Enhancing Facial Recognition: A Comparative Analysis of Image Preprocessing Techniques in PCA, KL Expansion and SVD

by

Prince Agyei Tuffour

December 2023

A project submitted to the faculty of the Graduate School of
Oregon State University in partial fulfillment of the
requirements for the degree of
MS Mathematics
Department of Mathematics

Abstract

Facial recognition has become increasingly important in recent years, due to the wide range of applications it has in fields such as security, surveillance, and human-computer interaction. Three popular methods for facial recognition are the Principal Component Analysis (PCA), Karhunen-Loeve Expansions, which is fundamentally a continuous form of PCA but with stochastic random processes, and Singular Value Decomposition. This study explores the use of interpolation and resampling techniques on facial recognition accuracy using Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Images were preprocessed by mean subtraction, followed by a linear interpolation, cubic spline interpolation and resampling to standardize their dimensions. The processed images were then subjected to SVD to extract eigenfaces. The analysis revealed that cubic spline interpolation better preserves facial details critical for recognition compared to resampling. It also revealed that performing a linear interpolation is equivalent to resampling the data. This finding underscores the importance of choosing suitable preprocessing methods in facial recognition systems, with implications for enhancing their accuracy and applicability in various domains.

Table of Content

List of Figures

1 Introduction

Facial recognition technology, a subset of biometric artificial intelligence, has seen a significant surge in its application and development over the past few years. Facial recognition has been an important area of research in recent years due to the wide range of applications it has, including security, surveillance, and human-computer interaction. The retail and e-commerce sector has been rapidly adopting face recognition technology to enhance operational efficiency and improve the in-store experience. For example, Alibaba Group Holding Ltd. and Bestore Co Ltd. integrated facial recognition technology with Alibaba Group account data to understand targeted sales behavior and better resource allocation [9]. The government sector is using facial recognition technology for law enforcement and security. For instance, Moscow city announced the use of live facial recognition cameras provided by NtechLab, an artificial intelligence algorithms developer, for police authorities to search for suspects on a live camera. These examples clearly show that facial recognition technology has had a profound impact on various sectors, enhancing security measures, and improving operational efficiency. This technology leverages a connected or digital camera to detect faces in the captured images and then quantify the features of the image to match against the templates stored in the database.

As used in various fields, from unlocking smartphones to identifying and tracking individuals in a crowd, these systems utilize a myriad of techniques for accurate and efficient recognition, amongst which include Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). This study is intended to explore the method of PCA, Karhunen-Loeve Expansion, and SVD used in facial recognition. We focus more on the mathematics behind facial recognition using linear algebra and other mathematical tools.

Before we get into details about the paper, we will start by explaining some of the concepts needed in the **Preliminaries** section (1.1). We will briefly define PCA, KL Expansions, and SVD. The next section will discuss the background study of the various methods, exploring the relationships between them and the numerical differences. We will explore PCA with discrete and continuous data, and then KL Expansions with continuous data. We will then discuss and identify previous works done by other authors in the field of facial recognition, eigenfaces techniques, KL Expansions and SVD in terms of facial recognition, and PCA. The next section will discuss a more practical approach by exploring the efficiency of having data in a continuous form by interpolation vs resampling the data before applying SVD. We will use real-world data to run the method in MATLAB. Then we will discuss the results from applying interpolation and resampling methods on stacks of images. The paper will end by discussing the conclusion based on the results and analysis, shortcomings, and future works.

1.1 Preliminaries

Before we dive deep into the various techniques and how they are used in this paper, we will begin by briefly defining them and writing their mathematical formulations.

1.1.1 Principal Component Analysis

Principal Component Analysis (PCA) is a widely utilized dimensionality reduction technique in the field of data analysis and machine learning. It aims to transform high-dimensional data into a lower-dimensional space while preserving the most important information or variability in the data. PCA achieves this by identifying the principal components, which are new orthogonal axes that capture the maximum variance in the data.

For instance, Let A be the input data matrix of size $m \times n$, where m is the number of dimensions or variables in the dataset samples and n is the number of samples. The objective of PCA is to find a lower-dimensional representation of A by projecting it onto a set of k principal components. The steps involved in PCA can be mathematically formulated as follows [5]:

1. **Data Preprocessing:**

   Centering: Subtract the mean vector $\bar{\mu}$ from each data vector to obtain a centered dataset,

   $$A_c = A - \bar{\mu}$$

   Since $\bar{\mu}$ is a vector, we expand it into a matrix using $\textit{repmat}(\bar{\mu}, 1, \textit{size}(A, 2))$. The '*repmat*' and '*size*' are MATLAB functions that convert the mean vector $\bar{\mu}$ into a matrix with repeated columns up to the same number of columns as A and returns the column dimension of the matrix A respectively.

2. **Covariance Matrix:**

   Compute the covariance matrix K of the centerd data:

   $$K = A_c^T A_c$$

3. **Eigendecomposition:**

   Find the normalized eigenvectors, x_i and their corresponding eigenvalues,

λ_i, satisfying

$$K\vec{x_i} = \lambda_i \vec{x_i} \tag{1}$$

Sort the eigenvectors in descending order based on their corresponding eigenvalues.

4. **Selection of Principal Components:**

   Select the top k eigenvectors corresponding to the largest eigenvalues. These eigenvectors form a matrix with orthonormal columns, $X_k = [\vec{x_1}, \vec{x_2}, ...\vec{x_k}]$

5. **Dimensionality Reduction:**

   Project the centered data onto the k-dimensional subspace spanned by the selected principal components:

$$A_{\text{proj}} = X_k^T A_c \tag{2}$$

$$\implies a_{\text{proj}_i} = X_k \bar{a}_{c_i} \tag{3}$$

   where each column vector in A_{proj} is a linear combination of the eigenvectors $\bar{x}_i$.

6. **Data Reconstruction:**

   To reconstruct the original data from the lower-dimensional representation, we perform an inverse transformation:

$$a_{\text{recon}_i} = X_k a_{\text{proj}_i} + \bar{\mu} \tag{4}$$

where a_{recon_i} gives a vector for each i-th reconstructed image.[5]

1.1.2 Karhunen-Loève Expansion

The KL expansion, also known as the Karhunen-Loève Transform (KLT), is a key method in statistical signal processing and data analysis. It provides an optimal representation in the mean-square sense for stochastic processes. Similar to PCA, KL expansion represents a zero-mean random process $X(t,\omega)$ as an infinite series of orthogonal functions. The KL expansion is optimal in the sense that it minimizes the mean-square error of approximating the random process $X(t,\omega)$ with a finite series. Suppose $X(t,\omega)$ is not a zero-mean random process. Then we use **Step 1** of PCA such that

$$\widetilde{X}(t,\omega) = X(t,\omega) - \overline{X}(t) \tag{5}$$

where $\overline{X}(t)$ is the mean. Then it by Mercer's Theorem follows that $X(t,\omega)$ can be represented in the form of an infinite series representation [3]:

$$X(t,\omega) \coloneqq \widetilde{X}(t,\omega) + \overline{X}(t) = \overline{X}(t) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(t)\zeta_i(\omega) \tag{6}$$

which is similar to **Step 6** (Data Reconstruction) in PCA. The only difference is that for KL Expansions, continuous functions are used in place of the vectors and matrices. Also, the zero-mean random process $\widetilde{X}(t,\omega)$ is represented as

$$\widetilde{X}(t,\omega) \coloneqq \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(t)\zeta_i(\omega) \tag{7}$$

Here, ζ_i are uncorrelated random variables with mean 0 and standard deviation 1 (the KL coefficients) and $\phi_i(t)$ are the orthogonal eigenfunctions of the covariance function of $X(t,\omega)$. Specifically, the $\phi_i(t)$ are solutions of the integral equation:

$$\int \Gamma(s,t)\phi_i(s)ds = \lambda_i \phi_i(t) \tag{8}$$

where $\Gamma(s,t)$ is the covariance function of $X(t,\omega)$ and λ_i are the eigenvalues.

