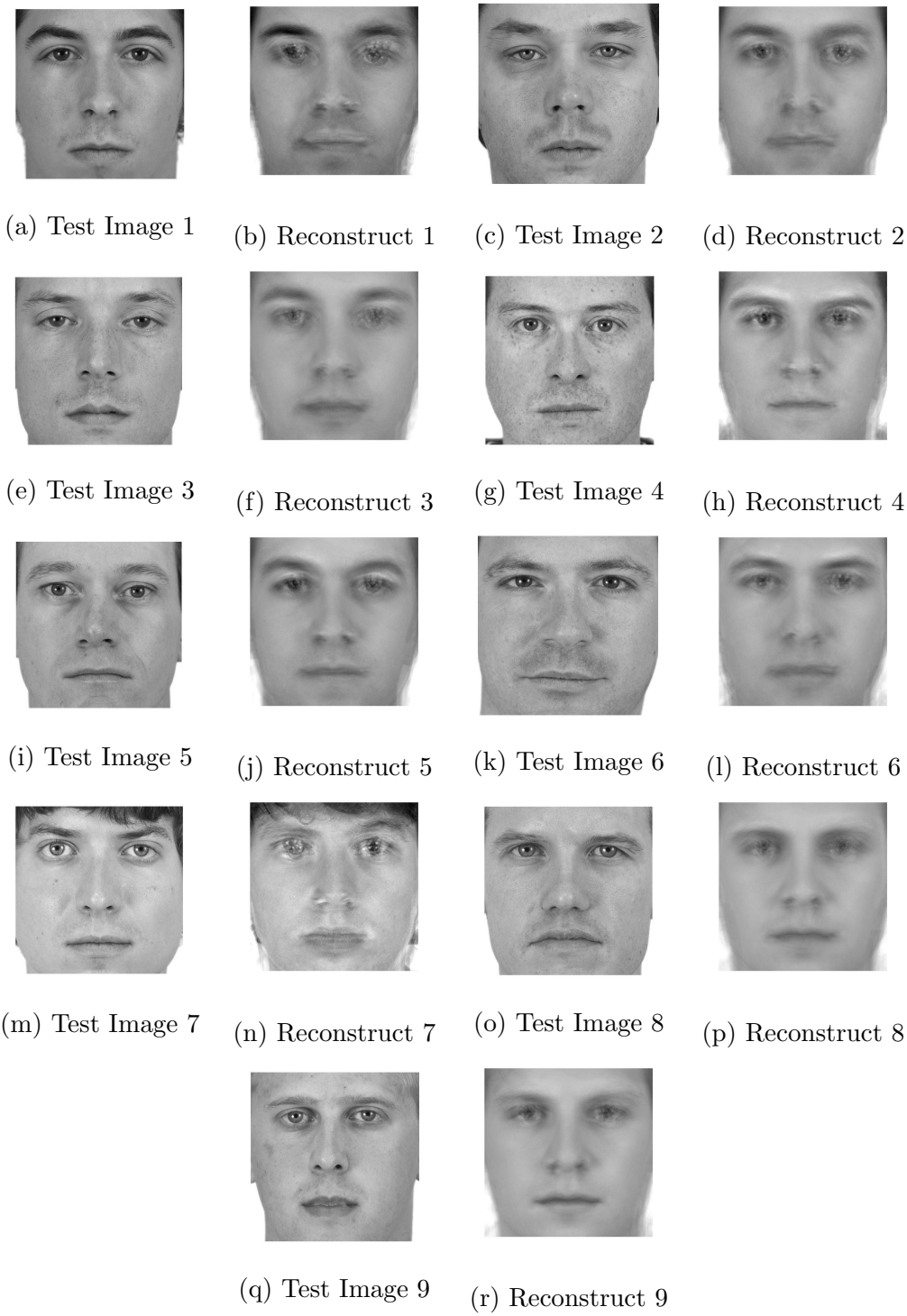
(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.3: Comparison of the nine original test images and reconstruction images using the exponential kernel.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

