DISSERTATION

FAST EIGENSPACE DECOMPOSITION OF CORRELATED IMAGES USING THEIR SPATIAL AND TEMPORAL PROPERTIES

Submitted by

Kishor Saitwal

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2006

COLORADO STATE UNIVERSITY

April 11, 2006

WE HEREBY RECOMMEND THAT THE **DISSERTATION** PREPARED UNDER OUR SUPERVISION BY **KISHOR SAITWAL** ENTITLED **FAST EIGENSPACE DECOMPOSITION OF CORRELATED IMAGES USING THEIR SPATIAL AND TEMPORAL PROPERTIES** BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Prof. Edwin K.P. Chong

Prof. Bruce A. Draper

Prof. Iuliana Oprea

Prof. Rodney G. Roberts

Prof. Anthony A. Maciejewski
Adviser

Prof. Anthony A. Maciejewski
Department Head/Director

ABSTRACT OF DISSERTATION

FAST EIGENSPACE DECOMPOSITION OF CORRELATED IMAGES USING THEIR SPATIAL AND TEMPORAL PROPERTIES

Eigendecomposition-based techniques play an important role in numerous image processing and computer vision applications. The advantage of these techniques is that they are purely appearance based and require few online computations. All eigenspace methods take advantage of the fact that a set of highly correlated images can be approximately represented by a small set of eigenimages. However, the offline calculation required to determine both the appropriate number of eigenimages as well as the eigenimages themselves can be prohibitively expensive. This thesis considers two issues associated with the calculation of the eigendecomposition of correlated images, i.e., the effect of spatial resolution reduction and correlations associated with three-dimensional pose estimation.

The first part of this thesis addresses the issue of computing the eigendecomposition of one-dimensional correlated images. It is well known that the computation of an eigendecomposition can become prohibitively expensive when dealing with very high-resolution images. While reducing the resolution of the images will reduce the computational expense, it is not known *a priori* how this will affect the quality of the resulting eigendecomposition. This work provides an analysis of how different resolution reduction techniques affect the eigendecomposition. A computationally efficient algorithm for calculating the eigendecomposition based on this analysis is also presented. Examples show that this algorithm performs very well on images of objects rotated along a single axis and on arbitrary video sequences.

The second part of this thesis considers the computation of the eigendecomposition of general three-dimensional image sets that can be used in pattern recognition applications; specifically in the three-dimensional pose estimation of objects. Previous work has shown

that the correlation associated with one-dimensional pose estimation can be used to accelerate the computation of the eigendecomposition. In this work, it is shown how this algorithm can be extended to take advantage of the correlations in three-dimensional pose estimation.

Kishor Saitwal
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, CO 80523
Summer 2006

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First of all, I would like to acknowledge my advisor, Dr. Anthony A. Maciejewski, for his valuable guidance throughout my graduate research. In my opinion, he is an ideal advisor. Although he knew his research area very well, he was very open to listen to new ideas from me. Also, his flexibility with his research needs gave me a lot of motivation to perform well as his Ph.D. student. It was a very steep learning curve for me in the beginning. However, I am very satisfied with the improvement that I have made in my research and communication skills under the guidance of Dr. Maciejewski.

The mathematical details in this dissertation would not have been complete without the help of Dr. Rodney Roberts from FAMU-FSU. I appreciate his patience, as he communicated with me over the phone for our research for almost four years. He was a constant source of inspiration to me and I wish I could add him in my committee as my co-advisor.

I would like to thank Dr. Edwin Chong, Dr. Bruce Draper, and Dr. Iuliana Oprea for their suggestions to improve my research work. I would also like to express my deep gratitude to Mr. Tom Aurand and Mr. Rodrigo Jamisola for helping me with several software and hardware related issues.

I owe a special thanks to Dr. Steven Schaeffer and Mr. Bryce Eldridge for their help in generating artificial and real images that were required for the research in the second half of my dissertation. I would also like to thank all my well wishers and the ECE office staff including Ms. Karen Bross, Ms. Andrea Leland, and Ms. Elizabeth Wadman, for their constant moral support.

Last but not the least, I would like to thank my wife, Dr. Himali Saitwal, for being there with me during the tough period of my student life. I would not have been able to finish my graduate research without her emotional support.

AUTOBIOGRAPHY

I was born in Pune, India, on Jan. 18, 1977. I received my Bachelor of Engineering (B.E.) degree in Instrumentation and Controls from Vishwakarma Institute of Technology, Pune University, India, in May 1998. I was a proud recipient of the National Talent Search scholarship and was ranked in top 2% in the University during my undergraduate studies. I completed my Masters and Ph.D. with Electrical and Computer Engineering (ECE) Department at Colorado State University (CSU), USA, in Dec. 2001 and May 2006, respectively. During my graduate research, I was honored with the best poster presentation award at IEEE Joint Conference on Neural Networks (IJCNN), 2001 and my paper was nominated for the best student paper award at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004. I served as a chairman of the IEEE Student Branch Chapter (formed at CSU) of Robotics and Automation Society from Jan. 2005 to May 2006. I have joined Behavioral Recognition Systems (BRS), Inc. in Houston, Texas, where I am working as a Computer Vision Engineer, Development and Research since May 1, 2006. My research interests include image/video processing, computer vision, video surveillance, and multimedia.

To my Mother.

LIST OF SYMBOLS

$\mathcal{X}$ = image matrix

$\mathbf{x}$ = image vector

X = image data matrix

h = number of pixels in one row of each image

v = number of pixels in one column of each image

m = total number of pixels ($h \times v$) in each image

n = total number of images in an image data set

$\bar{\mathbf{x}}$ = average image vector

$\bar{X}$ = average image data matrix

$\check{X}$ = "unbiased" (average subtracted) image data matrix

R = sample correlation matrix

C = sample covariance matrix

$\lambda_i = i^{\text{th}}$ eigenvalue

$\hat{\mathbf{u}}_i = i^{\text{th}}$ true eigenimage

$\sigma_i = i^{\text{th}}$ true singular value

$\hat{\mathbf{v}}_i = i^{\text{th}}$ true right singular vector

U = orthogonal matrix that consists of $\hat{\mathbf{u}}_i$'s as its columns

Σ = matrix containing σ_i's in their descending order as its diagonal elements

V = orthogonal matrix that consists of $\hat{\mathbf{v}}_i$'s as its columns

$U_{m \times n}$ = matrix that consists of the first n columns of U

$\Sigma_{n \times n}$ = diagonal matrix containing n σ_i's in their descending order as its diagonal elements

$\tilde{U}$ = matrix with approximate basis for U

$\tilde{\tilde{U}}$ = matrix with orthonormal approximate basis for U

$\tilde{\hat{\mathbf{u}}}_i = i^{\text{th}}$ approximated $\hat{\mathbf{u}}_i$

$\tilde{\sigma}_i = i^{\text{th}}$ approximated σ_i

$\tilde{\hat{\mathbf{v}}}_i = i^{\text{th}}$ approximated $\hat{\mathbf{v}}_i$

$\|\cdot\|_F$ = Frobenius norm

$\|\cdot\|_2$ = 2-norm

$\Delta\sigma_i$ = difference between the true and the approximated singular values

Δu_i = angle between individual true and approximated eigenimages

Δv_i = angle between individual true and approximated right singular vectors

Δ = rotation index between the two subspaces

γ = subspace criterion

ρ = energy recovery ratio

$\theta_i = i^{\text{th}}$ principal angle between two subspaces

μ, ϵ = user-specified thresholds

k = dimension of the approximate eigenspace

k^* = dimension of the true eigenspace

θ = Givens rotation angle

$F(\mathbf{e})$ = Rayleigh quotient

$\mathbf{e}$ = eigenvector

$\mathcal{K}_i = i^{\text{th}}$ Krylov subspace

${}^{(q)}\mathbf{e}_{i(m)} = i^{\text{th}}$ vector associated with q dimensional image vector, but enlarged to size m

${}^{(q)}E_{(m\times n)}$ = matrix that consists of ${}^{(q)}\mathbf{e}_{i(m)}$ as its columns (n columns)

$\hat{A}$ = matrix that consists of orthonormal columns

r = image reduction factor

$d = m - r + 1$

X_h = high-resolution X

$U_h, \Sigma_h, V_h = U$, Σ, V matrices for X_h

X_l = low-resolution X

$U_l, \Sigma_l, V_l = U$, Σ, V matrices for X_l

$\text{circ}(a_1, a_2, a_3, ..., a_n)$ = circulant matrix

Π = permutation matrix

F = Fourier matrix

$\mathbf{f}_i = i^{\text{th}}$ column of F

H = matrix consisting of successively higher frequencies (derived from F)

$\mathbf{h}_i = i^{\text{th}}$ column of H

p = number of harmonics required to achieve given energy recovery ratio for X

H_p = matrix consisting of the first p columns of H

Y = matrix whose i^{th} row is FFT of the i^{th} row of X

$X_b = X_l$ with box filtered images

$U_b, \Sigma_b, V_b = U$, Σ, V matrices for X_b

$X_r = X_l$ with images containing randomly selected pixels

$U_r, \Sigma_r, V_r = U$, Σ, V matrices for X_r

s = total number of columns in $\tilde{\hat{U}}$

$\breve{X}$ = three-dimensional image data set

$\breve{F}$ = 3D Fourier matrix

$\breve{H}$ = matrix containing successively higher 3D frequencies (derived from $\breve{F}$)

α_l, β_m, γ_n = three parameters that are used to characterize $\breve{X}$

α, β, γ = frequencies along α_l, β_m, γ_n

$L \times M$ = number of camera locations in spherical patch above object

N = number of images captured at each camera location

$L \times M \times N$ = total number of images in a 3D image data set

$\mathbf{x}_{lmn}$ = row-scanned image of an object taken from camera location (l, m) at image plane rotation n

$\text{fft}_i(\cdot)$ = FFT computed along i^{th} dimension

CHAPTER I

INTRODUCTION

1.1 Background

A major goal of computer vision problems is to automatically deduce information about a three-dimensional scene from two-dimensional images. Humans can interpret such scenes without much effort, however, this is a hard task for a computer and this has been an active research area for around 40 years. While general purpose vision systems are not yet realizable, computer vision has been applied widely on tasks in controlled environments, for example, visual inspection (including inspection of part dimensions, agricultural objects, material surfaces, printed circuit boards, etc.), automation (including assembly, material handling, spray coating, and welding), character recognition, medical diagnosis, remote sensing and traffic analysis [1–3]. Recently, human face and facial expression recognition have also gained considerable attention.

A fundamental problem in computer vision is the recognition and localization of three-dimensional objects from two-dimensional images. Most of the proposed methods for industrial applications are model-based [1], where recognition involves comparing the input image against the precompiled description of the objects. The recognition usually consists of three steps, i.e., image acquisition, image analysis, and image comprehension. Image acquisition involves capturing an image with a camera and converting it into digital format for further processing, while image analysis involves detection of the features (such as edges and boundaries) or the shapes of the objects in the scene. Finally, image comprehension tries to match the features or the shapes detected from the previous step with those of the pre-defined models of the objects.

Model-based object recognition methods can be categorized into three classes based on

the dimension of the spatial description of objects, i.e., 2-D, $2\frac{1}{2}$-D, and 3-D descriptions [1]. A 3-D description gives an object-centered, viewpoint-independent, and volumetric representation of the object, such as with CAD models, while a 2-D description gives a viewer-centered representation in the image space, e.g., shape features derived from the image of an object. An object description is considered $2\frac{1}{2}$-D if it is a viewer-centered representation, but depends on local surface properties of the object in each view, such as range (depth) and surface orientation.

This thesis considers a class of object recognition techniques called "eigenspace methods," which fall into the class of 2-D descriptions. Eigenspace methods represent one computationally efficient approach for dealing with object recognition problems. It differs from other 2-D methods in that no feature or shape information is extracted from the image; it is purely based on the appearance of the objects. Note that the content of an image is affected not only by the features of objects in the image but also by environmental factors such as the illumination conditions; therefore subspace methods have the potential to be more robust. Eigenspace methods have been variously referred to as subspace methods, singular value decomposition (SVD) methods, principal component analysis methods and Karhunen-Loeve transformation methods [4, 5]. At first they were used for image coding and image compression on single images [6–8], but more recently, they have been applied to sets of highly correlated images to take advantage of the fact that these types of images can be approximately represented by a small set of their "eigenimages." The applications include face characterization [9,10] and recognition [11–15], lip-reading [16,17], object recognition [18–21], pose estimation [22–26], visual tracking [27,28] and inspection [29–32].

Eigenspace methods for object recognition problems can be broken down into two phases, i.e., a training phase and a matching phase. The training phase involves coarsely sampling the possible appearances of an object and then calculating a low-dimensional subspace, called the eigenspace, which best distinguishes the training images. Each training image is then represented as a point in the eigenspace, which is normally done off-line. The matching phase involves representing the test image as a point in the eigenspace and finding the best match from points representing the training images. Because of the relatively low dimension

of the eigenspace, this can be done efficiently. However, the off-line calculation required to determine both the appropriate number of eigenimages as well as the eigenimages themselves can be prohibitively expensive. This issue has been previously addressed by several different approaches.

One class of techniques relies on finding the eigenimages iteratively. The variants of the SVD power method, which calculates the dominant singular values and vectors one at a time, are discussed in [33–37]. In [38], Vogel et al. considered the block power method and the Lanczos method to solve the SVD of ill-posed problems. These methods, unlike the power method, iterate with k pairs of singular vectors at a time instead of one. The gradient-type algorithms [39,40] recast the search for the dominant singular vectors into an optimization problem that is solved using gradient or conjugate gradient methods. There are other iterative methods that work on symmetric matrices [40,41] that have been applied to either X^TX or XX^T.

Another class of techniques relies on updating a small set of eigenimages by recursively adding one image at a time. Murakami et al. [42] illustrated a method for updating a fixed number of eigenimages. Chandrasekaran's method [43], on the other hand, adaptively changed the number of eigenimages calculated.

Other techniques include Murase et al. [44] that compute the eigenimages of the approximated matrix X^TX in the discrete cosine transform domain. Chang et al. [45] used a fundamentally different algorithm (refer to Chapter 4) that was motivated by the fact that for a set of planar rotated images, the matrix X^TX is a circulant matrix and the (unordered) SVD of X for this case is known in closed form. However, the major drawback of all these approaches is that their computation time is highly dependent on the resolution of the original images.