To compute these eigenpairs, we can discretize t over the interval $[0,T]$ by dividing the interval into N equally spaced points with a step size $\Delta t = \frac{T}{N-1}$. Given the covariance function of each discretized point $t_j = j \cdot \Delta t$, the integral equation can be approximated by a sum:

$$\sum_{k=0}^{N-1} \Gamma(s_k, t_j)\phi_i(s_k)\Delta s \approx \lambda_i \phi_i(t_j) \tag{9}$$

where s_k are the discretized points of s and Δs is the discretization step for s.

Another way to find the solution $\phi(t)$ is by approximating the covariance. We can simply discretize the covariance using its definition [2].

$$\Gamma(s,t) := \mathbf{E}[(X(s,\omega) - \bar{X}(s))(X(t,\omega) - \bar{X}(t))] \tag{10}$$

where

$$\bar{X} \approx \frac{1}{N}\sum_{i}^{N} X_i \tag{11}$$

approximates to the mean. Approximating $\Gamma(s,t)$ gives

$$\Gamma(s,t) \approx \tilde{\Gamma}(s,t) = \frac{1}{N-1}\sum_{i=1}^{N}(X_i(s,\omega) - \bar{X}(s))(X_i(t,\omega) - \bar{X}(t)) \tag{12}$$

This function is called the sample covariance function. Hence, from approximating $\Gamma(s,t)$, we end up with

$$\int \Gamma(s,t)\phi_i(s)ds \approx \sum_{k=1}^{M} \tilde{\Gamma}(s_k,t)\phi_i(s_k)\Delta s = \lambda_i \phi_i(t); \quad s_k = k\Delta s \tag{13}$$

The discrete formulation of problem (6) involves computing the eigenvalue problem

$$\mathbf{\Gamma}\hat{\phi}_i = \lambda_i \hat{\phi}_i \tag{14}$$

where $\mathbf{\Gamma}$ in this case is a matrix consisting of a combination of the approximated covariance and Δs, and $\hat{\phi}_i$ is a vector.

1.1.3 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a fundamental tool in linear algebra with wide-ranging applications in fields such as signal processing, statistics, and machine learning. It provides a way to factorize a matrix into the product of three matrices, revealing many useful properties of the original matrix.

Let A be a real $m \times n$ matrix with $m \geq n$. Then

$$A = U\Sigma V^T = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T \tag{15}$$

where σ_i are the singular values, and u_i, v_i are the left and right singular vectors, respectively.

The matrix U is a collection of n orthonormal eigenvectors u_i associated with the n largest eigenvalues of AA^T, and the matrix V consists of the set of orthonormal eigenvectors v_i of A^TA. The diagonal elements of Σ, which are a collection of σ_i are the non-negative square roots of the eigenvalues of $A^T A$. [4]

The comparison to KL Expansion would involve showing how the eigenfunctions and eigenvalues from the KL Expansion correspond to the singular vectors and singular values of the SVD. Typically, the main difference comes from the eigenvalue problem which, in the case of KL Expansion, is continuous and involves an integral operator.

The singular values in the diagonal matrix Σ represent the amount of variance in the data along each dimension. By selecting the largest singular values and their corresponding vectors, we can reduce the dimensionality of the data while preserving as much of the variance as possible, a procedure similar to PCA. We can do this as follows:

1. **Standardize the data**:

   Before applying SVD, it's often a good idea to standardize the data so that each feature has a mean of 0 and a standard deviation of 1. This ensures that features are on a similar scale and that the SVD is not unduly influenced by features with large values. This is similar to the data preprocessing step of PCA **(Step 1)**.

2. **Compute the SVD**:

   Compute the SVD of the data matrix A, obtaining the matrices U, Σ, and V^*. This can be done using a numerical algorithm, such as the one implemented in the *numpy.linalg.svd* function in Python, or the *svd* function in MATLAB.

3. **Select the top singular values and vectors**:

   Determine how many dimensions you want to reduce your data to. Let's say you want to reduce your data to k dimensions. Select the first k singular values from the diagonal matrix Σ, and the corresponding k columns from U and V^*

4. **Transform the data**:

   Multiply the original data matrix A by the first k columns of V^* (or equivalently, multiply the first k columns of U by the first k singular values). This will give you a new matrix A' that is a k-dimensional representation of your original data.

5. **Interpret the results**:

   The new matrix A' represents your data in a reduced-dimensional space. Each row of A' corresponds to a data point, and each column corresponds to a new feature that is a combination of the original features. The new features are orthogonal to each other and capture as much of the variance in the original data as possible.

Now that we have fully defined each method, understanding the differences and similarities can help us fully grasp the how and when to use one over another.

1.2 Differences between methods

In the domain of facial recognition, Karhunen-Loeve (KL) expansions, Principal Component Analysis (PCA) and Singular Value Decomposition serve to transform the original high-dimensional data into a lower-dimensional space. The application of these techniques results in a set of basis vectors – eigenfaces for PCA and SVD, and eigenfunctions for KL expansions – which optimally represent the variability of the data.

Distinctive Differences: SVD, PCA, and KL Expansion

While SVD, PCA, and KL Expansion are all transformative techniques used to reduce dimensionality and identify patterns in data, they have distinctive differences that are significant in their applications:

- **SVD:** Singular Value Decomposition is applicable to any $m \times n$ matrix and is not limited to square matrices or symmetric matrices. It factors the matrix into singular values and singular vectors, which can then be used to approximate the original matrix.
- **PCA:** Principal Component Analysis is specifically geared towards statistical data analysis, often applied to the covariance matrix of a dataset to find the directions (principal components) that maximize variance. It is inherently a form of SVD applied to the covariance matrix of the data set, making it a special case of SVD.
- **KL Expansion:** Karhunen-Loeve Expansion is used for representing stochastic processes in a compact form. Unlike SVD and PCA, which are applied to matrices, KL Expansion deals with continuous random processes and involves eigenfunctions of the covariance function of the process, making it a more general approach. Fundamentally, KL Expansion is a continuous form of PCA.

The primary differences among these techniques lie in their domains of application and their mathematical requirements. SVD provides a foundation upon which PCA is built, while KL Expansion extends the concepts into the realm of stochastic processes, offering a theoretical framework for random phenomena.