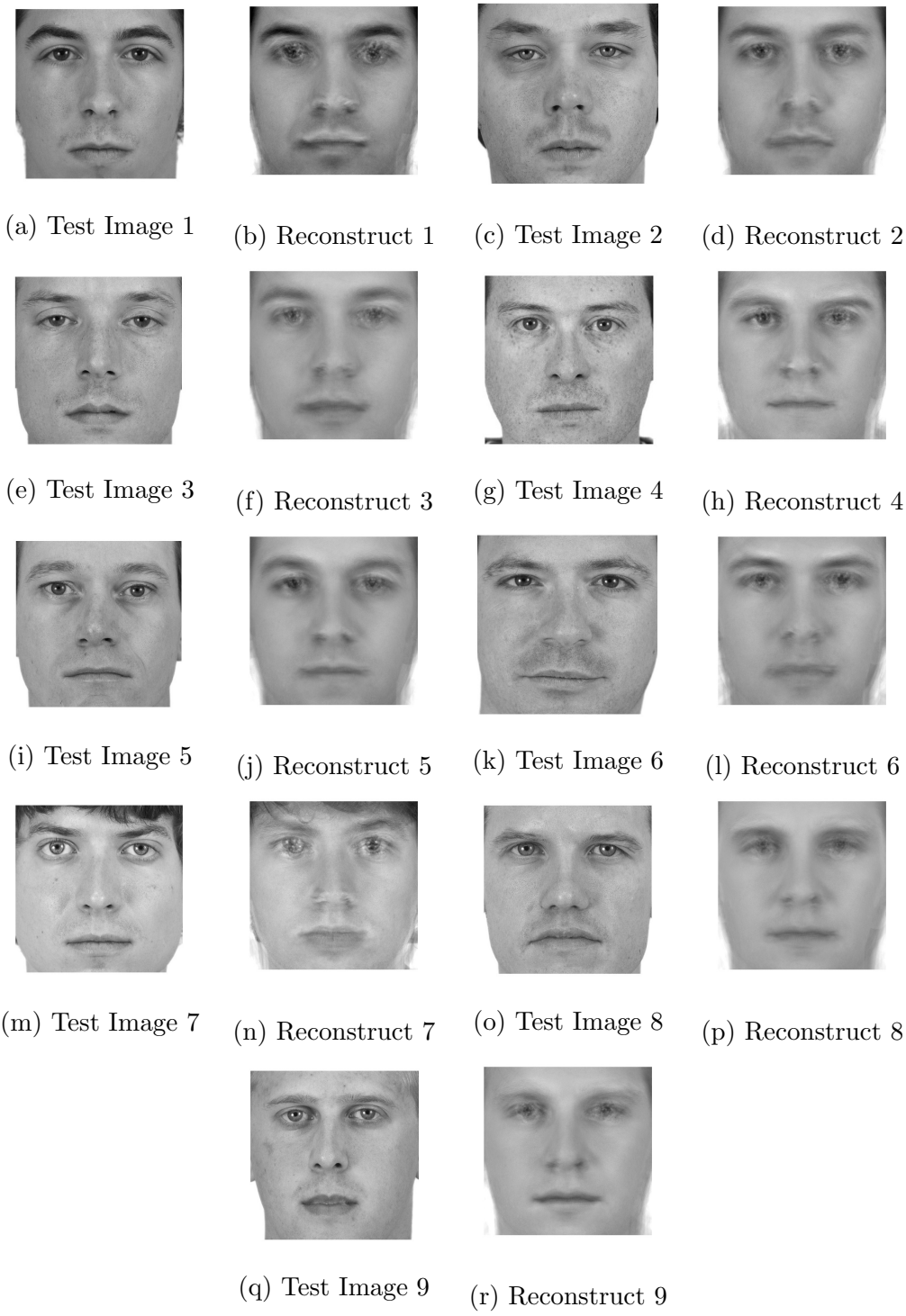
(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.4: Comparison of the nine original test images and reconstruction images using the logarithmic kernel.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