The first part of this work addresses the computational expense of computing the desired eigenimages by exploiting the spatial correlation of image sequences. For this purpose, two different image reduction methods are considered, i.e., traditional low-pass filtering techniques and random sampling. The theoretical/mathematical analysis explains why reduction by random sampling can be more effective than any low-pass filtering technique.

This analysis along with the low-resolution properties of correlated images motivated several changes to Chang's algorithm, outlined in Chapter 4, to quickly compute the desired portion of the eigendecomposition based on a user-specified measure of accuracy without sacrificing the quality of the resulting eigenimages. The proposed algorithm is then applied successfully to a variety of objects and arbitrary video sequences.

Previous computationally efficient eigendecomposition algorithms considered exploiting the correlation between the successive images by putting the i^{th} image between $(i-1)^{\text{th}}$ and $(i+1)^{\text{th}}$ images. However, certain pattern recognition applications require that different views of an object taken from different 3D spatial camera locations be considered in the training data set [46–51] and such correlated images should be arranged in three-dimensional array instead of traditional one-dimensional ordering. Hence the existing algorithms, which do not consider the correlation along more than one dimension, cannot be directly applied to these data sets. The second part of this work shows that these three-dimensional (3D) image data sets can be characterized by three different parameters (two for camera locations in spherical patch above the object and one for image plane rotation to capture different views of the object at same camera location) and considers improving the computational efficiency of calculating the desired eigenimages of such 3D image data sets by exploiting their correlations along three dimensions. In particular, Chang's algorithm is successfully extended so that it can be effectively used for offline eigendecomposition calculations involved in three-dimensional pose estimation of objects.

1.2 Organization of this Study

The rest of this thesis is organized as follows:

Chapter 2 begins with the mathematical representation of correlated images. Then the eigenspace methods are reviewed to explain how these images can be approximately represented using a small number of eigenimages. Different measures that are used to compare different subspaces in this work are also explained. Lastly, a brief review of previous work that deals with the eigendecomposition of correlated images is given, along with the contributions of this work.

Chapter 3 explains the need for reducing the resolution of images and gives a theoretical background for quantifying the tradeoff associated with performing eigendecomposition on these images at lower resolutions in order to mediate the high computational expense of performing these calculations at high resolutions. Two different low-resolution estimates that are used in this work to approximate the high-resolution eigenimages are also explained, along with their advantages and disadvantages.

Chapter 4 gives an overview of Chang's eigendecomposition algorithm. The chapter begins with the analysis of images resulting from planar rotation and shows that the eigenimages of such images are known in closed form, based on the properties of circulant matrices. Generalization of these derived analytical expressions to 3-D cases is then explained. Lastly, Chang's algorithm and its computational expense are summarized.

Chapter 5 begins by discussing two different image reduction methods, i.e., low-pass filtering techniques and random sampling. Then a theoretical background is provided to explain why downsampling by random sampling can be more effective than any low-pass filtering technique. This analysis is then used to modify Chang's eigendecomposition algorithm to quickly compute the desired portion of the eigendecomposition based on a user-specified measure of accuracy. Examples are given at the end of this chapter to illustrate the quality of the results and the efficacy of the proposed method as compared to Chang's algorithm.

Chapter 6 explains the generation of fully general 3D image sets in which the correlated images are characterized by three different parameters instead of one and explains how Chang's algorithm can be extended to compute the SVD of such image sets. While doing so, it is shown how to efficiently compute the multiplication of the image data matrix with three-dimensional frequency combinations using FFT techniques. The optimum ordering of the frequency combinations based on the specifications of the given image data sets is also discussed. Supporting results are explained at the end of this chapter to show the promise of the proposed algorithm.

Chapter 7 gives the concluding remarks for this work and suggests a direction for future research.

CHAPTER II

PRELIMINARIES

In this chapter, the mathematical representation of images is introduced in the next section. Then the basic concepts of eigenspace methods are reviewed (refer to Section 2.2 and Section 2.3) and the different measures that can be used to compare different subspaces are discussed in Section 2.4. Finally, the previous works that address the problem of calculating the partial singular value decomposition (SVD) of large matrices are reviewed in Section 2.5, and the contributions of this work are summarized in Section 2.6.

2.1 Mathematical Representation of Images

An image is an $h \times v$ array of square pixels with intensity values normalized between 0 and 1. Thus, an image will be represented by a matrix $\mathcal{X} \in [0,1]^{h \times v}$. Because the images considered here are sets of related images, it will be convenient to represent an image equivalently as a vector, obtained simply by "row-scanning," i.e., concatenating the rows to obtain the *image vector* $\mathbf{x}$ of length $m = hv$ (refer to Fig. 1):

$$\mathbf{x} = \mathrm{vec}(\mathcal{X}^T).$$

The *image data matrix* of a set of images $\mathcal{X}_1, \ldots, \mathcal{X}_n$ is an $m \times n$ matrix, denoted X, and defined as

$$X = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n],$$

typically with $m \gg n$.

The *average image vector* is denoted $\bar{\mathbf{x}}$ and defined as

$$\bar{\mathbf{x}} = (\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_n)/n.$$

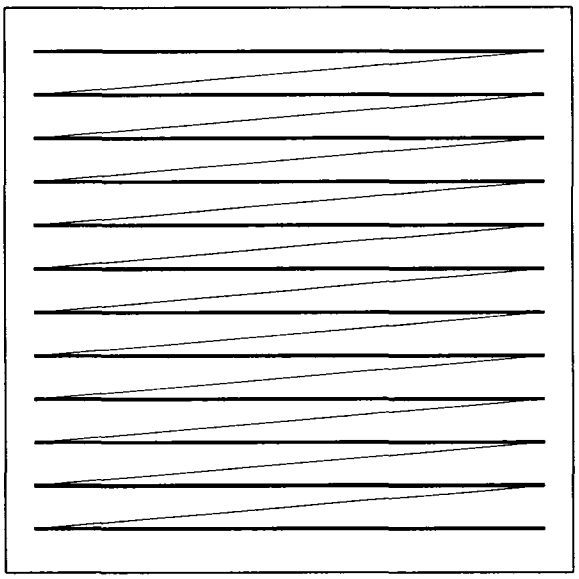


Figure 1: The image matrix $\mathcal{X}$ is row-scanned and concatenated to form the image vector $\mathbf{x}$. The scanning starts at the top leftmost pixel and proceeds to the bottom rightmost pixel.

The corresponding $m \times n$ *average image data matrix*, denoted $\bar{X}$, is

$$\bar{X} = [\bar{\mathbf{x}}\ \bar{\mathbf{x}}\ \cdots\ \bar{\mathbf{x}}]\,.$$

The matrix $X - \bar{X}$, which is denoted $\check{X}$, has the interpretation of an "unbiased" image data matrix.

2.2 The Concept of an Eigenspace

The eigenimages of the images $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n$ are defined as either the eigenvectors of the sample covariance matrix $C = \frac{1}{n}\check{X}\check{X}^T$ [11, 29] or the eigenvectors of the sample correlation matrix $R = \frac{1}{n}XX^T$ [42–44]. The eigenspace method is also referred to as singular value decomposition (SVD), principle component analysis (PCA) or the Karhunen-Loeve (KL) Expansion [4]. The motivation for using the eigenspace method is to reduce the amount of computation required for image representation or image comparison by representing images as linear combinations of the eigenimages.

In this section, two general issues will be considered: (1) Which k-dimensional subspace can best represent a set of n images? (2) Which k-dimensional subspace can best distinguish

a set of n images? It will be shown that for the first case, the optimal subspace is spanned by the eigenvectors of R. For the second case, the optimal subspace is spanned by the eigenvectors of C.

2.2.1 Image Representation

This subsection addresses the issue of what k-dimensional subspace can best represent a set of n images. The assumptions and derivations are summarized as follows (the details can be found in [4]):

Let $\mathbf{x}_t$ be a vector generated with the same statistics as the image vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$. Then this vector $\mathbf{x}_t$ can be represented without error using

$$\mathbf{x}_t = \sum_{j=1}^{m} (\mathbf{x}_t^T \boldsymbol{\phi}_j)\boldsymbol{\phi}_j \tag{1}$$

where the $\boldsymbol{\phi}_j$s form an orthonormal set. Consider an approximation of $\mathbf{x}_t$, denoted $\tilde{\mathbf{x}}_t$, given by

$$\tilde{\mathbf{x}}_t(k) = \sum_{j=1}^{k} (\mathbf{x}_t^T \boldsymbol{\phi}_j)\boldsymbol{\phi}_j \tag{2}$$

The corresponding mean-square error in this approximation is given by[1]

$$\begin{aligned} \bar{\varepsilon}^2(k) &= E\left\{\|\mathbf{x}_t - \tilde{\mathbf{x}}_t(k)\|^2\right\} \\ &= \sum_{j=k+1}^{m} E\left\{(\mathbf{x}_t^T \boldsymbol{\phi}_j)^2\right\} \\ &= \sum_{j=k+1}^{m} \boldsymbol{\phi}_j^T E\{\mathbf{x}_t \mathbf{x}_t^T\} \boldsymbol{\phi}_j \\ &\approx \sum_{j=k+1}^{m} \boldsymbol{\phi}_j^T R \boldsymbol{\phi}_j \end{aligned} \tag{3}$$

where the correlation matrix $E\{\mathbf{x}_t \mathbf{x}_t^T\}$ is replaced by the sample correlation matrix $R = XX^T$. The optimal $\boldsymbol{\phi}_j$s, i.e., the eigenimages for this case, can be shown to satisfy

$$R\boldsymbol{\phi}_j = \lambda_j \boldsymbol{\phi}_j \tag{4}$$

which are the eigenvectors of R, with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$ and $\lambda_i = 0$ for $i > n$. (This is true because the image data matrix X is at most rank n.) The minimum mean-square

[1]Norms in all the equations in this thesis always represent the 2-norm unless otherwise stated.

error becomes

$$\bar{\varepsilon}^2(k)_{opt} = \sum_{j=k+1}^{n} \lambda_j. \tag{5}$$

Note that (2) is optimal for the cost function given in (3) with $\mathbf{x}_t$ approximated by (2). The whole set of image vectors, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, can thus be "compressed" by storing the first k eigenimages $\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_k$, and the projections of $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ onto the first k eigenimages, which is roughly $k+1$ images, versus the n original images that would need to be stored.

2.2.2 Image Comparison

Consider the case when a set of n training images, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, is obtained by coarsely sampling the possible appearances of an object and a test image $\mathbf{x}_t$ generated with the same statistics as the training images is compared against each of the training images to find the best match (this is typical for pose detection problems). Let $\mathbf{x}_q$ be any of the training images. The "distance" between the two images $\mathbf{x}_t$ and $\mathbf{x}_q$ is usually measured as $\|\mathbf{x}_t - \mathbf{x}_q\|^2$. This process can be performed more efficiently by using the eigenspace representation, i.e., by measuring $\|\tilde{\mathbf{x}}_t(k) - \tilde{\mathbf{x}}_q(k)\|^2$ instead. Here $\tilde{\mathbf{x}}_t(k)$ and $\tilde{\mathbf{x}}_q(k)$ are approximations of $\mathbf{x}_t$ and $\mathbf{x}_q$, respectively, using (2).

It is desirable to have the approximated measurement $\|\tilde{\mathbf{x}}_t(k) - \tilde{\mathbf{x}}_q(k)\|^2$ be as close to $\|\mathbf{x}_t - \mathbf{x}_q\|^2$ as possible, in the least square sense. This requirement can be formed into another optimization problem, with the cost function given by

$$\begin{aligned}
\bar{\varepsilon}^2(k) &= E\left\{\|\mathbf{x}_t - \mathbf{x}_q\|^2 - \|\tilde{\mathbf{x}}_t(k) - \tilde{\mathbf{x}}_q(k)\|^2\right\} \\
&= E\left\{\sum_{j=1}^{m}(\mathbf{x}_t^T\boldsymbol{\phi}_j - \mathbf{x}_q^T\boldsymbol{\phi}_j)^2 - \sum_{j=1}^{k}(\mathbf{x}_t^T\boldsymbol{\phi}_j - \mathbf{x}_q^T\boldsymbol{\phi}_j)^2\right\} \\
&= \sum_{j=k+1}^{m} E\left\{(\mathbf{x}_t^T\boldsymbol{\phi}_j - \mathbf{x}_q^T\boldsymbol{\phi}_j)^2\right\} \\
&= \sum_{j=k+1}^{m} \boldsymbol{\phi}_j^T E\left\{(\mathbf{x}_t - \mathbf{x}_q)(\mathbf{x}_t - \mathbf{x}_q)^T\right\}\boldsymbol{\phi}_j \\
&= \sum_{j=k+1}^{m} \boldsymbol{\phi}_j^T E\left\{\mathbf{x}_t\mathbf{x}_t^T + \mathbf{x}_q\mathbf{x}_q^T - \mathbf{x}_t\mathbf{x}_q^T - \mathbf{x}_q\mathbf{x}_t^T\right\}\boldsymbol{\phi}_j \\
&= \sum_{j=k+1}^{m} \boldsymbol{\phi}_j^T \left(2E\{\mathbf{x}_t\mathbf{x}_t^T\} - 2\bar{\mathbf{x}}\bar{\mathbf{x}}^T\right)\boldsymbol{\phi}_j
\end{aligned}$$

$$\approx \sum_{j=k+1}^{n} 2\phi_j^T C \phi_j. \tag{6}$$

where the covariance matrix $E\left\{(\mathbf{x}_t - E\{\mathbf{x}_t\})(\mathbf{x}_t - E\{\mathbf{x}_t\})^T\right\}$ is replaced by the sample covariance matrix $C = \frac{1}{n}\check{X}\check{X}^T$. Following the same argument as for (4), the optimal ϕ_js in this case should satisfy

$$C\phi_j = \lambda_j \phi_j \tag{7}$$

which are the eigenvectors of C. Thus the norm of the distance between the test images $\mathbf{x}_t$ and the training images $\mathbf{x}_q$ can be approximated by

$$\|\mathbf{x}_t - \mathbf{x}_q\|^2 \approx \sum_{j=1}^{k} \left(\mathbf{x}_t^T \phi_j - \mathbf{x}_q^T \phi_j\right)^2, \tag{8}$$

and finding the best match of $\mathbf{x}_t$ against the training images can be approximated by

$$\min_{q=1,\ldots,n} \|\mathbf{x}_t - \mathbf{x}_q\|^2 \approx \min_{q=1,\ldots,n} \sum_{j=1}^{k} \left(\mathbf{x}_t^T \phi_j - \mathbf{x}_q^T \phi_j\right)^2. \tag{9}$$

Note that $\mathbf{x}_q^T \phi_j$ can be precomputed. Calculation of the right-hand side of (9) requires k m-dimensional inner products, followed by n norm computations of the difference between two k-dimensional vectors and hence it requires $O(km) + O(kn)$ flops. On the other hand, calculation of the left-hand side of (9) requires n norm calculations of the difference between two m-dimensional vectors and hence requires $O(nm)$ flops. In practice, $k \ll m$ and $k \ll n$, hence the left-hand side of (9) is typically computed approximately, by evaluating its right-hand side.