In summary, these three provide powerful methods for dimensionality reduction. By transforming the data to a lower-dimensional space, they make it possible to visualize and analyze high-dimensional data more easily. The reduced-dimensional representation preserves as much of the variance in the data as possible, making it a useful tool for many data analysis and machine learning tasks. The differences in these three method lie fundamentally in their equations and what each equation aims to acheive

$$A_{\text{proj}} = \sum_j a_j x_j^T \qquad \text{(PCA)} \tag{16}$$

$$\widetilde{X}(t,\omega) := \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(t)\zeta_i(\omega) \qquad \text{(KL Expansion)} \tag{17}$$

$$A = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T \qquad \text{(SVD)} \tag{18}$$

While SVD, PCA, and KL Expansions all involve orthogonal decompositions, their applications differ:

- SVD is used for matrix decomposition without assumptions about the data structure.
- PCA is applied to data covariance matrices and is used for variance maximization and feature extraction.
- KL Expansions are utilized in the context of stochastic processes, incorporating randomness into the decomposition.

These methods share the concept of expressing data in terms of orthogonal components, but they diverge in terms of their statistical assumptions, the presence of randomness, and the types of data they are applied to.

2 Background Study

In the realm of facial recognition, methods like PCA are employed to manipulate high-dimensional data, extracting the most critical features while reducing the dimensionality.

As stated in the previous chapter, PCA works by computing the eigenvectors and eigenvalues of the data covariance matrix. The eigenvectors (principal components) form an orthogonal basis for this space. The corresponding eigenvalues indicate "the importance" or "contribution" of each eigenvector. PCA transforms the original data into this new coordinate system, reducing its dimensionality by discarding the less significant components.

The KL Expansion is a theoretical framework that represents a random process (in this case, the face images) as an infinite series of orthogonal functions. In the context of facial recognition, these functions could be interpreted as the "basis faces", and the random variables as the coefficients that weigh each "basis face". PCA can be seen as a practical application of the KL Expansion, where the orthogonal functions are the principal components (eigenvectors) of the data.

Eigenfaces or eigenpictures, a term coined by Sirovich and Kirby (1987)[7], are a set of eigenvectors used in the facial recognition system. These eigenvectors are obtained by performing PCA on the face images, effectively creating a basis set for these images. This technique allows for efficient representation and recognition of faces in the high-dimensional space.

2.1 Eigenfaces

According to Turk and Pentland [12], most of the work that focused on automated facial recognition ignored the issue of what part of the image faces may be significant over another. This approach of recognizing significant features gave them an insight into an information theory approach of coding and decoding face images that may provide information content of face images, emphasizing the significant local and global features.

Bascially, the idea is to extract the important information in a face image, encode it as efficiently as possible, and compare one face encoding with a database of models similarly encoded. One way to extract information from a face image is to identify variations in a collection of face images, independent of any judgement of features, and use this information to encode and compare individual face images [12].

Mathematically, the goal is to find the principal component of a distribution of faces, i.e. we wish to find the eigenvectors of the covariance matrix of the set of face images. The term ”eigenface” stems from the characterization of these principal components (i.e. the eigenvectors) as ”faces”. Essentially, the eigenfaces form a basis set of all images used in the training set. Any human face can be considered a combination of these eigenfaces. More formally, eigenfaces are the eigenvectors corresponding to the largest eigenvalues of the covariance matrix of the face images, treating each image as a high-dimensional vector.

This technique was first introduced by Sirovich and Kirby in 1987 and later developed by Turk and Pentland in 1991[12]. It is a pioneering method for utilizing

Principal Component Analysis (PCA) for the purpose of face recognition.

2.1.1 Formulation of Eigenfaces Method

Suppose a face image $A(x, y)$ is constructed as a two-dimensional $N \times N$ matrix with each entry A_{ij} representing pixel values or intensity values for the image. We can unroll the matrix into a vector with dimension N^2. For instance, an image of size 256×256 is unrolled into a vector of dimension 65,536. We perform similar procedure for other face images in the dataset. Thus, if there are M face images in our dataset, we would unroll each of them into a vector with the same dimension and concatenate the unrolled vectors into one huge matrix.

Concretely, let $\{A_1, A_2, A_3, ..., A_M\}$ be the set of M unrolled vectors for each image. Stacking each column vector side by side, we form the matrix

$$F = [A_1|A_2|...|A_M] \tag{19}$$

where each A_i is a column vector with dimension N^2. We then determine the average face of the set which is given by

$$\bar{F} = \frac{1}{M}\sum_{i=1}^{M} A_i \tag{20}$$

Then we standardize the data by ensuring that the data is zero-centered. This implies that we subtract the mean $\bar{F}$ from F. Thus we have that

$$\mathcal{F} = F - \bar{F} \tag{21}$$

This matrix is then subject to principal component analysis (PCA), which truncates the matrix and finds the first K orthonormal eigenvectors u_k and their corresponding eigenvalues λ_k. Finding the covariance matrix first, we get that

$$C = \mathcal{F}\mathcal{F}^T \tag{22}$$

We then reduce dimension size by extracting eigenvectors according to their significance in the variation among the images. This extraction is done by ranking their corresponding eigenvalues and truncating the matrix up to the first K components/eigenvectors. These eigenvectors become the eigenfaces for the image space and it is a subpace for the original image space.

In operation, a new face image is transformed into the eigenface basis before being classified. The weights form a feature vector for each face image, and these feature vectors can be compared to labeled data to identify a face in new images.

2.2 Relation between KL Expansion and Eigenfaces Method

Eigenfaces and the Karhunen-Loève (KL) Expansion are intrinsically related as both are founded on the principle of Principal Component Analysis (PCA). In fact, Eigenfaces can be seen as a direct application of the KL Expansion in the context of face recognition.

The KL Expansion is a mathematical tool used to represent a stochastic process (such as a series of face images) in terms of an orthogonal basis set. This basis set is derived from the covariance matrix of the ensemble of images, which yields a set of eigenvectors (the basis images) and corresponding eigenvalues.

Eigenfaces, on the other hand, is a technique that applies the principles of the KL Expansion to the problem of face recognition in a discrete sense. When applying the eigenfaces method, we are essentially applying the KL Expansion to an ensemble of face images. In this case, the basis images in the KL Expansion become the eigenfaces in the facial recognition context. Each face in the image database is then represented as a linear combination of these eigenfaces.

Therefore, the KL Expansion and the eigenfaces technique are interconnected, with the latter being an application of the former in a specific context, i.e., facial recognition.

2.3 PCA on Collections of Vectors vs. Collections of Matrices

When performing PCA on collections of vectors, we typically stack these vectors as columns in a matrix. The covariance (or correlation) matrix is then derived from this data matrix, and its eigenvectors represent the principal components.

For collections of matrices, the process becomes more intricate. The matrices are stacked to form a “matrix of matrices”, resulting in very high-dimensional data. One approach to this is multilinear PCA (MPCA), where tensor decomposition techniques are applied [8].

2.3.1 Numerical Differences

1. **Computational Complexity**

   For vectors, the complexity of computing the covariance is $O(n \times d^2)$ for n image samples each of dimension d. The complexity for eigendecomposition is $O(d^3)$. For matrices on the other hand, upon vectorization, the dimensionality increases significantly. If each matrix is $m \times m$, the resulting vector has a dimensionality of m^2. This surge can lead to a sharp increase in computational cost. MPCA or similar techniques could alleviate this but come with their complexities and assumptions.