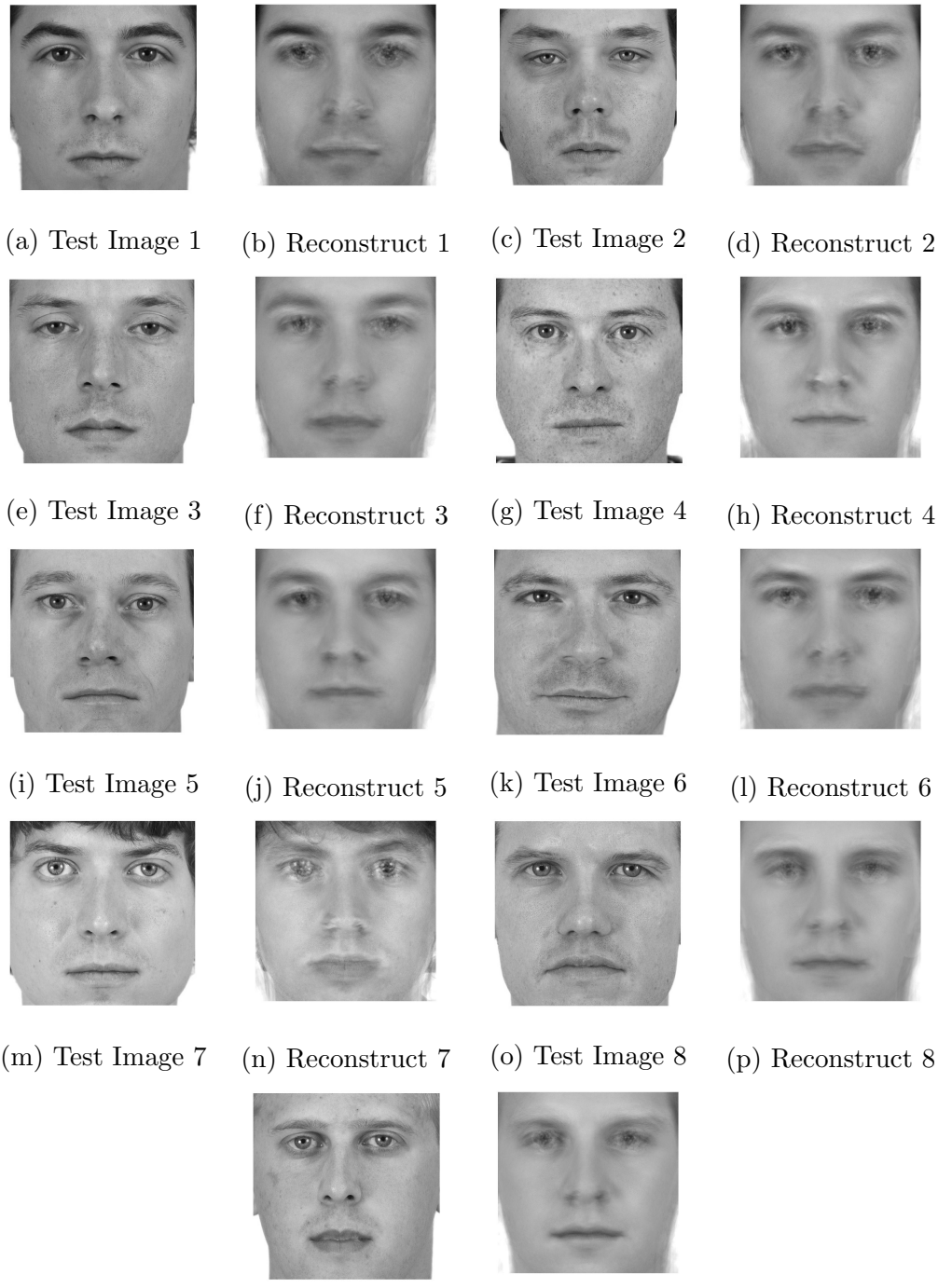
(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.5: Comparison of the nine original test images and reconstruction images using the Cauchy kernel.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