2.3 Eigendecomposition and the SVD

In the previous section, it was shown that the eigenimages are the eigenvectors of $R = \frac{1}{n}XX^T$ (or $C = \frac{1}{n}\check{X}\check{X}^T$). For applications of eigenspace methods such as object recognition and pose detection problems, the size of the image data matrix X is $m \times n$, where m (the number of pixels in each images) is typically much larger than n (the number of images). Therefore, forming the matrix R requires a large amount of memory space. Also, the cost of multiplying X with X^T is nm^2 flops, which should be avoided if possible.

A much more efficient way of calculating the eigenvectors of R is via singular value decomposition (SVD). The SVD of X is given by

$$X = U\Sigma V^T, \tag{10}$$

or

$$X = \sum_{i=1}^{n} \sigma_i \hat{\mathbf{u}}_i \hat{\mathbf{v}}_i^T \tag{11}$$

where $U = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, ..., \hat{\mathbf{u}}_m] \in I\!R^{m \times m}$ with $\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, ..., \hat{\mathbf{u}}_m$ being the left singular vectors of X and $V = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, ..., \hat{\mathbf{v}}_n] \in I\!R^{n \times n}$ with $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, ..., \hat{\mathbf{v}}_n$ being the right singular vectors of X. Both U and V are orthogonal. The matrix $\Sigma \in I\!R^{m \times n}$, with $\Sigma^T = [\Sigma_d \ \mathbf{0}_{n \times (m-n)}]^T$, where $\Sigma_d = \text{diag}(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ are the singular values of X. Many common algorithms that compute the complete SVD of a general matrix require $O(mn^2)$ flops.

Note that the sample correlation matrix $R = \frac{1}{n} XX^T = U\Sigma V^T V \Sigma U^T = U\Sigma^2 U^T$. That is, the eigenvectors of R are the left singular vectors of X. Similarly, the eigendecomposition of $X^T X$ yields $V\Sigma^2 V^T$ [42], thus the eigenvectors of $X^T X$ are the right singular vectors of X. Because left and right singular vectors are related by $XV = U\Sigma$, one can use XX^T or $X^T X$ to compute the SVD of X. However, there are three steps involved in this before the full SVD of X can be obtained, i.e., multiplication of X and X^T, computation of the eigenvectors of the resulting matrix, and computation of the remaining set of singular vectors. Although the second step can be computationally efficient, the cost of the remaining two steps can be expensive. Therefore, these techniques are generally not much faster than applying the SVD directly on X.

Each part of the SVD of X in (10) can be interpreted as follows: The left singular vectors (eigenimages) provide an orthonormal basis for the span of $\mathbf{x}_i$, ordered in terms of importance; the corresponding singular values measure how "aligned" the vectors $\mathbf{x}_i$ are, with the associated eigenimage, i.e., $\|\hat{\mathbf{u}}_i^T X\| = \sigma_i$. The components of the i^{th} column of V measure how much each individual image contributes to the i^{th} eigenimage. Compared with the SVD of X, the eigendecomposition of R gives only the left singular vectors and singular values of X, the information contained in the right singular vectors is lost during

forming the sample correlation matrix $R = \frac{1}{n}XX^T$. In the following chapters, it will be shown that the frequency analysis of the right singular vectors of X plays an important role in the development of a new algorithm which can calculate the approximated eigenimages efficiently.

The equations relating left and right singular vectors of X can also be written as $XV = U_{m\times n}\Sigma_{n\times n}$. In this matrix equation, $\Sigma_{n\times n}$ is a diagonal matrix that gives the singular values of X as its diagonal elements and $U_{m\times n}$ consists of the left singular vectors of X as its orthonormal columns. Because V is an orthogonal matrix, the *thin SVD* of X can be given by

$$X = U_{m\times n}\Sigma_{n\times n}V^T. \tag{12}$$

It is easy to see that the first n columns of U are the same as the n columns of $U_{m\times n}$ with the same corresponding singular values. One can also observe that the last $m - n$ columns of U are not part of the range space of X. Hence the two expressions for singular value decomposition given in (10) and (12) are interchangeable when $m \geq n$.

2.4 Difference Measures for SVDs

This work addresses one computationally efficient method to determine the first k approximate eigenimages of correlated images computed to a user-specified accuracy. Before getting into its details, it is important to discuss how to (1) determine the needed dimension k of the eigenspace and (2) measure the quality of the eigenimages calculated (if they are calculated approximately). A commonly used criterion [29] for determining the dimension of the eigenspace needed is given by

$$\frac{\sum_{i=1}^{k}\lambda_i^2}{\sum_{i=1}^{n}\lambda_i^2} \geq T \tag{13}$$

where the left-hand side is a measure of what percent of the total energy in X is contained in the first k eigenimages. (The authors suggested selecting the threshold T to be around 90 percent for object recognition and pose detection problems.) To determine the dimension k, the singular values are needed, which are not available *a priori*. Also, for a given threshold T, the dimension k may be different from object to object. In short, it is needed to compute

the SVD of correlated images before one can find the first k eigenimages to the user-specified accuracy.

In practice, the singular values and the corresponding singular vectors are not known or computed exactly, and instead their estimates are used. Hence it is important to define appropriate comparison criteria that can measure the errors between the true and approximated eigenspaces. This section defines six such error measures that are relevant to a user's motivation for preforming an eigendecomposition.

2.4.1 Difference Between Singular Values

The simplest error measure considered is the difference between the true and the approximated singular values calculated for a set of correlated images. These singular values can be compared directly if they are computed at the same spatial resolution. However, if the images are reduced spatially before performing the eigendecomposition, then the approximated singular values need to be properly scaled up before calculating this measure (refer to Section 3.2.2 and Section 3.2.3 for scaling), which is given by

$$\Delta\, \sigma_i = \sigma_i - \tilde{\sigma}_i, \tag{14}$$

where σ_i and $\tilde{\sigma}_i$ are the i^{th} true and approximated (scaled if required) singular values of the image data set, respectively.

2.4.2 Angles Between Singular Vectors

True and approximated singular vectors calculated for a set of correlated images can also be compared by calculating the angle between the corresponding vectors. Recall that $U = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, ..., \hat{\mathbf{u}}_m] \in I\!R^{m \times m}$ and $V = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, ..., \hat{\mathbf{v}}_n] \in I\!R^{n \times n}$. Hence if the images are reduced spatially, then the left singular vectors need to be enlarged and renormalized before applying this measure. On the other hand, if the number of images are reduced, the approximated right singular vectors need to be enlarged and renormalized before applying this measure. The angle between the two unit vectors can be calculated by performing their dot product. Thus the cosines of the angles between the singular vectors are given by

$$\Delta\, u_i = \hat{\mathbf{u}}_i^T \tilde{\hat{\mathbf{u}}}_i \tag{15}$$

and

$$\Delta\, v_i = \hat{\mathbf{v}}_i^T \tilde{\mathbf{v}}_i \tag{16}$$

where $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{v}}_i$ are the i^{th} true left and right singular vectors, respectively, while $\tilde{\hat{\mathbf{u}}}_i$ and $\tilde{\hat{\mathbf{v}}}_i$ are the corresponding approximated left and right singular vectors of the image data set, respectively.

2.4.3 Residue Between Subspaces

The previous error measure calculates the angles between the individual singular vectors. However the i^{th} approximated singular vector may not be aligned with the i^{th} true singular vector even though the subspaces containing the first k vectors may span the same vector space. Hence another error measure is used to calculate the rotation between these subspaces. The possibility that the data matrix $B \in \mathbb{R}^{m\times k}$ can be rotated into the data matrix $A \in \mathbb{R}^{m\times k}$ is explored [52] by solving the problem

$$\Delta = \min_{Q} ||A - BQ||_F \tag{17}$$

where $||\cdot||_F$ represents the Frobenius norm, $Q \in \mathbb{R}^{k\times k}$ is an orthogonal matrix, and Δ is the residue. Expanding the right-hand side results in

$$\begin{aligned} ||A - BQ||_F^2 &= \text{tr}(A^TA) + \text{tr}(B^TQ^TQB) - 2\text{tr}(Q^TB^TA) \\ &= \text{tr}(A^TA) + \text{tr}(B^TB) - 2\text{tr}(Q^TB^TA). \end{aligned} \tag{18}$$

Thus, (17) is equivalent to maximizing $\text{tr}(Q^TB^TA)$. The maximizing Q can be found by calculating the SVD of $C = B^TA$. Thus,

$$\text{tr}(Q^TC) = \text{tr}(Q^TU_c\Sigma_cV_c^T) = \text{tr}(Z\Sigma) = \sum_{i=1}^{k} z_{ii}\sigma_{ci} \leq \sum_{i=1}^{k} \sigma_{ci}, \tag{19}$$

where $Z = V_c^TQ^TU_c$, U_c and V_c are the matrices containing the left and right singular vectors of C, respectively, while Σ_c is a diagonal matrix containing the singular values of C in descending order. Thus, the upper bound is attained by setting $Q = U_cV_c^T$.

To summarize the algorithm, the Q that minimizes $||A - BQ||_F$ (denoted by Q_{min}) in (17) can be calculated as follows:

- Form the matrix $C = B^T A$.
- Compute the SVD of C, i.e., $C = U_c \Sigma_c V_c^T$.
- Find the orthogonal matrix $Q_{min} = U_c V_c^T$.

The residue Δ after solving (17) using above Q_{min} will satisfy

$$\Delta^2 = \mathrm{tr}(A^T A) + \mathrm{tr}(B^T B) - 2\sum_{i=1}^{k} \sigma_{ci}. \tag{20}$$

The smaller the residue Δ, the closer A and B are to representing the same subspace. To determine the rotation between two sets of eigenimages, let

$$\begin{aligned} A &= U_k, \\ B &= \tilde{U}_k \end{aligned} \tag{21}$$

where U_k and $\tilde{U}_k$ are the matrices containing the first k true and approximated (with proper preprocessing) eigenimages as their columns, respectively. The subspaces containing true and approximated right singular vectors can similarly be compared using (20) and (21).

It is easy to see that two properties are evident for this measure: (1) the two subspaces A and B need to be of the same dimension before this measure can be applied and (2) if these two subspaces have orthonormal columns, then the residue in (20) can be given by

$$\Delta^2 = 2\left(k - \sum_{i=1}^{k} \sigma_{ci}\right). \tag{22}$$

2.4.4 Angles Between Subspaces

Two sets of singular vectors that represent two subspaces, both having dimension k, share k different *principal angles* [52] with the k^{th} *principal angle* giving the largest angle between the subspaces. In particular, let $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{m \times k}$ be orthogonal matrices representing two subspaces. Then the principal angles $\theta_1, \theta_2, \cdots, \theta_k \in [0, \pi/2]$ between A and B are defined recursively by

$$\cos(\theta_k) = \max_{\mathbf{a} \in A} \max_{\mathbf{b} \in B} \mathbf{a}^T \mathbf{b} = \mathbf{a}_k^T \mathbf{b}_k. \tag{23}$$

Note that the principal angles satisfy $0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_k \leq \pi/2$. The vectors $\{\mathbf{a}_1, \cdots \mathbf{a}_k\}$ and $\{\mathbf{b}_1, \cdots \mathbf{b}_k\}$ are called the *principal vectors* between the subspaces A and B.

The largest principal angle is related to the notion of the distance between the equidimensional subspaces, i.e., $\text{dist}(A, B) = \sqrt{1 - \cos(\theta_k)^2} = \sin(\theta_k)$. If the matrices representing the subspaces are orthogonal, then this measure is closely related to the previous measure concerning the residue between subspaces. In particular, the Σ_c matrix containing the singular values of C gives the *principal angles* between the subspaces, i.e.,

$$\text{diag}(\cos(\theta_1), \cdots, \cos(\theta_k)) = \Sigma_c \tag{24}$$

where θ_i is the i^{th} *principal angle.* If the maximum angle θ_k is close to 0, then the two subspaces are considered to be closely aligned with each other. It can be noted that the two subspaces A and B do not need to be of the same dimension, i.e., if $A \in \mathbb{R}^{m \times k_1}$ and $B \in \mathbb{R}^{m \times k_2}$, where $k_1 < k_2$, then there are essentially k_1 principal angles between A and B, with θ_{k_1} being the largest.

True and approximated left and right singular vectors, computed in this study, can be compared by defining A and B as in (21).

2.4.5 Energy Recovery Ratio

True and approximated eigenimages of X can also be compared in terms of their capability of recovering the amount of the total energy in X. If the approximated eigenimages are ordered as per their importance, the first eigenimage ($\tilde{\mathbf{u}}_1$) will give the following inequality:

$$\begin{aligned} \|\tilde{\mathbf{u}}_1^T X\|^2 &= \|\sum_{j=1}^{n} \left(\sigma_j \tilde{\mathbf{u}}_1^T \hat{\mathbf{u}}_j \hat{\mathbf{v}}_j^T\right)\|^2 \\ &= \sum_{j=1}^{n} (\sigma_j \tilde{\mathbf{u}}_1^T \hat{\mathbf{u}}_j)^2 \|\hat{\mathbf{v}}_j^T\|^2 \\ &= \sum_{j=1}^{n} (\sigma_j \tilde{\mathbf{u}}_1^T \hat{\mathbf{u}}_j)^2 \\ &\leq \sigma_1^2. \end{aligned} \tag{25}$$

Note that the maximum of $\|\tilde{\mathbf{u}}_1^T X\|^2$ is achieved when $\tilde{\mathbf{u}}_1 = \hat{\mathbf{u}}_1$. This suggests a more general measure, the "energy recovery ratio," for approximated eigenimages of X, which is defined

as

$$\rho(X, \tilde{\hat{\mathbf{u}}}_1, \tilde{\hat{\mathbf{u}}}_2, ..., \tilde{\hat{\mathbf{u}}}_k) = \frac{\sum_{i=1}^{k} \|\tilde{\hat{\mathbf{u}}}_i^T X\|^2}{\|X\|_F^2}. \tag{26}$$

Note that $\rho(X, \hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, ..., \hat{\mathbf{u}}_k)$ is the same as the left-hand side of (13). From the theory of principle component analysis, the true eigenimages yield the highest energy recovery ratio. The quality of the approximated eigenimages can be obtained from comparing the energy recovery ratio of the approximated eigenimages with that of the true ones.