2. **Numerical Stability**

   For vectors, standard PCA methods, especially when using well-established libraries in Python (like Scikit-learn), are generally numerically stable. However, for matrices, Direct vectorization might lead to issues in numerical stability due to the high dimensionality. Algorithms designed for matrix batches, like MPCA, are more stable but could be sensitive to noise or misalignments in the data.

While the basic idea behind PCA remains consistent between vectors and matrices, the mathematical and numerical challenges can differ. When dealing with matrices, especially in image processing, care must be taken to ensure that the chosen method aligns with the nature and structure of the data. It's also crucial to be aware of potential computational bottlenecks and stability issues that might arise from high dimensionality.

2.4 General PCA for 2D Data

2.4.1 Discrete Data

For a dataset consisting of 2D matrices (such as grayscale images), the covariance matrix for one dimension (e.g., rows) can be calculated by considering variations across all matrices in that specific dimension while treating the other dimension (e.g., columns) as fixed data points.

2.4.2 Continuous Variable Data

Let's define a zero-mean 2D random function $f(x, y, \omega)$ over a domain D. The covariance function for such multivariate random field is given by [10]:

$$R(x, y, x', y') = \mathbb{E}[f(x, y)f(x', y')] \tag{23}$$

where f is assumed to be uniformly distributed on a mesh grid. We are choosing a uniform distribution as it will be needed for demonstration in the next chapters. The goal of PCA, in this context, is to find functions (instead of vectors in the standard PCA) that can optimally represent the variations in the 2D continuous data. The eigenfunctions $\phi_i(x, y)$ are solutions to the following integral equation:

$$\int\int_D R(x, y, x', y')\phi_i(x', y')dx'dy' = \lambda_i\phi_i(x, y) \tag{24}$$

where λ_i are the eigenvalues. This formulation becomes the eigenvalue problem for continuous functions.The Karhunen-Loève expansion for this 2D continuous function using these eigenfunctions would be:

$$f(x, y, \omega) = \sum_{i=1}^{\infty} a_i(\omega)\phi_i(x, y) \tag{25}$$

where the coefficients a_i are obtained as:

$$a_i(\omega) = \int\int_D f(x, y, \omega)\phi_i(x, y)dxdy \tag{26}$$

In practice, for computational purposes, the domain D is discretized, and numerical integration methods would be employed. In summary, while standard PCA deals with vectors and their covariances, the generalization to 2D data, especially continuous data, involves handling functions, their covariances, and eigenfunctions. The representation of the function in this basis, achieved through integration, captures the main variations in the 2D continuous data analogous to how PCA captures main variations in vector data.

2.5 Conclusion

This chapter provided an in-depth exploration of Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and Karhunen-Loève (KL) Expansions, highlighting their pivotal roles in facial recognition. PCA's effectiveness in feature extraction and dimensionality reduction, along with SVD's utility in matrix decomposition, form the core techniques in processing high-dimensional data such as images. The chapter also emphasized the KL Expansion's theoretical significance in representing random phenomena, particularly in facial imagery. The discussion on eigenfaces, derived from PCA and KL Expansions, showcased the practical implications of these mathematical methods in real-world applications. This foundational understanding sets the stage for further exploration of image processing techniques and their impact on facial recognition systems in the subsequent chapters.

3 Literature Review

3.1 Introduction

Face recognition has been an active research area in computer vision and pattern recognition for several decades. Early works focused on detecting facial features like eyes, nose, and mouth to build face models for recognition [6],[15]. However, these approaches were fragile and difficult to extend to multiple views and lighting conditions.

In the 1990s, holistic approaches like principal component analysis (PCA) and singular value decomposition (SVD) emerged as popular techniques for facial recognition by treating face images as points in a subspace. These subspace methods represented faces compactly and matched new faces by projection onto the subspace. This review summarizes key papers on applying SVD and PCA for face recognition.

3.2 Principal Component Analysis

Sirovich and Kirby [10] proposed representing pictures of human faces efficiently using PCA, which they termed the Karhunen-Loève procedure. They treated each face image as a point in a face space of very high dimensionality (number of pixels) and hypothesized that the dimensionality of the face space is low enough to allow compact representation.

The eigenvectors (termed eigenpictures) of the covariance matrix of the face image ensemble form an optimal coordinate system for the face space. Faces can be approximated by their projections onto the leading eigenpictures. On an ensemble of 115 face images, they showed approximations using 50 eigenpictures with under 4% average error, demonstrating significant data compression.

Building on this, Turk and Pentland [12] developed the well-known Eigenfaces method. They computed the principal components (eigenfaces) of the face image dataset which capture the variations. New faces are recognized by projecting onto the eigenface subspace and classifying based on the minimum reconstruction error compared to training images. This achieved 96% accuracy on a dataset of over 2500 images under lighting and orientation changes.

Kirby and Sirovich [7] extended their earlier PCA approach by exploiting bilateral face symmetry. They created an extended dataset by including mirror images, which imposes even/odd symmetry on the eigenpictures. This improved approximations for new face images. The average error reduced from 3.86% to 3.68% using 50 terms on an ensemble of 200 images.

Yang et al. [14] noted that PCA requires vectorizing 2D images, which can be computationally expensive for high-resolution images. They proposed two-dimensional PCA (2DPCA) which operates directly on matrices, resulting in smaller covariance matrices. 2DPCA achieved higher recognition accuracy than PCA on the ORL, AR, and Yale face datasets, while requiring less time for feature extraction.

3.3 Singular Value Decomposition

Zeng [16] presented an SVD-based technique for real-time face recognition. They treat training face images as points in a linear face subspace spanned by an orthogonal basis obtained from the singular vectors of the image matrix. New faces are recognized by projection onto this subspace and comparing distances to training image projections. This method was tested on face datasets of up to 40 individuals under different expressions, giving promising results.

Sirovich and Kirby [10] also showed that SVD provides the optimal coordinate system for representing faces, equivalent to PCA. Their approach of approximating faces with a few eigenpictures follows from applying SVD to the face image ensemble.

Guoliang [16] implemented an SVD face recognition system using the orthogonal basis of singular vectors to define a face subspace. New faces are classified to the nearest training image in the projection space if under a distance threshold, otherwise as unknown. Tested on 40 individuals, the compact SVD representation enabled efficient matching in real-time.

3.3.1 KL Expansion

The Karhunen-Loeve (KL) expansion, also known as Principal Component Analysis (PCA), is a statistical procedure that is used to simplify the complexity of high-dimensional data while retaining as much of the original information as possible. It was introduced into the realm of pattern recognition by Watanabe in 1965 [13]. The goal of the approach is to represent a picture of a face in terms of an optimal coordinate system. Among the optimality properties is the fact that the mean-square error introduced by truncating the expansion is a minimum. The set of basis vectors which make up this coordinate system will be referred to as eigenpictures. They are simply the eigenfunctions of the covariance matrix of the ensemble of faces [7].