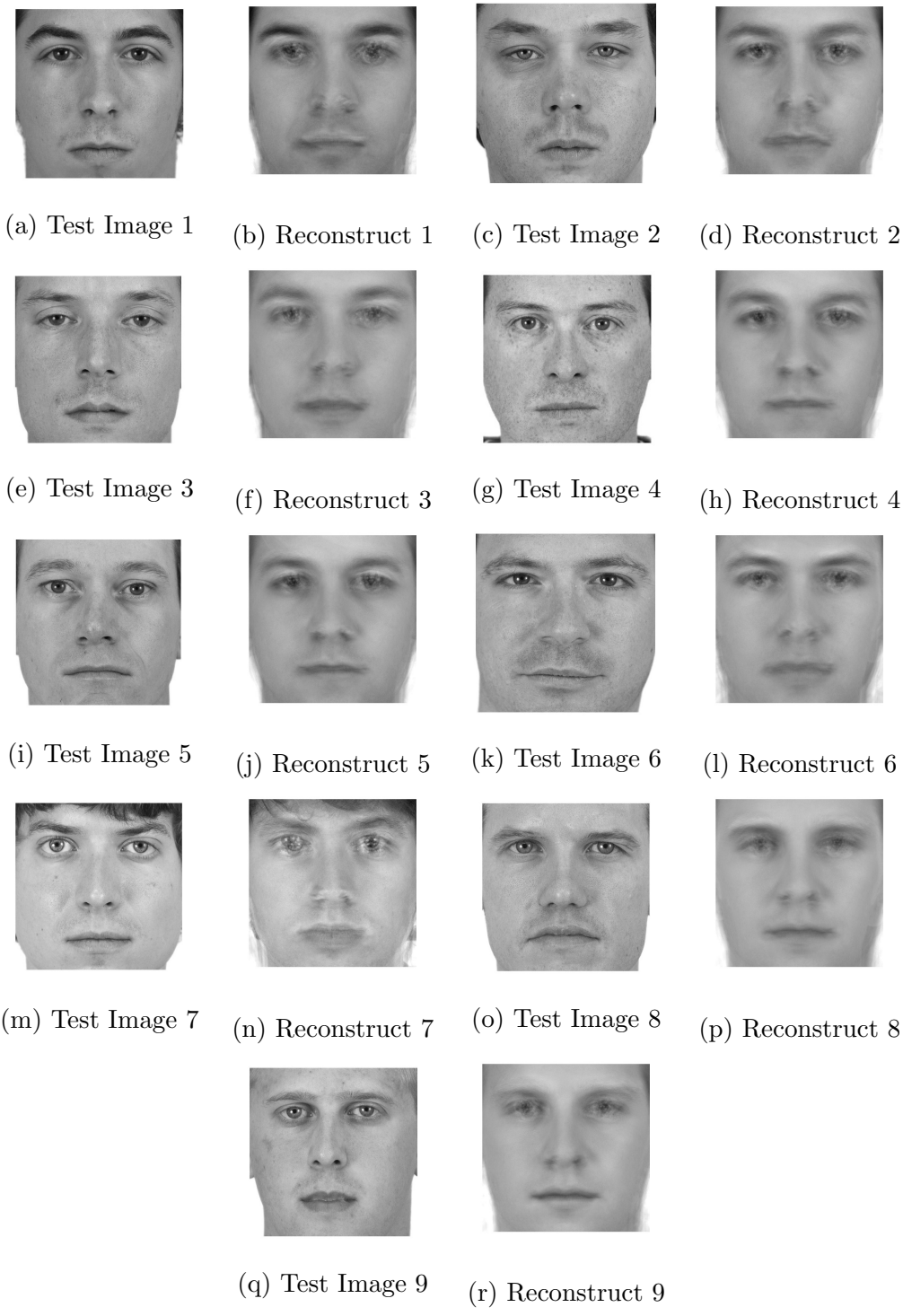
(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.6: Comparison of the nine original test images and reconstruction images using the Polynomial kernel of Degree 2.