For the comparison of true and approximated right singular vectors, the measure in (26) can be modified as

$$\rho(X^T, \tilde{\hat{\mathbf{v}}}_1, \tilde{\hat{\mathbf{v}}}_2, ..., \tilde{\hat{\mathbf{v}}}_k) = \frac{\sum_{i=1}^{k} \|X\tilde{\hat{\mathbf{v}}}_i\|^2}{\|X\|_F^2} \tag{27}$$

and the quality of the approximated right singular vectors can be obtained from comparing the energy recovery ratio of the approximated ones with that of the true ones.

2.4.6 Subspace Criterion

True eigenimages give an optimum energy recovery ratio in (26). Hence, it is possible that more approximated eigenimages are required than the true ones to achieve the same energy recovery ratio. Hence another measure used in this study is the degree to which approximate eigenimages span the subspace of the first k^* true eigenimages, which will be referred to as the subspace criterion, γ, given by

$$\gamma = \sqrt{\frac{1}{k^*} \sum_{i=1}^{k} \sum_{j=1}^{k^*} (\tilde{\hat{\mathbf{u}}}_i \cdot \hat{\mathbf{u}}_j)^2}. \tag{28}$$

Consider $U_{k^*} = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \cdots, \hat{\mathbf{u}}_{k^*}]$ and $\tilde{U}_k = [\tilde{\hat{\mathbf{u}}}_1, \tilde{\hat{\mathbf{u}}}_2, \cdots, \tilde{\hat{\mathbf{u}}}_k]$. If the column space of U_{k^*} is included in that of $\tilde{U}_k$, then $\|U_{k^*}^T \tilde{\hat{\mathbf{u}}}_j\| = 1$ for $j = 1, 2, \cdots, k$. Hence, if the entire subspace of U_{k^*} is spanned by $\tilde{U}_k$, then $\gamma = 1$, otherwise $\gamma < 1$. Note that this measure is closely related to residue between subspaces (Δ) and angles between subspaces (θ_i) giving the similar qualitative difference, but different quantitative difference between true and approximated singular vector spaces.

The last four error measures provide slightly different information regarding the "quality" of the estimated eigenimages. The energy recovery ratio ρ, implicitly includes the effect of the singular values and thus weights the estimated eigenimages differently based on their

importance. In contrast, the residue between the subspaces Δ and subspace criterion γ are purely subspace measures. The principal angles θ_i that constitute Δ provide detailed information about how the estimated eigenspace is oriented relative to the true eigenspace.

2.5 Previous Work

The principal calculation required in eigenspace methods is the precomputation of estimates of the left singular vectors $\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_k$ of the $m \times n$ matrix X. This can be a computationally expensive operation when m and n are very large. Reducing this computational expense by taking advantage of the fact that only the principle singular vectors are of interest has been the subject of previous work.

One class of techniques relies on finding the eigenimages iteratively. One of the first few iterative techniques is SVD power method [33–36], which calculates the dominant singular values and vectors one at a time. The algorithm is relatively easy to implement: starting with a random vector $\mathbf{v}^{(0)}$, perform the iteration

$$\mathbf{u}^{(k+1)} = \frac{X\mathbf{v}^{(k)}}{\|X\mathbf{v}^{(k)}\|} \tag{29}$$

and

$$\mathbf{v}^{(k+1)} = \frac{X^T\mathbf{u}^{(k+1)}}{\|X^T\mathbf{u}^{(k+1)}\|} \tag{30}$$

until $\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\|$ becomes comparable to the preset threshold. To obtain the next pair of singular vectors, the singular vectors calculated in the previous stage are removed from X. That is, the matrix X is updated as $X' = X - \sigma_1\hat{\mathbf{u}}_1\hat{\mathbf{v}}_1^T$, and the same iteration is repeated with X' to find σ_2, $\hat{\mathbf{u}}_2$ and $\hat{\mathbf{v}}_2$. In [37], Shlien modified this power algorithm slightly, so that it will have better convergence properties. The cost of both these methods is $O(mnki)$ where k is the desired dimension of the eigenspace and i is the average number of iterations needed for each pair of singular vectors.

The gradient-type algorithms [39,40] recast the search for the dominant singular vectors into an optimization problem. By definition, the largest left singular vector of X is the unit vector that maximizes $\|X^T\mathbf{u}\|$, therefore the Rayleigh quotient $F(\mathbf{e})$, defined as

$$F(\mathbf{e}) = \frac{\|X^T\mathbf{e}\|^2}{\|\mathbf{e}\|^2}, \tag{31}$$

is maximized when $\mathbf{e}$ is colinear with $\hat{\mathbf{u}}_1$. The search for the maximum of $F(\mathbf{e})$ is through gradient or conjugate gradient methods. The cost of each iteration is on the order of mn, therefore the total cost is $O(mnki)$.

In [38], Vogel et al. considered the block power method and the Lanczos method to solve the SVD of ill-posed problems, i.e., large matrices with rapidly decaying singular values. The block power method, also known as simultaneous iteration, is similar to the power method except that it iterates with k pairs of singular vectors instead of with one pair at a time. The computational expense for this method is also $O(mnki)$.

The Lanczos method is a different approach to this problem. Let A be a symmetric matrix and $\mathbf{q}_0$ be an initial unit vector, then each iteration i of the Lanczos method can be viewed as a projection of the matrix A onto the i^{th} Krylov subspace

$$\mathcal{K}_i(\mathbf{q}_0) = \left\langle \mathbf{q}_0, A\mathbf{q}_0, ..., A^{i-1}\mathbf{q}_0 \right\rangle . \tag{32}$$

The matrix representing this projection is a symmetric $k \times k$ tridiagonal matrix T_i, where the eigenvalues of T_i converge rapidly to the extremal eigenvalues of A and the corresponding eigenvectors of T_i can be used to compute approximate eigenvectors of A. For a non-symmetric matrix X with size $m \times n$, one can apply the Lanczos method to the symmetric matrix

$$A = \begin{bmatrix} 0_{m\times m} & X \\ X^T & 0_{n\times n} \end{bmatrix} . \tag{33}$$

Note that forming A as given by (33) requires at least twice the memory space of storing X. The code listed in [38] shows a way of applying the Lanczos method on a non-square matrix X without forming the matrix A as in (33). The computational expense for this method is $O(mnki)$.

There are other iterative methods that work on symmetric matrices [40, 41] that have been applied to either X^TX or XX^T. (Note that the cost of this matrix multiplication is mn^2 or nm^2 flops, respectively.) If one applies (33), this requires twice as much memory space and hence neither approach is practical when m and n are large.

Another class of techniques relies on updating a small set of eigenimages by recursively adding one image at a time. Murakami et al. [42] illustrated a method for updating a

fixed number of eigenimages. If the total number of images is n and the desired number of eigenimages is k, then the eigendecomposition of the first $k+1$ images is calculated and the first k eigenimages are kept. Then one image is added at a time to update the k eigenimages. The updating can be done efficiently by taking advantage of the orthogonality of the eigenimages from the previous stage. The cost of this method is $O(mnk^2)$ and hence it has an advantage over the direct SVD algorithm when k^2 is smaller than n.

Chandrasekaran et al. [43] took a similar approach to that of [42]. The major difference is that the number of eigenimages calculated is adaptively changed. Instead of keeping only the first k eigenimages in each iteration, it is suggested to keep the eigenimages with the corresponding eigenvalues higher than a preset threshold. Also, when adding one image that does not change the eigenimages appreciably, the next one or several images may be skipped. The authors claimed that this method is as efficient as that of [42] when the required dimension of eigenimages is small.

An efficient method for quickly computing an approximate value for X^TX was developed by Murase et al. in [44]. (Recall that the eigenvectors of X^TX are the right singular vectors of X and the left singular vectors of X can be easily computed from the corresponding right singular vectors.) The discrete cosine transform (DCT) was applied to blocks of each image. The approximated matrix X^TX and its eigenvectors were found in the frequency domain (DCT), and then the inverse DCT was applied on the eigenvectors to transform them back to the spatial domain. This method is referred to as the spatial temporal adaptive method (STA). The number of multiplications required for applying this method on an $m \times n$ matrix to find the first k eigenimages is given by:

$$N_{STA} = 1.25mn + nm(1+\alpha^2)\beta_a + kin^2 + mn(1+k\alpha)\beta_b + 1.25kn, \tag{34}$$

where i is the average number of iterations, while α, β_a and β_b are constants. For their implementation, the authors reported that the speed is 6 to 10 times faster than the direct SVD algorithm[2] for calculating the first 8 eigenimages from a set of 256 images.

[2]The authors did not specify which algorithm they used to implement the direct SVD.

Chang et al. [45] proposed a fundamentally different algorithm to compute the eigenimages of correlated images (refer to Chapter 4 for its overview). The algorithm was motivated by the observation that for a set of planar rotated images (i.e., the i^{th} image of a set of n images is obtained from the first image by a planar rotation of $360(i-1)/n$ degrees), the matrix X^TX is a "circulant matrix". The (unordered) SVD of X for this case is known in closed form, where the right singular vectors are pure sinusoids and the left singular vectors can be calculated by applying FFT to the rows of X. For arbitrary video sequences, it was found through empirical evidence that the first k approximated eigenimages can be found using only a small number of frequencies, i.e., not much larger than k. The cost of Chang's algorithm is $O(mn\log_2 n)$, which compares favorably with the direct SVD algorithm.

2.6 Summary of Contributions

Currently, Chang's algorithm appears to be one of the fastest algorithms for computing the first k approximate eigenimages of correlated images to a user-specified accuracy. However, the calculation of the required number of harmonics and the computation of SVD of low-pass filtered version of X, still requires a significant amount of time because the algorithm always works with the full spatial resolution of the images. Hence it is desirable to reduce the images in the spatial dimension first. In the first part of this work, two different image reduction methods are considered, i.e., low-pass filtering techniques and random sampling. The effect that these techniques have on the spatial and the temporal properties of X is illustrated explaining why downsampling by random sampling can be more effective than any low-pass filtering technique.

Once the images are reduced, their low-resolution SVDs can be used to approximate the high-resolution eigenimages. Two techniques are considered for this purpose, i.e., interpolating the low-resolution eigenimages and using low-resolution right singular vectors. The analysis shows that the low-resolution right singular vectors give a "good" approximation of their high-resolution counterparts. It is also observed in [53] that the estimation of high-resolution SVDs with low-resolution right singular vectors is better than with low-resolution eigenimages. These observations motivated several modifications to Chang's algorithm that

improve its computational efficiency. In the proposed approach, random sampling technique was used to reduce X in the spatial dimension and then Chang's method was used to reduce it further in the temporal dimension to achieve improved computational efficiency. This algorithm is applied to several sets of rotationally correlated images and arbitrary video sequences. The results of the experiments show that the proposed algorithm is much faster than other methods, while the accuracy of approximation is comparable.

The second part of this work has illustrated an extension of Chang's algorithm to compute an estimate of the partial SVD of 3D image data sets, in which the images are characterized by three parameters instead of one. One important issue that needed to be addressed was related to the optimum ordering of the frequencies of 3D image data sets based on their energy recovery ability. In Chang's algorithm, it was observed that the ordering of the frequencies for 1D image data sets was trivial. However, the frequency combinations along three parameters for 3D image data sets are unordered in terms of their energy recovery ability. This issue was resolved in this work. Another issue was related to the implementation of the multiplication of the image data matrix with the 3D frequencies using FFT techniques. This issue was also addressed and Chang's algorithm was modified accordingly. The empirical results showed that this extension was computationally efficient in calculating the partial SVD of 3D image data sets commonly used in 3D pose estimation.

CHAPTER III

EFFECT OF RESOLUTION ON THE EIGENDECOMPOSITION

This chapter explains the necessity of reducing the spatial resolution of correlated images before computing their SVDs and discusses two different techniques that can be used to approximate the high-resolution eigenimages using a low-resolution SVD. In particular, the next section considers the importance of working at the lower resolution. Section 3.2 explains the two techniques and the preprocessing required of the approximated eigenimages before they can be compared with the true ones. The empirical results are also provided in this section to evaluate the two techniques. This empirical evaluation is then supported with a theoretical background for a simple example described in Section 3.3.

3.1 Introduction

Eigendecomposition-based techniques have been used extensively in a variety of applications such as face characterization and recognition, lip-reading, object recognition, pose detection, visual tracking, and inspection, etc. All of these applications are based on taking advantage of the fact that a set of highly correlated images can be approximately represented by a small set of eigenimages [45]. Once the set of principal eigenimages is determined, online computation using these eigenimages can be performed very efficiently. However, the offline calculation required to determine both the appropriate number of eigenimages as well as the eigenimages themselves can be prohibitively expensive.

The resolution of the given correlated images, in terms of the number of pixels, is one of the factors that greatly affects the amount of calculation required to compute an eigendecomposition. In particular, many common algorithms that compute the complete SVD of a general matrix require $O(mn^2)$ flops, where m is the total number of pixels in a

single image and n is the number of images. Most users of eigendecomposition techniques would like to use as high a resolution as is available for the original images in order to maintain as much information as possible; however, this frequently results in an impractical computational burden. Thus users are typically forced to downsample their images to a lower resolution using a "rule of thumb" or some *ad hoc* criterion to obtain a manageable level of computation.

The purpose of this chapter is to provide an analysis of how different approximation techniques using low-resolution SVDs affect the resulting eigendecomposition. This analysis can then be used to modify the fastest known eigendecomposition algorithm, proposed by Chang et al. [45], to improve its computational efficiency without sacrificing the quality of the resulting eigenimages.

3.2 Approximation of High-Resolution Eigenimages

As described earlier, it is important to quantify the tradeoff associated with performing the SVD on correlated images at lower resolutions in order to mediate the high computational expense of performing these calculations at high resolutions. It is also important to make sure that such approximation of high-resolution SVDs using low-resolution SVDs would not introduce too large an error. Hence the goal here is to find a "good" approximation of high-resolution eigenimages using a low-resolution SVD with better computational efficiency. Two different techniques that can be used for this purpose are considered in this section. The first technique works with the low-resolution eigenimages (refer to Section 3.2.2), while the second technique works with the low-resolution right singular vectors (refer to Section 3.2.3), in order to approximate high-resolution eigenimages. These two techniques are analyzed and evaluated using the empirical results in Section 3.2.4.