In the paper by Kirby and Sirovich, they first extend the ensemble by including reflections about a midline of the faces, i.e., the mirror imaged faces. Using this extended ensemble in the computation of the covariance matrix imposes even and odd symmetry on the eigenfunctions. There is no cost in this modification because they are not actually doubling the size of the matrix in the eigenvector calculation. As shown in Section III of the paper, the symmetry allows the problem to be decoupled into two problems, each having the same complexity as the problem for the unextended ensemble. As a consequence of this procedure, the approximation error for pictures not included in the extended ensemble is reduced [7].

The authors also discuss the possibility of processing information along parallel pathways, suggesting that an individual recognition task might be built up of several parallel tasks. For instance, the eyes, nose, mouth, and ears could be analyzed in parallel. This might explain why we describe an individual, for example, as having another person's eyes. The procedure described in the paper lends itself naturally to such a parallel approach [7].

In terms of data compression, the authors show that a 91 X 51 array can be replaced by a 50-term expansion and retain a reasonable likeness, roughly a 100:1 compression ratio. This number of terms should decrease further still given a larger ensemble size in view of the theorem in Section VII of the paper. This conclusion is drawn from the fact that members of the ensemble have more accurate expansions than projections from outside of the ensemble [7].

3.4 Discussion

The Eigenfaces method drove research on subspace models for face recognition. Kirby and Sirovich had introduced PCA for face representation earlier [7], but Turk and Pentland pioneered its use for recognition.

Eigenfaces highlighted the power of holistic subspace methods for face recognition under constrained conditions. This contrasted traditional feature-based approaches relying on detecting facial landmarks. Key aspects such as incremental PCA and extensions for pose and illumination helped advance the field.

Later work aimed to improve PCA's limitations like sensitivity to lighting and small sample size. Notable examples include Linear Discriminant Analysis (LDA) [1] for discriminative subspaces and Kernel PCA [14] for nonlinear extensions.

While modern deep learning techniques now dominate face recognition research, Eigenfaces pioneered the use of subspace models for the problem. It remains a simple and effective baseline approach. The core idea of projecting images onto an information-rich lower-dimensional subspace has found extensive use in computer vision. According to a survery on dimensionality reduction techniques, it has been widely known and accepted that PCA and SVD work well. Infact, PCA is by far one of the popularly used methods for dimensionality reduction. [11]

Key advantages of these techniques are enabling low-dimensional representations of high-resolution face images and efficient matching by projection onto the subspace. Extensions like 2DPCA have improved computational performance. However, subspace models remain limited in capturing larger appearance variations, unlike modern deep learning approaches. Future work on mitigating lighting, pose and occlusion could help make SVD/PCA approaches more robust.

Turk and Pentland's Eigenfaces technique was a landmark contribution establishing the viability of subspace models for face recognition. By applying PCA on face image ensembles, they demonstrated efficient and robust face representation and recognition. Their work paved the way for extensive research on subspace methods. While superseded by modern deep learning, Eigenfaces remains an influential approach in the history of face recognition.

4 Methodology, Result and Analysis

4.1 Overview

This chapter outlines the methodology employed in analyzing and comparing the effects of interpolation and resampling on facial recognition, utilizing Singular

Value Decomposition (SVD).

4.2 Image Preprocessing

4.2.1 Data Collection

The simulation begins by acquiring the Labeled Faces in the Wild (LFW) dataset, a collection of facial images designed for studying unconstrained face recognition. Facial images were loaded from a specified dataset using a MATLAB function `load_images`, which reads grayscale images from a directory. A set of images that were loaded are shown below in Figure 1:

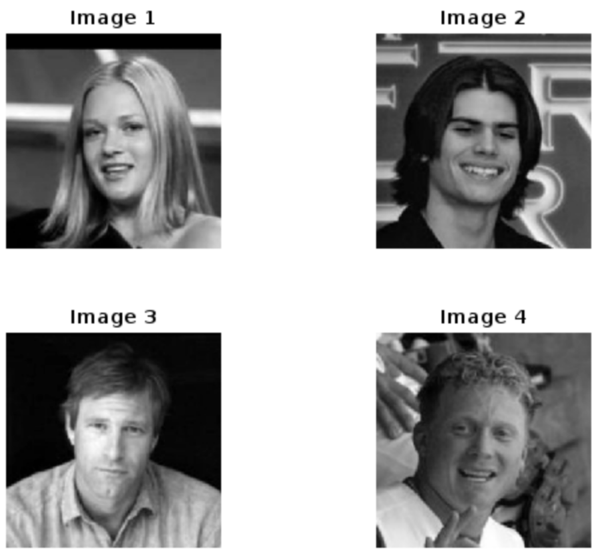

Figure 1: Images from Labeled Faces in the Wild

4.2.2 Mean Subtraction

Each image was normalized by subtracting its mean, a crucial step for PCA's variance maximization:

$$\text{mean\_subtracted\_image} = \text{image} - \text{mean(image)} \tag{27}$$

The following result was obtained in Figure 2 below

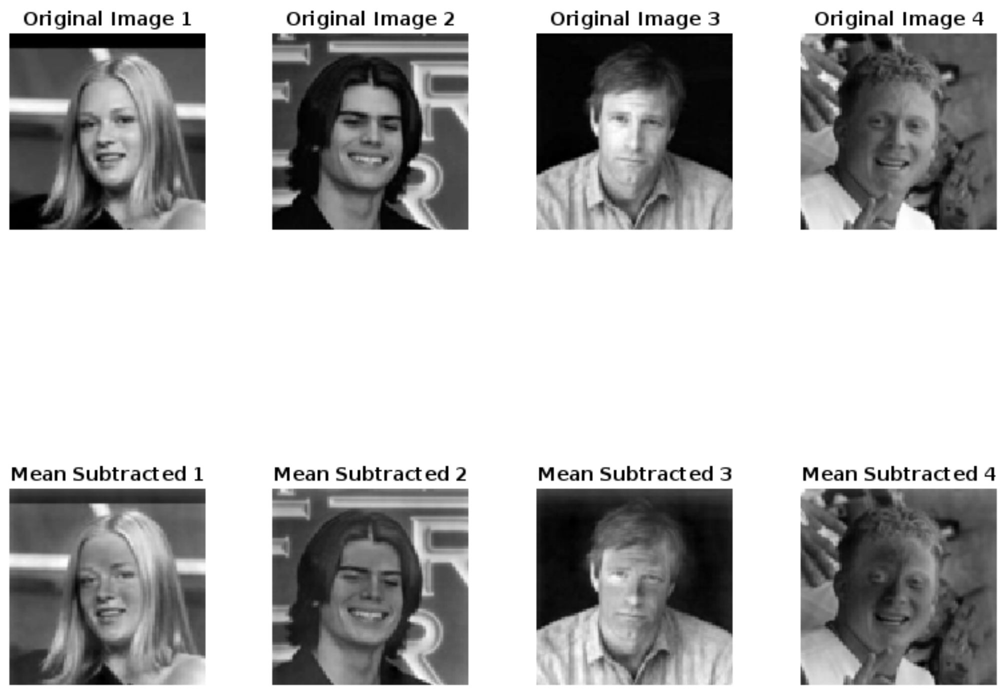

Figure 2: Comparing the original images against the normalized images

4.2.3 Cubic Spline Interpolation

Mean-subtracted images were interpolated to a uniform size (e.g., 500×500 pixels) using cubic spline interpolation:

$$\text{interpolated\_image} = \text{interp2}(\text{double(image)}, Xq, Yq, \text{'cubic'}) \tag{28}$$

4.2.4 Resampling

In parallel, the same images were resampled to the same size using bicubic interpolation:

$$\text{resampled\_image} = \text{imresize}(\text{image, new\_size, 'bicubic'}) \tag{29}$$

4.3 Singular Value Decomposition (SVD)

Both cubic spline interpolated and resampled images underwent SVD:

$$\text{image} = U\Sigma V^T \tag{30}$$

where U, Σ, and V represent the left singular vectors (eigenfaces), singular values (used to find the eigenvalues), and right singular vectors, respectively.