(a) Test Image 1 (b) Reconstruct 1 (c) Test Image 2 (d) Reconstruct 2

(e) Test Image 3 (f) Reconstruct 3 (g) Test Image 4 (h) Reconstruct 4

(i) Test Image 5 (j) Reconstruct 5 (k) Test Image 6 (l) Reconstruct 6

(m) Test Image 7 (n) Reconstruct 7 (o) Test Image 8 (p) Reconstruct 8

(q) Test Image 9 (r) Reconstruct 9

Figure 5.7: Comparison of the nine original test images and reconstruction images using the Laplacian kernel.

References

[1] Jafri R, and Arabnia H, "A Survey of Face Recognition Techniques", *Journal of Information Processing Systems, Vol. 5, No. 2*, 41-68, 2009.

[2] Belhumeur P, Hespanha J, and Kriegman D, "Eigenfaces vs. Fischerfaces: Recognition Using Class Specific Linear Projection", *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7*, 711-720, 1997

[3] Sirovich L, and Kirby M, "Low-Dimensional Procedure for the Characterization of Human Faces", *Journal of the Optical Society of America A*, 519-524, 1987

[4] Turk M, and Pentland A, "Kernel Eigenfaces vs. Kernel Fischerfaces" Face Recognition Using Kernel Methods", *In Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recogntion*, 215-220, 2002

[5] Turk M, and Pentland A, "Face Recognition Using Eigenfaces", *Journal of Cognitive Neuroscience, Vol. 3, No. 1*, 71-86, 1991

[6] Parmar D, and Mehta B, "Face Recognition Methods and Applications", *International Journal of Computer Technology and Applications, Vol. 4*, 84-86, 2013

[7] Wang P and Ji Q, "Performance Modeling and Prediction of Face Recognition Systems", *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2*, 1566-1573, 2006

[8] Owen J, "Principal Component Analysis: Data Reduction and Simplification", *McNair Scholars Research Journal: Vol. 1, Article 2*, 1-23, 2014

[9] Hofmann T, Scholkopf B, and Smola A, "Kernel Methods in Machine Learning", *The Annals of Statistics, Vol. 36, No. 2*, 1171-1220, 2008

[10] Anderson T, "Asymptotic Theory for Principal Component Analysis", *The Annals of Mathematical Statistics, Vol. 34, No. 1*, 122-148, 1963

[11] Lay D, "Linear Algebra and its Applications", *Pearson, 4th Edition*, 2012

[12] Jeong D, Ziemkiewicz C, Ribarsky W, and Chang R "Understanding Principal Component Analysis Using a Visual Analytics Tool", *Charlotte Visualization Center UNC Charlotte*, 2009

[13] Pearson K, "On Lines and Planes of Closest Fit to Systems of Points in Space", *Philosophical Magazine 2*, 559-572, 1901

[14] Bhattacharyya S, and Chakraborty S, "Reconstruction of Human Faces from Its Eigenfaces", *International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 1*, 209-214, 2014

[15] Scholkpf B, Smola A, and Mller K, "Kernel Principal Component Analysis", *Advances in Kernel Methods*, 327-352, 1999

[16] Boser B, Guyon I, and Vapnik V, "A Training Algorithm for Optimal Margin Classifiers", *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144-152, 1992

[17] Wang Q "Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models", *arXiv:1207.3538*, 2014

[18] Boughorbel S, Tarel J, and Boujemaa N, "Conditionally Positive Definite Kernels from SVM Based Image Recognition", *IEEE International Conference on Multimedia and Expo*, 113-116, 2005

[19] Rakse S, and Shukla S, "Spam Classification using new kernel function in Support Vector Machine", *International Journal on Computer Science and Engineering, Vol. 2, No. 5*, 1819-1823, 2010

[20] Ma D, Correll J, and Wittenbrink B, "The Chicago Face Database: A Free Stimulus Set of Faces and Norming Data", *Behavior. Research Methods, Vol. 47*, 1122-1135, 2015

[21] Harding B, and Jubinski C, *FaceAccess: A Portable Face Recognition System*, URL: `https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/bjh78_caj65/bjh78_caj65/`

[22] Damle A, and Chikati V, *PCA based Low-Resolution Face Reconstruction*, URL: `http://home.iitk.ac.in/~shubtri/se367/hw2/`