3.2.1 Comparison of SVD at Different Resolutions

Recall that the thin SVD of an $m \times n$ image data matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$ is given by $X = U_{m\times n}\Sigma_{n\times n}V^T$, where $\Sigma_{n\times n}$ is a diagonal matrix that contains the singular values of X as its diagonal elements, $U_{m\times n}$ consists of the eigenimages of X as its orthonormal columns, and V is an $n \times n$ matrix that consists of the right singular vectors of X as its orthonormal

columns.

The singular value decomposition of correlated images at different resolutions give different U, Σ and V matrices. These different sets of matrices can be compared against each other to study the effect of different resolutions on the SVD. To perform a meaningful comparison, the singular values and singular vectors of the low-resolution image data matrix must be modified. To distinguish both the resolution and the size of a singular vector, the notation ${}^{(q)}\mathbf{e}_{i(m)}$ is used, where the preceding superscript q denotes the fact that the vector is associated with q dimensional image vectors and the subscript m denotes the actual dimension of the vector $\mathbf{e}_i$. The matrix consisting of ${}^{(q)}\mathbf{e}_{i(m)}$ column is represented as ${}^{(q)}E_{(m\times n)}$.

Each eigenimage $\hat{\mathbf{u}}_i$ of X is an m-vector. As the resolution of the images is varied, the dimension of each eigenimage will also change accordingly. Hence the eigenimages computed at different resolutions, for e.g., ${}^{(q)}\hat{\mathbf{u}}_{i(q)}$ and ${}^{(m)}\hat{\mathbf{u}}_{i(m)}$, cannot be directly compared with each other. The following two subsections explain how the low-resolution SVD can be used to approximate the high-resolution eigenimages so that there can be a meaningful comparison between approximated and true eigenimages.

Each right singular vector $\hat{\mathbf{v}}_i$ of X is an n-vector. Thus the change in the resolution of the images does not affect the size of these vectors and the respective right singular vectors, e.g., ${}^{(q)}\hat{\mathbf{v}}_{i(n)}$ and ${}^{(m)}\hat{\mathbf{v}}_{i(n)}$, can be directly compared with each other, as long as the number of images for the given data set remains fixed for all resolutions[1].

The matrix $\Sigma_{n\times n}$ containing the singular values of X need not be resized before the comparison of singular values at different resolutions. However, due to the lower dimension of the low-resolution images, these values are scaled-down versions of those for the high-resolution images. Hence the singular values at low resolution must be scaled up properly before they can be compared with those at high resolution. These scaling methods are also explained with the two techniques described in the following subsections.

[1] In this study, the number of images in the given data set is fixed. Therefore the subscript n for the right singular vectors is dropped henceforth for notational simplicity.

3.2.2 Interpolation of Low-Resolution Eigenimages

This subsection considers the method of approximating high-resolution eigenimages by using low-resolution eigenimages. In particular, to compare the eigenimages at different resolutions, the eigenimages at lower resolutions can be enlarged to match in size with those at higher resolution. This can be performed by using a number of different interpolation techniques. Due to the enlargement and interpolation, the resulting eigenimages are typically no longer orthonormal. If ${}^{(q)}U_{(m\times n)}$ consists of all the interpolated eigenimages, ${}^{(q)}\mathbf{u}_{i(m)}$, as its columns, then QR decomposition can be carried out on this matrix, i.e.,

$$ {}^{(q)}U_{(m\times n)} = {}^{(q)}\hat{U}_{(m\times n)}R, \tag{35} $$

where R is an upper triangular matrix and the columns of ${}^{(q)}\hat{U}_{(m\times n)}$ are an orthonormal basis for the interpolated eigenimages in ${}^{(q)}U_{(m\times n)}$. These interpolated and orthonormal eigenimages in ${}^{(q)}\hat{U}_{(m\times n)}$ can now be compared with the true eigenimages in ${}^{(m)}\hat{U}_{(m\times n)}$.

The singular values of the correlated image data set determine the scaling of the associated eigenimages. Because the low-resolution eigenimages are enlarged to the size of high-resolution eigenimages, each low-resolution singular value should be scaled using

$$ {}^{(q)}\sigma_i = {}^{(q)}\tilde{\sigma}_i||{}^{(q)}\mathbf{u}_{i(m)}||, \tag{36} $$

where ${}^{(q)}\tilde{\sigma}_i$ represents the i^{th} low-resolution singular value that is to be compared with a higher resolution singular value.

Once the eigenimages and the corresponding singular values are properly scaled, true and approximated SVD can be compared using the difference measures defined in Section 2.4. For "residue between subspaces" and "angles between subspaces," the subspaces A and B used for the left singular vectors will be $A = {}^{(m)}\hat{U}_{(m\times k)}$ and $B = {}^{(q)}\hat{U}_{(m\times k)}$. Similarly the subspaces A and B used for the right singular vectors will be $A = {}^{(m)}V_k$ and $B = {}^{(q)}V_k$, where ${}^{(m)}V_k$ and ${}^{(q)}V_k$ consists of the first k right singular vectors for high-resolution and low-resolution image data matrices, respectively.

3.2.3 Using Low-Resolution Right Singular Vectors

High-resolution eigenimages can also be approximated using low-resolution right singular vectors. In particular, the high-resolution image data matrix X can be multiplied by ${}^{(q)}V$ to get the basis for the corresponding approximate high-resolution eigenimages, i.e.,

$$ {}^{(q)}U_{(m\times n)} = X \ {}^{(q)}V, \tag{37} $$

where the columns of ${}^{(q)}U_{(m\times n)}$ (denoted ${}^{(q)}\mathbf{u}_{i(m)}$) give the approximated eigenimages that are not orthonormal to each other. Thus QR decomposition can be carried out on this matrix, i.e.,

$$ {}^{(q)}U_{(m\times n)} = {}^{(q)}\hat{U}_{(m\times n)}R, \tag{38} $$

where the columns of ${}^{(q)}\hat{U}_{(m\times n)}$ are an orthonormal basis for ${}^{(q)}U_{(m\times n)}$. These orthonormal eigenimages in ${}^{(q)}\hat{U}_{(m\times n)}$ can now be compared with the true eigenimages in ${}^{(m)}\hat{U}_{(m\times n)}$.

The norms of the ${}^{(q)}\mathbf{u}_{i(m)}$'s can be used as an approximation of the corresponding singular values of X. This approximation can also be obtained from the matrix R, i.e.,

$$ {}^{(q)}\sigma_i = \|{}^{(q)}\mathbf{u}_{i(m)}\| = \sqrt{\sum_{j=1}^{i} R_{j,i}^2}. \tag{39} $$

The approximated eigenimages and the corresponding singular values can now be compared with the true ones using the difference measures defined in Section 2.4.

3.2.4 Empirical Results to Evaluate Two Techniques

The two techniques used to approximate high-resolution eigenimages are evaluated here using two different image data sets. The first image data set consists of 20 different objects that are rotated throughout 360° with 360 images obtained for each of them. Each image is of size 128 × 128 with a single image of each object shown in Fig. 2. The second image data set consists of images from successive frames of arbitrary video sequences. Specifically, there are 18 such video sequences that are used in this study. Each video sequence consists of 150 images; the first, middle, and last frames from each set are shown in Fig. 3.

The original image sizes for both the data sets are given in Table 1 in the second row (labelled "Index" = (0)). This "Index" gives the reduction factor for the original images

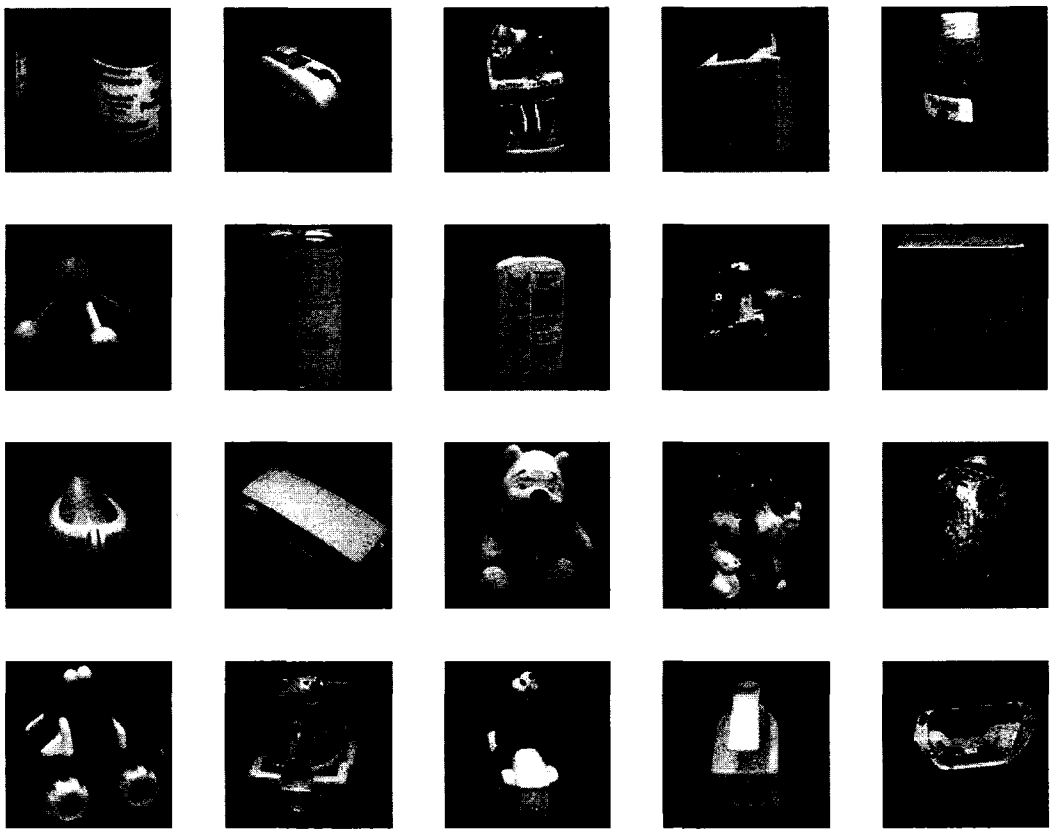


Figure 2: This figure shows the 20 objects that are used in this study. The objects are rotated throughout 360° and 360 images are obtained for each of them. Each image is of size 128×128, thus the resulting image data matrix X for each object is of size 16384×360. These objects are numbered from 1 through 20 from left to right and top to bottom.

(the larger the index, the larger the reduction of the original images). The high-resolution image data matrices X are formed for all 20 objects as well as all 18 video sequences using the original images and the corresponding "true" SVDs are computed. The images are then reduced to some lower resolutions using either bicubic interpolation or box filtering. The image sizes at the lower resolutions for both the data sets are given in Table 1 for the different reduction indices. Note that for the objects, the images are reduced with integer reduction factors all the way down to an index of (6), however, for the video sequences, the reduction factor is not always an integer value. Also, bicubic interpolation is used to reduce the original images until the reduced images no longer have enough pixels (at least

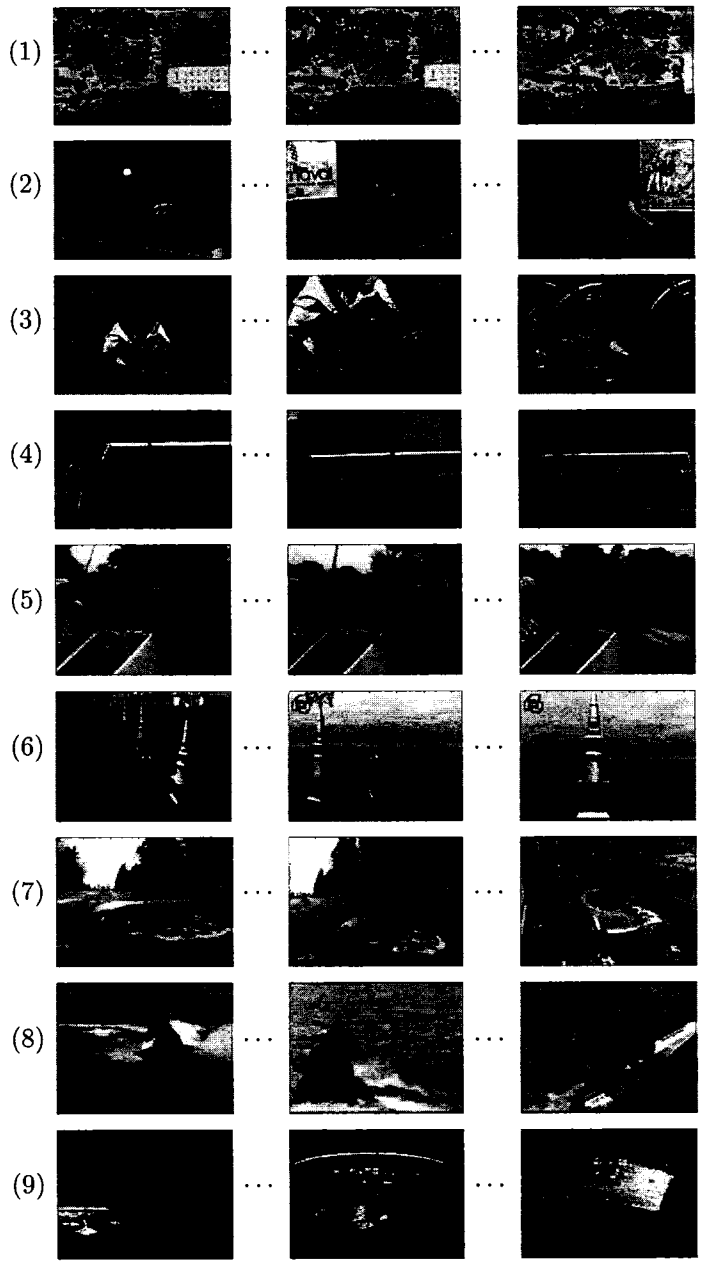


Figure 3: This figure shows the first, middle, and last frames of the first nine video sequences used in this study. Each video sequence consists of 150 images and the corresponding image sizes are given in Table 1 in the row labelled "Index" (0).