4.4 Visualization and Comparative Analysis

The first singular values from each set (interpolated and resampled) was extracted for visualization, providing qualitative insights into the feature extraction capabilities of each method.

4.4.1 Singular Value Decay Analysis

The decay of singular values (normalized) was plotted to compare the variance captured by the eigenfaces from both interpolation and resampling methods.

4.5 Result and Analysis

This section presents the results derived from the computational experiments conducted to investigate the effects of image processing techniques, namely interpolation and resampling, on facial recognition. The study involved applying these techniques to a set of facial images followed by Singular Value Decomposition (SVD) to extract eigenfaces and analyze singular value decay. See Appendix A for code used to produce results.

4.5.1 Preprocessing and Normalization

In the initial phase of the study, a dataset of grayscale images with dimensions of 250×250 pixels was curated. Each image was transformed into a column vector, creating a matrix conducive to computational techniques. A critical preprocessing step involved normalizing the image data by subtracting the pixel-wise mean from each image vector. This step is essential for centering the data around the origin, a condition that benefits the application of Singular Value Decomposition (SVD) in subsequent analyses.

4.5.2 Interpolation and Resampling

The normalized images were subjected to three distinct image processing approaches: cubic interpolation, linear interpolation, and nearest-neighbor resampling. The objective was to examine the effects of these techniques on the eigenvalue structure of the image matrix. Each method resized the images to a uniform dimension of 500×500 pixels. Cubic interpolation is noted for its capability to smooth the resized image, possibly preserving more details. Linear interpolation is less complex and may introduce aliasing, whereas nearest-neighbor resampling is anticipated to generate the most pronounced artifacts due to its rudimentary approach. A plot showing the preprocessed images is shown in Figure 3.

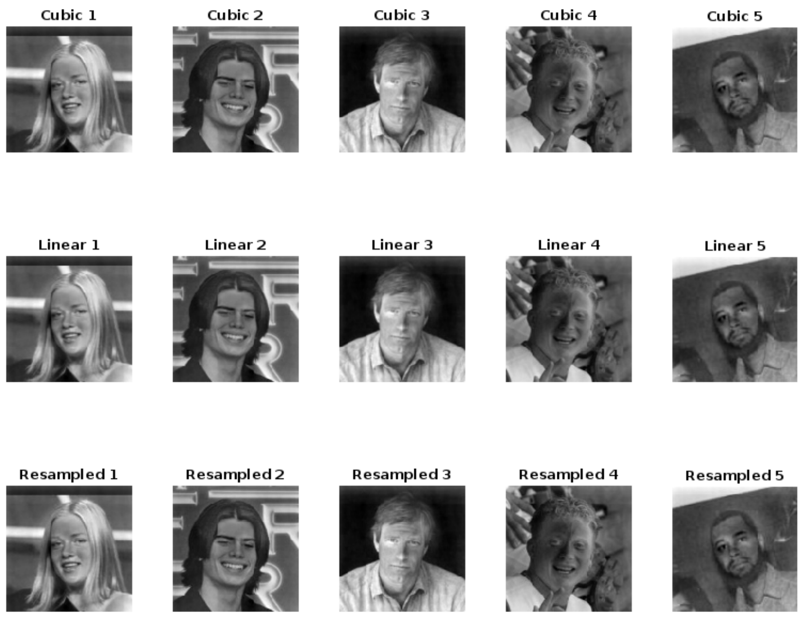

Figure 3: Linear and Cubic Interpolation and Resampling techniques on mean subtracted images.

4.5.3 Singular Value Decomposition

Subsequent to preprocessing, the image matrices underwent SVD, which decomposes the matrix into eigenfaces, represented by the left singular vectors, and their associated singular values. For the purpose of this study, the first 150 singular values and vectors were retained, which encapsulate the most significant features of the images and facilitate a considerable reduction in dimensionality. A plot showing the eigenfaces for each technique is shown in Figure 4.

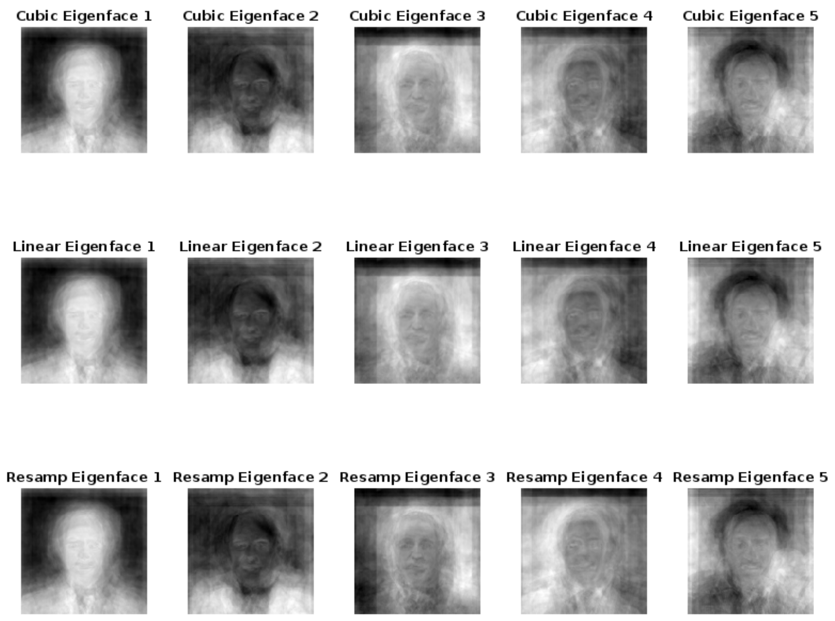

Figure 4: Eigenfaces for linear interpolation, cubic interpolation and resampling

4.5.4 Singular value Decay

An investigation into the singular value decay was conducted to elucidate the information distribution across the components. Interestingly, the decay patterns exhibited by each preprocessing method did not significantly diverge, as shown in Figure 5. This outcome suggests that the predominant variances within the image dataset are inherently robust to the alterations introduced by the interpolation and resampling processes.