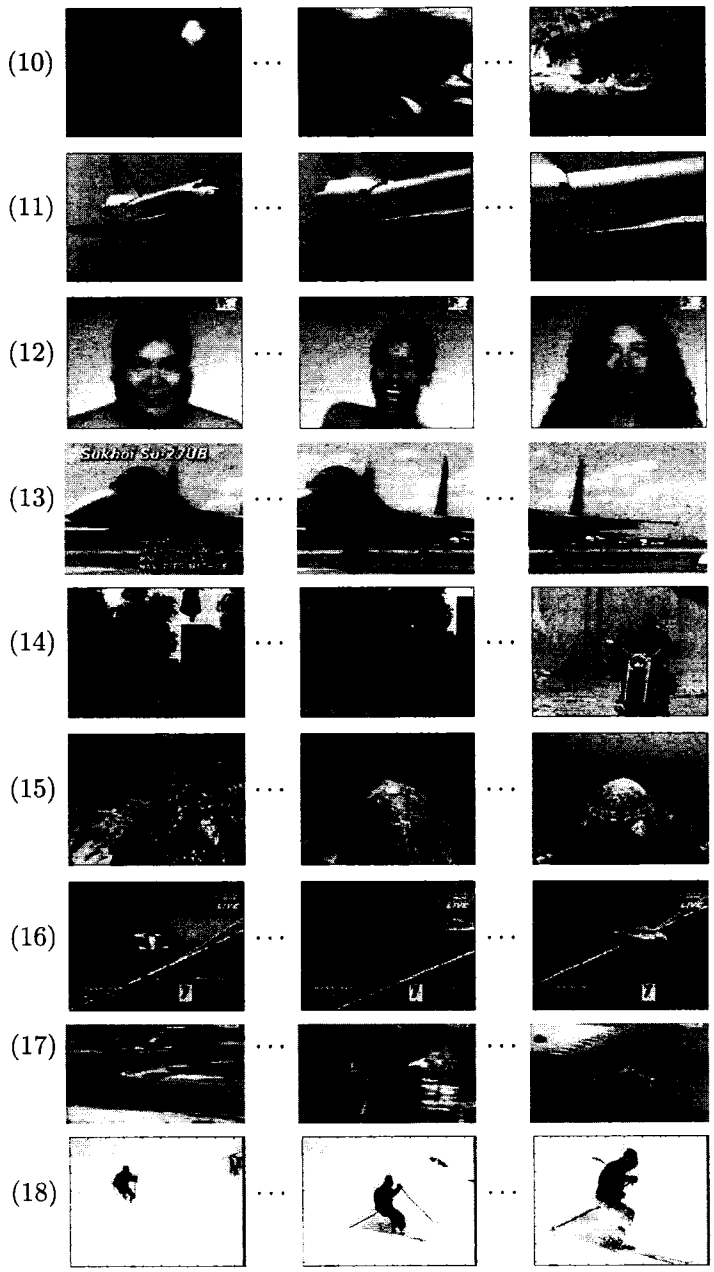


Figure 3: (Continued) This figure shows the first, middle, and last frames of the second set of nine video sequences used in this study. Each video sequence consists of 150 images and the corresponding image sizes are given in Table 1 in the row labelled "Index" (0).

4×4) to apply this technique. There is no such restriction on box filtering, therefore it is applied to reduce the images to all the sizes indicated in Table 1. Once the images are reduced to the lower resolutions, the corresponding low-resolution SVDs are computed. These low-resolution SVDs are then used to approximate the high-resolution SVD using the two techniques described earlier.

Table 1: Image sizes at different resolutions for the objects and the video sequences

	Obj.	Videos					
Index	All	(1) – (4)	(5) – (8)	(9) – (13)	(14), (15)	(16)	(17), (18)
(0)	128^2	352×240	320×240	160×120	304×228	240×180	320×180
(1)	64^2	176×120	160×120	80×60	152×114	120×90	160×90
(2)	32^2	88×60	80×60	40×30	76×57	60×45	80×45
(3)	16^2	44×30	40×30	20×15	38×28	30×22	40×22
(4)	8^2	22×15	20×15	10×8	19×14	15×11	20×11
(5)	4^2	12×8	10×8	5×4	10×7	8×6	10×6
(6)	2^2	6×4	5×4	3×2	5×4	4×3	5×3
(7)		3×2	3×2		3×2		

The results of the two approximation techniques using the first image data set, i.e., the rotated objects will now be evaluated. The first object (refer to Fig. 2) is used here as a representative example with its results using the two approximation techniques given in Figures 4 through 8. The results in Fig. 4 clearly indicate that, visually, low-resolution right singular vectors give a much better approximation of $\hat{\mathbf{u}}_1$ than low-resolution eigenimages. However, it is important to validate this result using the error measures defined in Section 2.4.

Fig. 5 shows the comparison of the U subspaces when the eigenimages at the lower resolutions are interpolated to the size of high-resolution eigenimages using bicubic interpolation (technique in Section 3.2.2). This figure shows that the approximation of high-resolution eigenimages deteriorates rapidly with an increase in the reduction factor. On the other hand, Fig. 6 shows the comparison of high-resolution and low-resolution V subspaces, which clearly indicates that low-resolution V subspaces give a good approximation of the high-resolution V subspace even at a very low resolution. Therefore, it is desirable to use the low-resolution V subspaces to approximate the high-resolution eigenimages. Fig. 7

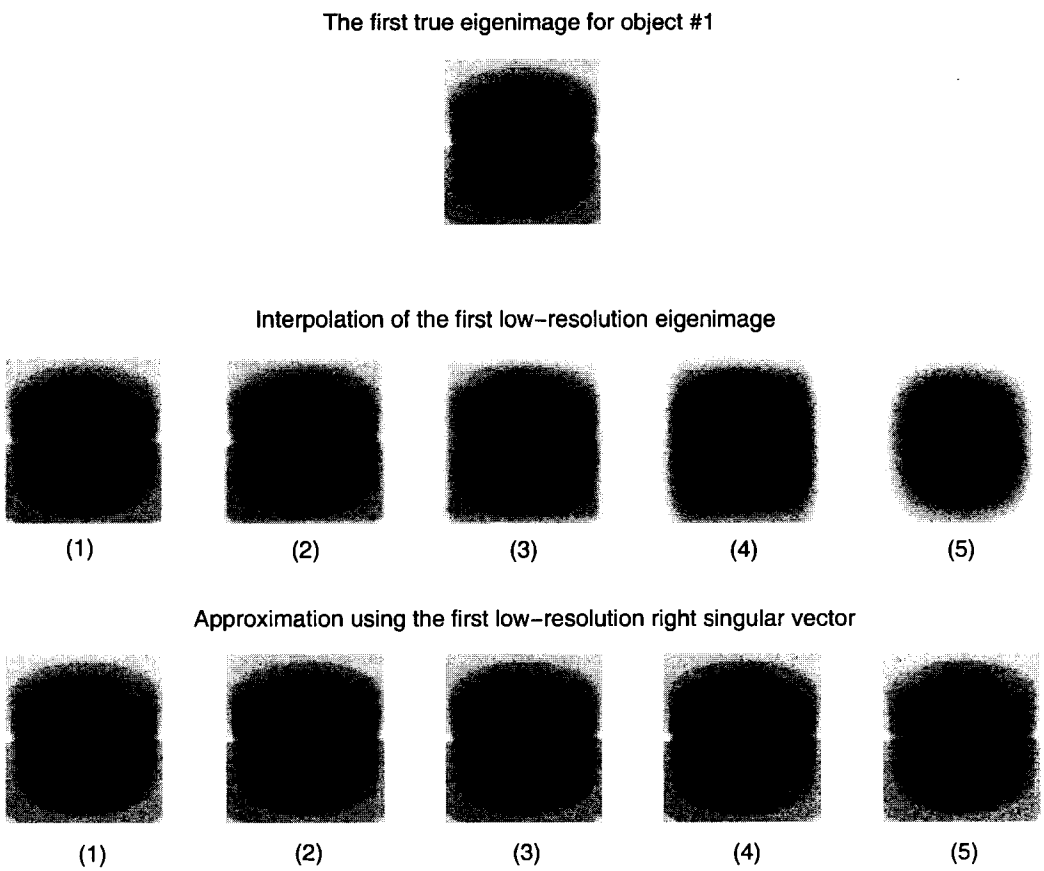


Figure 4: This figure shows the approximation of $\hat{\mathbf{u}}_1$ for object #1 in Fig. 2 using low-resolution SVD. The first row shows the true $\hat{\mathbf{u}}_1$. The second row shows the approximated $\hat{\mathbf{u}}_1$ ($\tilde{\hat{\mathbf{u}}}_1$) by interpolating the first low-resolution eigenimage at the lower resolutions, while the third row shows $\tilde{\hat{\mathbf{u}}}_1$ using the first low-resolution right singular vector at the lower resolutions. The numbers in the parentheses under the approximated eigenimages indicate the reduction indices, which indicate the size of the original low-resolution image data matrices.

shows the comparison of U subspaces when high-resolution eigenimages are approximated by using low-resolution V subspaces (technique in Section 3.2.3). The comparison of the results of the two techniques clearly reveals that using right singular vectors give a much better approximation of high-resolution eigenimages than the interpolation of low-resolution eigenimages.

Fig. 8 shows the same comparison plots as in Fig. 7, but for the cases where the images are reduced using box filtering instead of bicubic interpolation. One can observe that the results using box filtered images are better than bicubic interpolation. This is due to the fact that bicubic interpolation introduces more of a smoothing effect than box filtering, thus changing the spatial properties of X. Hence, it is advisable to reduce the images using box filtering, as it gives better results, is simple to implement, and is computationally more efficient.

Similar results are observed for the two approximation techniques, when the second image data set, i.e., the arbitrary video sequences, is used. The first video sequence (refer to Fig. 3) is used here as a representative example with its results using the two approximation techniques given in Figures 9 through 13. The results in Fig. 9 once again show that, visually, low-resolution right singular vectors give much better approximation of $\hat{\mathbf{u}}_1$ than low-resolution eigenimages.

Fig. 10 and Fig. 11 also indicate that low-resolution V subspaces give a good approximation of high-resolution V subspace even at a very low resolution, while the approximation of high-resolution eigenimages by interpolating low-resolution eigenimages deteriorates rapidly as the reduction factor increases. Therefore, low-resolution V subspaces are again used to approximate the high-resolution eigenimages and Fig. 12 again reveals that the right singular vectors give a much better approximation of the high-resolution eigenimages than interpolation of the low-resolution eigenimages. Likewise, Fig. 13 again shows that the results using box filtered images are better than bicubic interpolated images. Hence the images are reduced using box filtering hereafter.

The above empirical results clearly indicate that using the low-resolution right singular

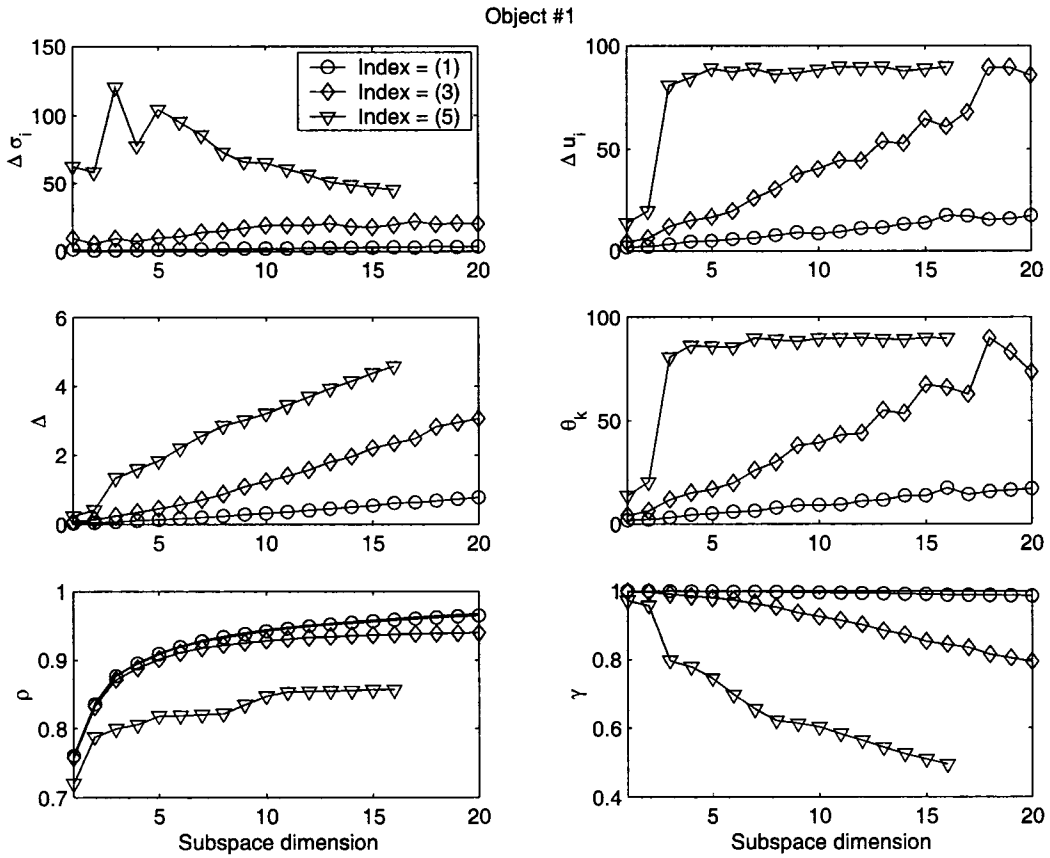


Figure 5: This figure shows the plots for comparison between true and approximated U subspaces for object #1 in Fig. 2, when high-resolution eigenimages are approximated by interpolating low-resolution eigenimages. (The images are reduced using bicubic interpolation and low-resolution eigenimages are also interpolated using bicubic interpolation.) The first 20 eigenimages are used for the comparison. The plots with $\circ$, $\Diamond$, and $\triangledown$ give the comparison between the true SVD and the approximated SVD using images with reduction index (1), (3), and (5), respectively. The plots for the reduction index (5) are cut short before 20, because there are only 16 eigenimages in the range of X at that resolution. The first row shows the difference between the true and the approximated singular values (left plot), and the angles in degrees between the individual true and approximated eigenimages. The second row shows the plots for the rotation indices (Δ) and the maximum principal angles (θ_k) in degrees when the subspace dimension k is varied from 1 to 20. The third row shows the plots for the energy recovery ratio (ρ) and subspace criterion (γ), when the subspace dimension k is varied from 1 to 20. (One of the plots for ρ's is plotted using a solid line that gives the "true" energy recovery ratio plot. This plot is invisible, as it is hidden under the plot with $\circ$.)

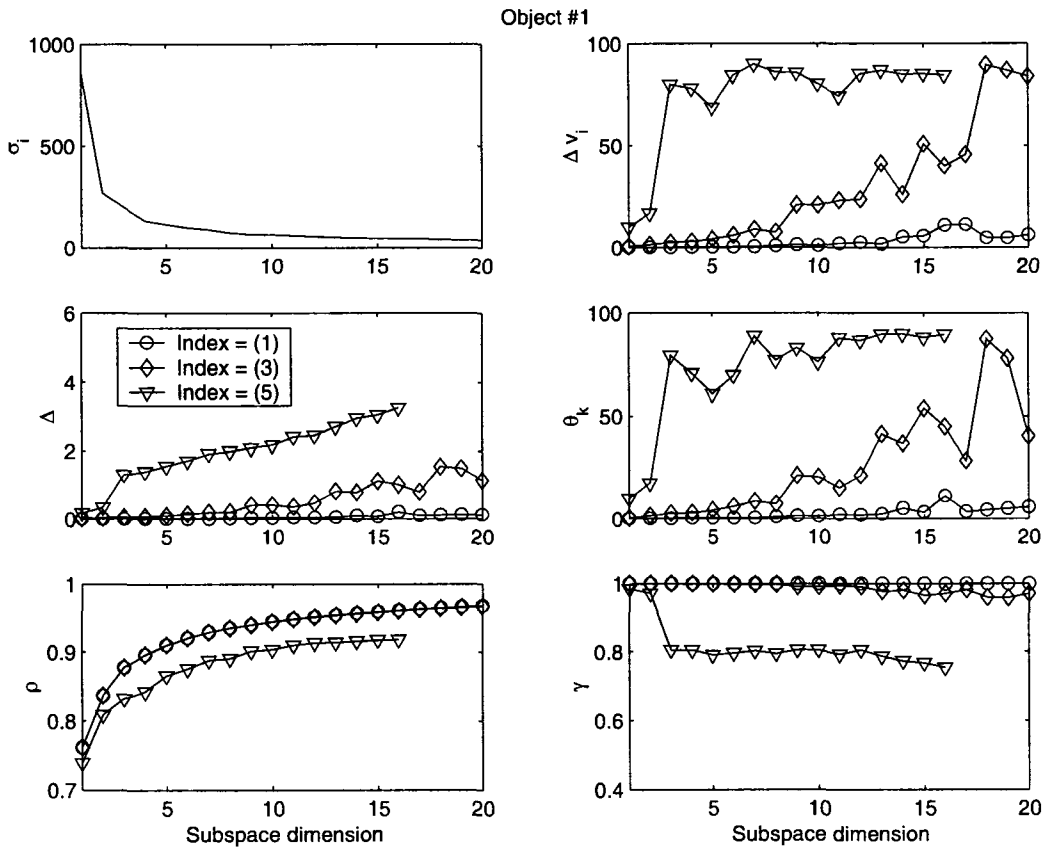


Figure 6: This figure shows the plots for comparison between high-resolution and low-resolution V subspaces for object #1 in Fig. 2. The first 20 right singular vectors are used for the comparison. The plots with $\circ$, $\Diamond$, and ∇ give the comparison between high-resolution V subspaces and low-resolution V subspaces obtained using images with reduction index (1), (3), and (5), respectively. The plots for the reduction index (5) are cut short before 20, because there are only 16 eigenimages in the range of X at that resolution. The first row shows high-resolution singular values (left plot), and the angles in degrees between the individual high-resolution and low-resolution right singular vectors. The second row shows the plots for the rotation indices (Δ) and the maximum principal angles (θ_k) in degrees when the subspace dimension k is varied from 1 to 20. The third row shows the plots for the energy recovery ratio (ρ) and subspace criterion (γ), when the subspace dimension k is varied from 1 to 20. (One of the plots for ρ's is plotted using a solid line that gives the "true" energy recovery ratio plot. This plot is invisible, as it is hidden under the plot with $\circ$.)

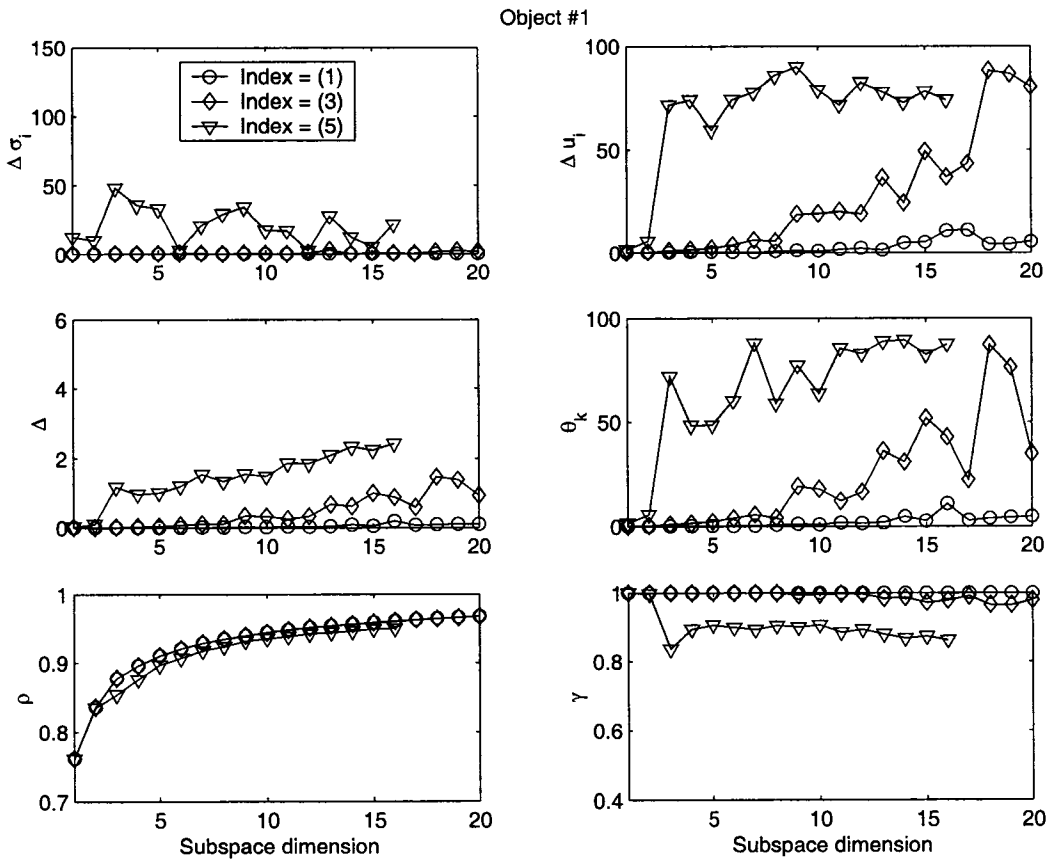


Figure 7: This figure shows the same measures as in Fig. 5 for comparison between true and approximated U subspaces for object #1 in Fig. 2, when high-resolution eigenimages are approximated by using low-resolution right singular vectors. The original images are reduced using bicubic interpolation.

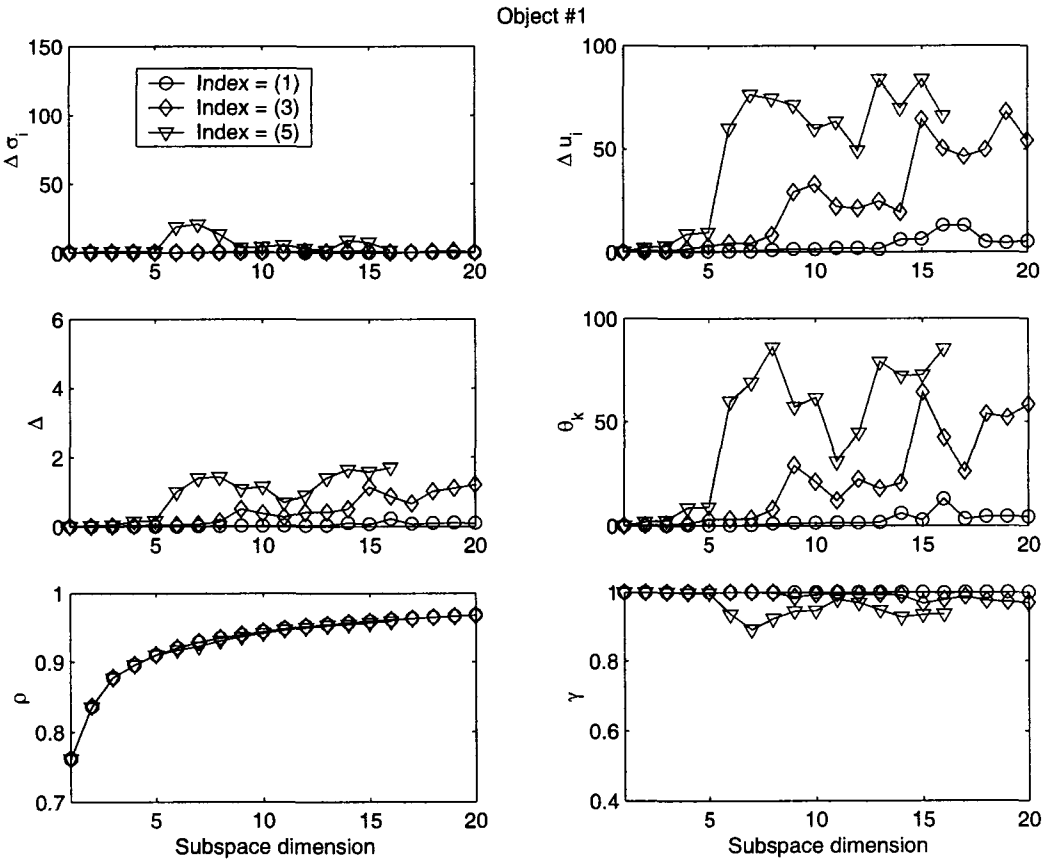


Figure 8: This figure shows the same measures as in Fig. 5 for comparison between true and approximated U subspaces for object #1 in Fig. 2, when high-resolution eigenimages are approximated by using low-resolution right singular vectors. The original images are reduced using simple box filtering.

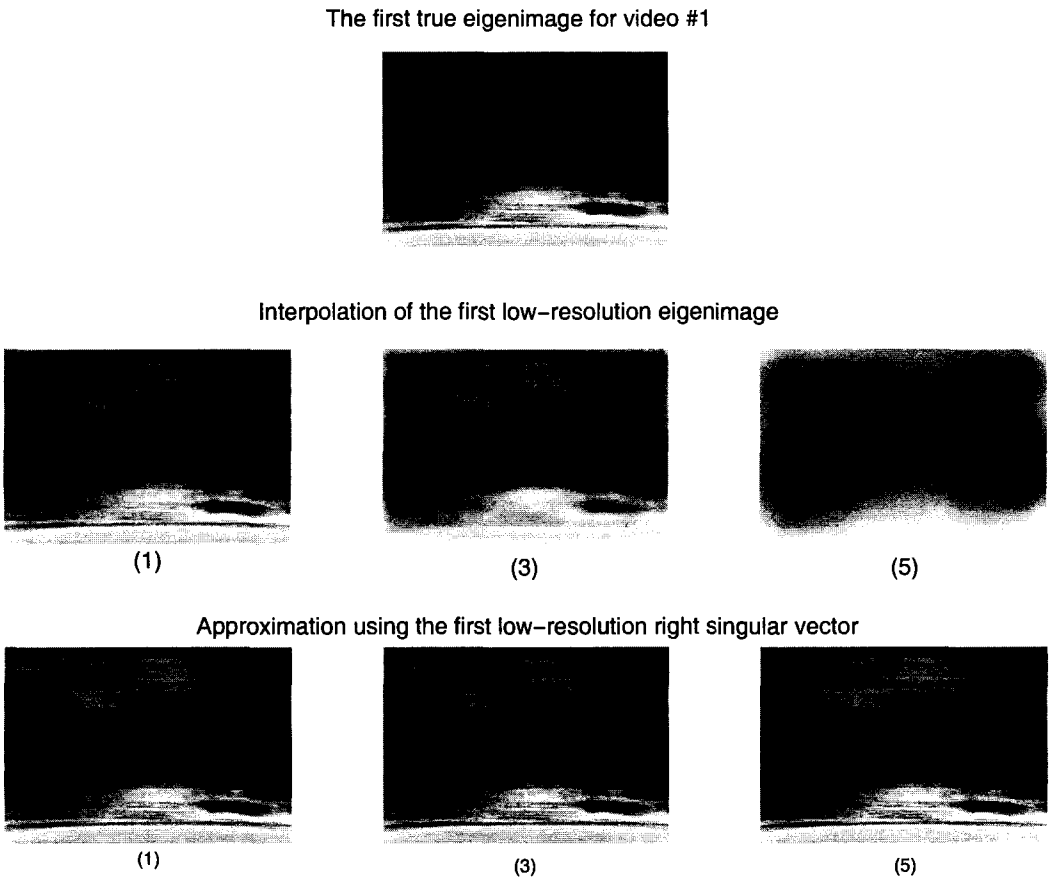


Figure 9: This figure shows the approximation of $\hat{\mathbf{u}}_1$ for video #1 in Fig. 3 using a low-resolution SVD. The first row shows the true $\hat{\mathbf{u}}_1$. The second row shows the approximated $\hat{\mathbf{u}}_1$ ($\tilde{\hat{\mathbf{u}}}_1$) by interpolating the first low-resolution eigenimage at the lower resolutions, while the third row shows $\tilde{\hat{\mathbf{u}}}_1$ using the first low-resolution right singular vector at the lower resolutions. The numbers in the parentheses under the approximated eigenimages indicate the reduction indices, which indicate the size of the original low-resolution image data matrices.