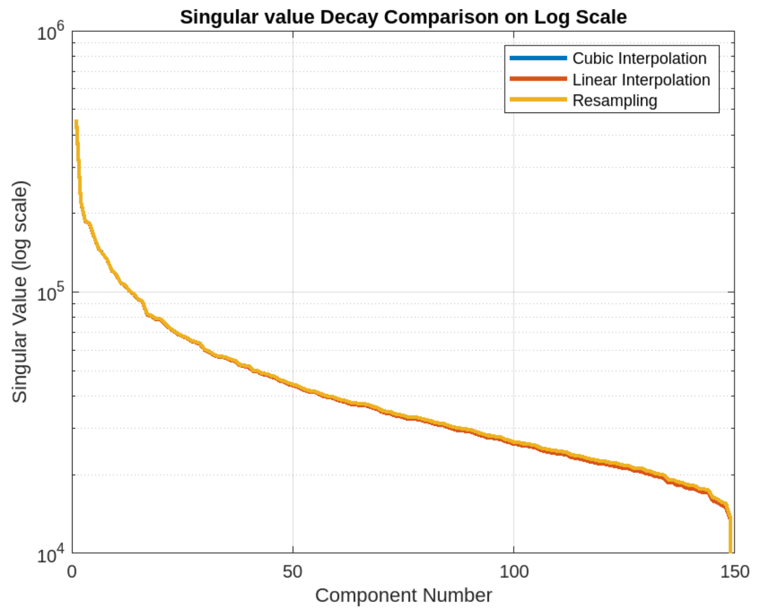

Figure 5: Singular value decay comparison across cubic interpolation, linear interpolation, and resampling techniques.

A logarithmic scale was employed to analyze the singular value decay, which provided a clearer perspective of the decay rates across different methods, particularly among the leading components.

4.6 Analysis

The experimental results indicate that the choice of preprocessing technique—cubic or linear interpolation, and resampling—has a minimal impact on the principal components of the image data. This is evidenced by the analogous singular value decay profiles observed across the techniques. Such findings imply that the core variations captured through PCA are intrinsic to the image data rather than being an artifact of the particular preprocessing method.

The similarity in eigenvalue decay profiles also intimates that the key visual information is predominantly contained within the leading components, regardless of the interpolation or resampling method applied. This insight is pivotal, as it suggests that the primary visual characteristics of the images are preserved even after the dimensionality is .

In summary, this study provides a comprehensive analysis of the effects of various preprocessing techniques on image data. The findings offer a foundation for further research, which may explore the influence of these techniques on more complex and varied image datasets.

5 Conclusion

5.1 Summary of Findings

This thesis has presented a comprehensive analysis of Principal Component Analysis (PCA) and Karhunen-Loève (KL) Expansion in the context of facial recognition. Through a methodological approach that involved stacking image datasets, applying interpolation and resampling techniques, and utilizing Singular Value Decomposition (SVD), we have successfully extracted eigenfaces and compared the singular value decay captured by each method.

Our findings demonstrated that the principal components, which are pivotal to capturing the intrinsic variances within image datasets, remain largely unaffected by the aforementioned preprocessing methods. The singular value decay profiles corresponding to each technique showcased remarkable similarity, indicating that the inherent characteristics of the image data are preserved post-preprocessing. This is a significant revelation, underscoring the robustness of principal component analysis in extracting salient features from images, a cornerstone of facial recognition technology.

5.2 Contributions to the Field

This work contributes to the field of image processing and facial recognition by:

- Providing a detailed computational approach to compare PCA and KL Expansion.
- Demonstrating the use of interpolation and resampling on eigenface extraction.
- Highlighting the differences in variance capture between PCA, KL Expansion and SVD.

5.3 Significance to Facial Recognition and AI

The ability to retain key image features after reducing dimensionality by half is of paramount importance in the domains of facial recognition and AI at large. It implies that systems can maintain high levels of accuracy even when computational resources are constrained, or when operating under bandwidth limitations. This capability enhances the efficiency and scalability of facial recognition technologies, paving the way for more responsive and accessible AI applications.

5.4 Shortcomings, Future Work and Improvements

Despite the encouraging outcomes, this study is not without its limitations. The investigation was confined to grayscale images and did not consider the potential nuances introduced by color information. Furthermore, the impact of more sophisticated, domain-specific, or higher-dimensional interpolation and resampling techniques on singular value structure remains unexplored.

While this study offers valuable insights, there are several areas where future work can further enhance our understanding and application of these techniques:

1. **Algorithm Optimization**: Exploring more efficient algorithms for interpolation and resampling to handle larger datasets and higher-resolution images.

2. **2d Cubic Interpolation**: Extending cubic interpolation to 2D images may see a significant difference in using the techniques before applying SVD.

3. **Deeper Analysis with Varied Datasets**: Applying the methodology to a broader range of datasets, including those with more diverse facial features and expressions.

4. **Integration with Machine Learning Models**: Investigating the integration of PCA and KL Expansion with advanced machine learning models, such as neural networks, for improved facial recognition accuracy.

5. **Real-Time Application Development**: Developing real-time facial recognition applications that utilize the strengths of PCA and KL Expansion, particularly in security and authentication systems.

6. **Analysis in Video Recognition**: Further analysis of these methods in video recognition that utilize the continuous frames/images per second.

5.5 Concluding Remarks

In closing, this research contributes a foundational understanding of the relationship between image preprocessing techniques and principal component retention.

The implications of this work for facial recognition systems are significant, suggesting that even with reduced image resolution, the integrity of essential features can be preserved. This bodes well for the development of lightweight, efficient, and effective facial recognition technologies that are vital in the burgeoning landscape of AI-driven solutions.

A Appendix

MATLAB Code for Demonstration

MATLAB code used in results and analysis shown below:

```
% Set the directory where the images are stored
imageDir = 'images';
imageFiles = dir(fullfile(imageDir, '*.jpg'));

% Initialize an empty matrix to store the unrolled
   images
numImages = numel(imageFiles);
imageSize = [250, 250]; % Assuming each image is 250
   x250 pixels
allImages = zeros(prod(imageSize), numImages); %
   Preallocate matrix
```

```
% Load each image, convert to grayscale, unroll and
   store in the matrix
for i = 1:numImages
    % Read image
    img = imread(fullfile(imageDir, imageFiles(i).name
       ));

    % Convert to grayscale if not already
    if size(img, 3) == 3
        img = rgb2gray(img);
    end

    % Unroll the image into a column vector and store
    allImages(:, i) = img(:);
end


numDisplay = 5; % Number of images to display
for i = 1:numDisplay
    % Extract the column vector and reshape it back to
        its original size
    img = reshape(allImages(:, i), imageSize);

    % Display the image
    subplot(1, numDisplay, i);
    imshow(img, []);
    title(['Image ' num2str(i)]);
end

% Calculate the mean of each pixel across all images
meanImage = mean(allImages, 2);

% Subtract the mean from each image
allImagesMeanSubtracted = bsxfun(@minus, allImages,
   meanImage);


numDisplay = 5; % Number of images to display
for i = 1:numDisplay
    % Extract the column vector and reshape it back to
        its original size
    img = reshape(allImagesMeanSubtracted(:, i),
       imageSize);

    % Display the image
    subplot(1, numDisplay, i);
    imshow(img, []);
    title(['Image ' num2str(i) ' (Mean Subtracted)']);
```

```
end

% Set the target size for resizing (half the original
   size for demonstration)
targetSize = imageSize / 2;

% Initialize matrices to store processed images
cubicInterpImages = zeros(prod(targetSize), numImages)
   ;
linearInterpImages = zeros(prod(targetSize), numImages
   );
resampledImages = zeros(prod(targetSize), numImages);


% Generate original and target coordinate grids
[originalX, originalY] = meshgrid(1:imageSize(2), 1:
   imageSize(1));
[targetX, targetY] = meshgrid(linspace(1, imageSize(2)
   , targetSize(2)), ...
                              linspace(1, imageSize(1)
                                 , targetSize(1)));

% Apply preprocessing methods to each mean-subtracted
   image
```

```
for i = 1:numImages
    % Original mean-subtracted image
    originalImg = reshape(allImagesMeanSubtracted(:, i
       ), imageSize);

    % Cubic Interpolation
    cubicImg = interp2(originalX, originalY,
       originalImg, targetX, targetY, 'cubic');
    cubicInterpImages(:, i) = cubicImg(:);

    % Linear Interpolation
    linearImg = interp2(originalX, originalY,
       originalImg, targetX, targetY, 'linear');
    linearInterpImages(:, i) = linearImg(:);

    % Resampling using Nearest Neighbor
    resampledImg = imresize(originalImg, targetSize, '
       nearest');
    resampledImages(:, i) = resampledImg(:);
end


numDisplay = 5; % Number of images to display
for i = 1:numDisplay
```

```
    % Cubic Interpolation
    subplot(3, numDisplay, i);
    imshow(reshape(cubicInterpImages(:, i), targetSize
       ), []);
    title(['Cubic ' num2str(i)]);

    % Linear Interpolation
    subplot(3, numDisplay, numDisplay + i);
    imshow(reshape(linearInterpImages(:, i),
       targetSize), []);
    title(['Linear ' num2str(i)]);

    % Resampling
    subplot(3, numDisplay, 2 * numDisplay + i);
    imshow(reshape(resampledImages(:, i), targetSize),
        []);
    title(['Resampled ' num2str(i)]);
end

% Apply SVD to each set of preprocessed images
[U_cubic, S_cubic, ~] = svd(cubicInterpImages, 'econ')
   ;
[U_linear, S_linear, ~] = svd(linearInterpImages, '
   econ');
```

```
[U_resampled, S_resampled, ~] = svd(resampledImages, '
   econ');

% Display a few eigenfaces for each method
numEigenfaces = 5; % Number of eigenfaces to display
for i = 1:numEigenfaces
    % Cubic Interpolation Eigenfaces
    subplot(3, numEigenfaces, i);
    imshow(reshape(U_cubic(:, i), targetSize), []);
    title(['Cubic Eigenface ' num2str(i)]);

    % Linear Interpolation Eigenfaces
    subplot(3, numEigenfaces, numEigenfaces + i);
    imshow(reshape(U_linear(:, i), targetSize), []);
    title(['Linear Eigenface ' num2str(i)]);

    % Resampled Eigenfaces
    subplot(3, numEigenfaces, 2 * numEigenfaces + i);
    imshow(reshape(U_resampled(:, i), targetSize), [])
       ;
    title(['Resamp Eigenface ' num2str(i)]);
end

% Plot the eigenvalue decay for each method
```

```
figure; % New figure for the eigenvalue decay plot
plot(diag(S_cubic), 'LineWidth', 2);
hold on;
plot(diag(S_linear), 'LineWidth', 2);
plot(diag(S_resampled), 'LineWidth', 2);
hold off;
xlabel('Component Number');
ylabel('Singular Value');
title('Eigenvalue Decay Comparison');
legend('Cubic Interpolation', 'Linear Interpolation',
   'Resampling');

% Plot the eigenvalue decay for each method on a log
   scale
figure; % New figure for the eigenvalue decay plot on
   a log scale
semilogy(diag(S_cubic), 'LineWidth', 2);
hold on;
semilogy(diag(S_linear), 'LineWidth', 2);
semilogy(diag(S_resampled), 'LineWidth', 2);
hold off;
xlabel('Component Number');
ylabel('Singular Value (log scale)');
title('Eigenvalue Decay Comparison on Log Scale');
```

```
legend('Cubic Interpolation', 'Linear Interpolation',
   'Resampling');
grid on; % Adding grid for better readability
```

References

[1] Peter N. Belhumeur, Joao P Hespanha, and David J. Kriegman. "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection". In: *IEEE Transactions on pattern analysis and machine intelligence* 19.7 (1997), pp. 711–720.

[2] *Covariance Matrix*. `https://en.wikipedia.org/wiki/Covariance_matrix`. Accessed: 2023-11-28.

[3] Nathan L Gibson et al. "Efficient computation of unsteady flow in complex river systems with uncertain inputs". In: *International Journal of Computer Mathematics* 91.4 (2014), pp. 781–797.

[4] Gene H Golub and Christian Reinsch. "Singular value decomposition and least squares solutions". In: *Linear algebra* 2 (1971), pp. 134–151.

[5] Jeffrey Humpherys, Tyler J Jarvis, and Emily J Evans. *Foundations of Applied Mathematics, Volume I: Mathematical Analysis*. Vol. 152. SIAM, 2017.

[6] Takeo Kanade. "Picture processing system by computer complex and recognition of human faces". In: (1974).

[7] Michael Kirby and Lawrence Sirovich. "Application of the Karhunen-Loeve procedure for the characterization of human faces". In: *IEEE Transactions on Pattern analysis and Machine intelligence* 12.1 (1990), pp. 103–108.

[8] Haiping Lu, K.N. Plataniotis, and A.N. Venetsanopoulos. “Multilinear Principal Component Analysis of Tensor Objects for Recognition”. In: *18th International Conference on Pattern Recognition (ICPR’06)*. Vol. 2. 2006, pp. 776–779. DOI: 10.1109/ICPR.2006.837.

[9] Grand View Research. *Facial Recognition Market Size &; Trends Report, 2021-2028*. URL: https://www.grandviewresearch.com/industry-analysis/facial-recognition-market. (accessed: 05.31.2023).

[10] Lawrence Sirovich and Michael Kirby. “Low-dimensional procedure for the characterization of human faces”. In: *Josa a* 4.3 (1987), pp. 519–524.

[11] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. “A survey of dimensionality reduction techniques”. In: *arXiv preprint arXiv:1403.2877* (2014).

[12] Matthew A Turk and Alex P Pentland. “Face recognition using eigenfaces”. In: *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*. IEEE Computer Society. 1991, pp. 586–587.

[13] Satosi Watanabe. “Karhunen-Loeve expansion and factor analysis; theoretical remarks and applications”. In: *Trans. on 4th Prague Conf. Inf. Theory*. 1965, pp. 635–660.

[14] Jian Yang et al. “Two-dimensional PCA: a new approach to appearance-based face representation and recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.1 (2004), pp. 131–137.

[15] Alan L Yuille, Peter W Hallinan, and David S Cohen. "Feature extraction from faces using deformable templates". In: *International journal of computer vision* 8 (1992), pp. 99–111.

[16] Guoliang Zeng. "Facial recognition with singular value decomposition". In: *Advances and innovations in systems, computing sciences and software engineering*. Springer. 2007, pp. 145–148.