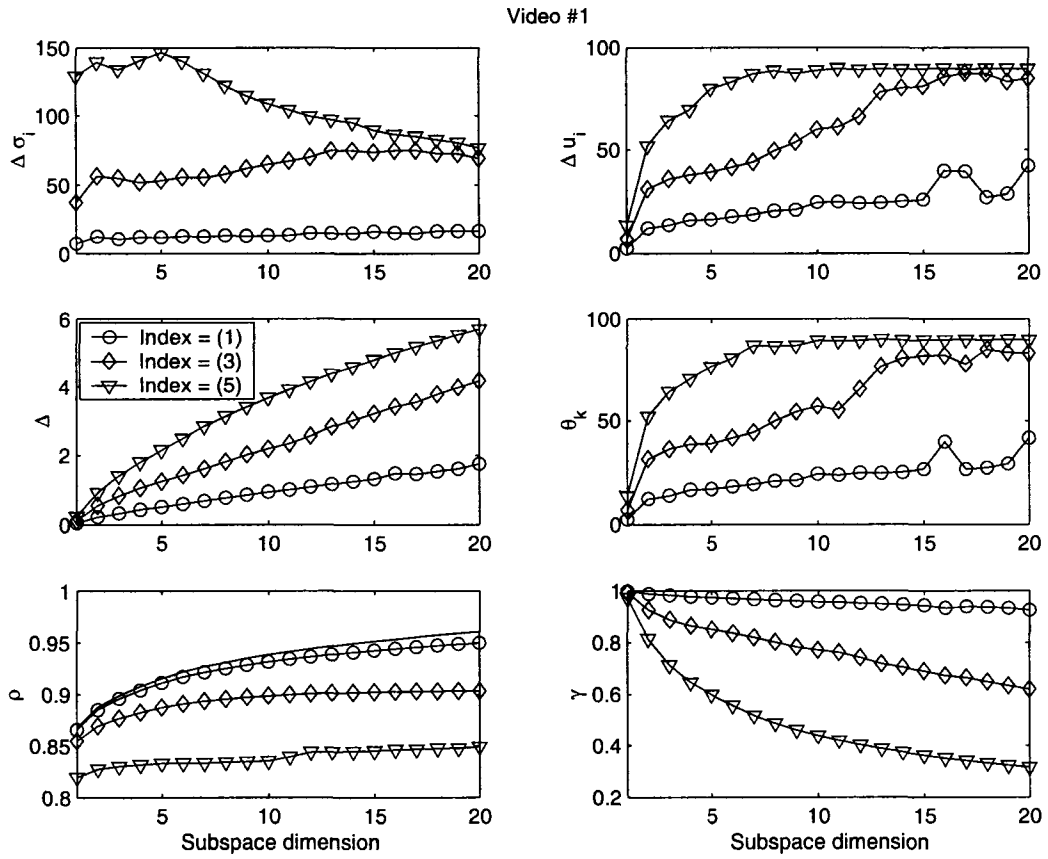


Figure 10: This figure shows the plots for comparison between true and approximated U subspaces for video #1 in Fig. 3, when high-resolution eigenimages are approximated by interpolating low-resolution eigenimages. (The images are reduced using bicubic interpolation and low-resolution eigenimages are also interpolated using bicubic interpolation.) The first 20 eigenimages are used for the comparison. The plots with $\circ$, $\Diamond$, and $\triangledown$ give the comparison between the true SVD and the approximated SVD using images with reduction index (1), (3), and (5), respectively. The first row shows the difference between the true and the approximated singular values (left plot), and the angles in degrees between the individual true and approximated eigenimages. The second row shows the plots for the rotation indices (Δ) and the maximum principal angles (θ_k) in degrees when the subspace dimension k is varied from 1 to 20. The third row shows the plots for the energy recovery ratio (ρ) and subspace criterion (γ), when the subspace dimension k is varied from 1 to 20. Solid line plot in the different ρ plots gives the "true" energy recovery ratio plot.

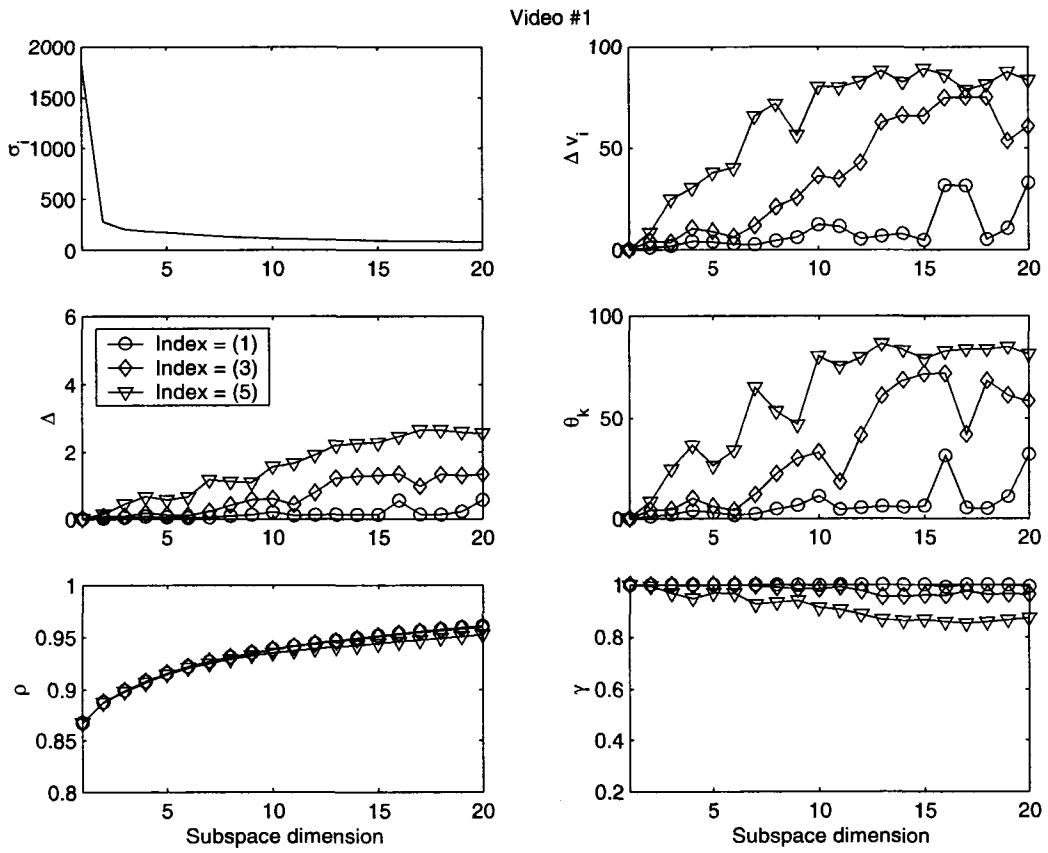


Figure 11: This figure shows the plots for comparison between high-resolution and low-resolution V subspaces for video #1 in Fig. 3. The first 20 right singular vectors are used for the comparison. The plots with $\circ$, $\Diamond$, and $\triangledown$ give the comparison between high-resolution V subspaces and low-resolution V subspaces obtained using images with reduction index (1), (3), and (5), respectively. The first row shows high-resolution singular values (left plot), and the angles in degrees between the individual high-resolution and low-resolution right singular vectors. The second row shows the plots for the rotation indices (Δ) and the maximum principal angles (θ_k) in degrees when the subspace dimension k is varied from 1 to 20. The third row shows the plots for the energy recovery ratio (ρ) and subspace criterion (γ), when the subspace dimension k is varied from 1 to 20. (One of the plots for ρ's is plotted using a solid line that gives the "true" energy recovery ratio plot. This plot is invisible, as it is hidden under the plot with $\circ$.)

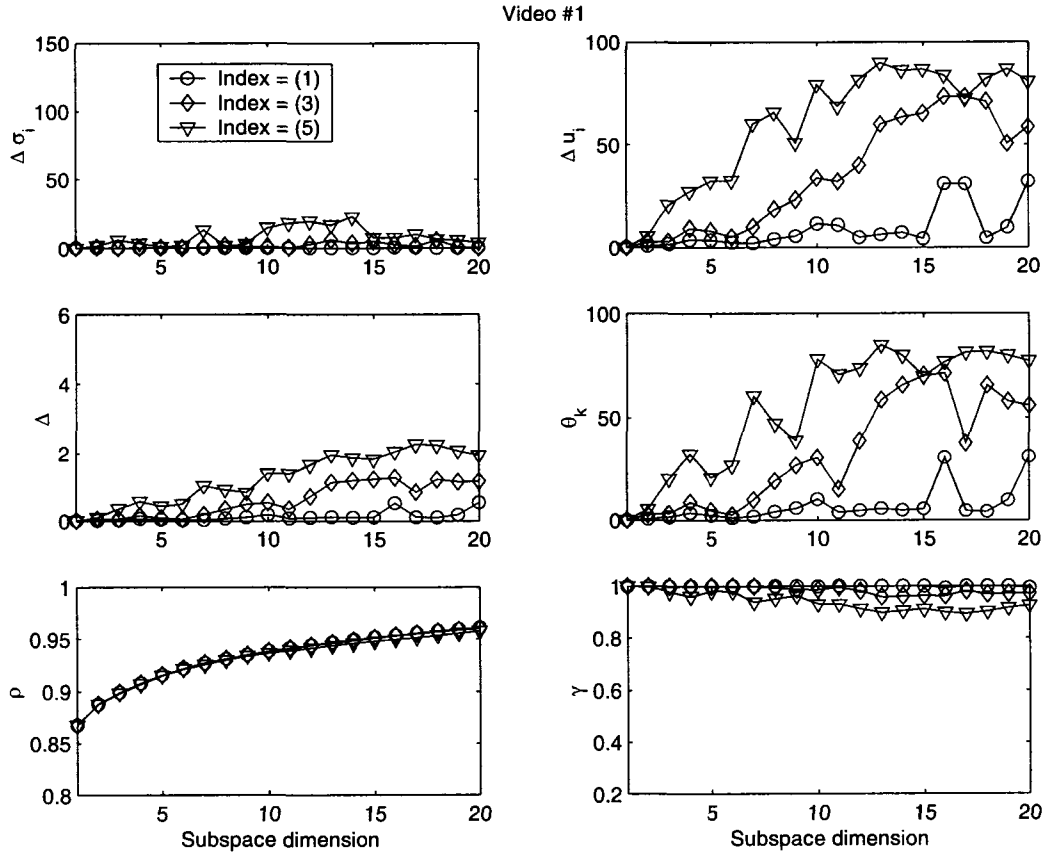


Figure 12: This figure shows the same measures as in Fig. 10 for comparison between true and approximated U subspaces for video #1 in Fig. 3, when high-resolution eigenimages are approximated by using low-resolution right singular vectors. The original images are reduced using bicubic interpolation.

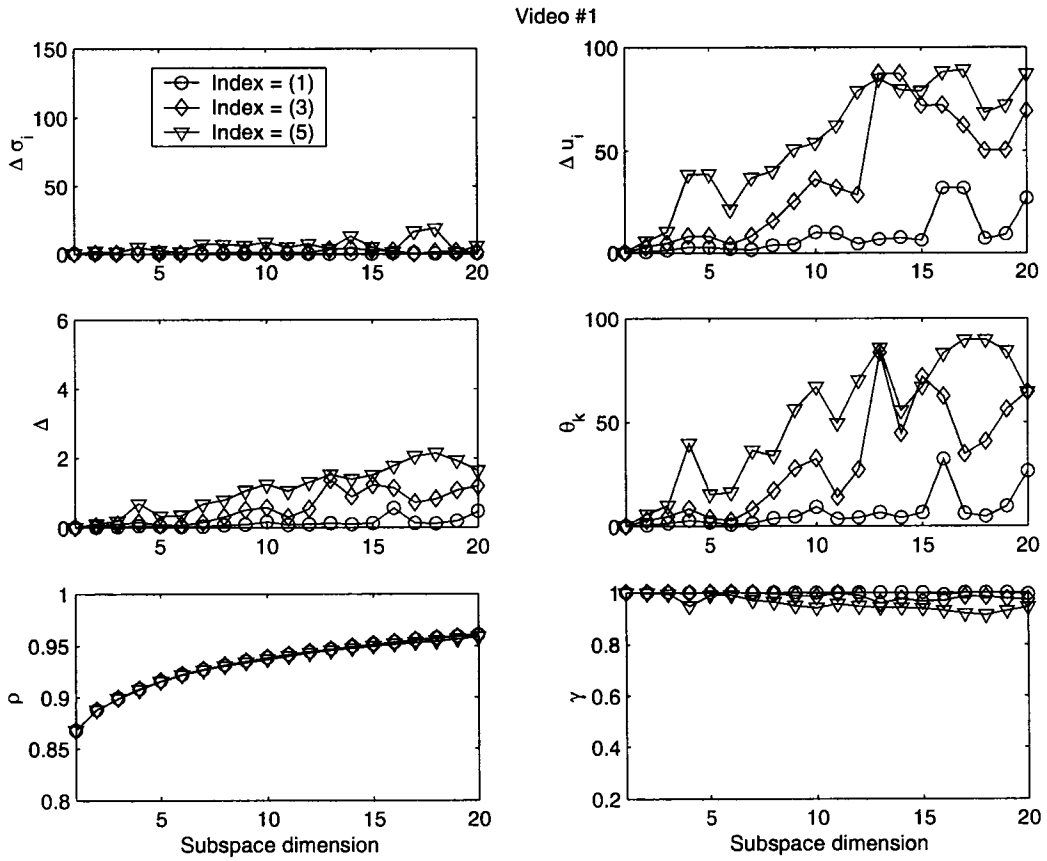


Figure 13: This figure shows the same measures as in Fig. 10 for comparison between true and approximated U subspaces for video #1 in Fig. 3, when high-resolution eigenimages are approximated by using low-resolution right singular vectors. The original images are reduced using simple box filtering.

vectors to compute high-resolution eigenimages is more accurate than interpolating low-resolution eigenimages. Both techniques were also evaluated for the common application of image reconstruction. The energy recovery ratio (ρ) was used to quantify the errors associated with the image reconstruction, when different approximated eigenimages are used. In particular, the energy recovery ratio using true high-resolution eigenimages (denoted ρ_t) was calculated for the given image data matrix X. The images in X were then reduced to some lower resolution using box filtering, the low-resolution image data matrix was formed, and the corresponding SVD was computed. Then the low-resolution SVD was used to approximate the high-resolution eigenimages and these approximated eigenimages were used to compute the energy recovery ratio (denoted ρ_a) for X. The relative ρ is given by $\frac{\rho_a}{\rho_t}$. Note that the true eigenimages give the optimum eigenspace for X, hence the relative ρ will always be less than 1.

Tables 2 and 3 show the relative image reconstruction results of the two techniques in terms of ρ for the rotated objects, while Tables 4 and 5 show the relative image reconstruction results of the two techniques in terms of ρ for the video sequences. These results again show that low-resolution right singular vectors give a much better approximation of high-resolution eigenimages than the interpolated eigenimages.

Recall that the individual images can be reconstructed using

$$\tilde{\mathbf{x}}_t(k) = \sum_{j=1}^{k} (\mathbf{x}_t^T \boldsymbol{\phi}_j)\boldsymbol{\phi}_j \tag{40}$$

where the $\boldsymbol{\phi}_j$s form an orthonormal set. Thus, these $\boldsymbol{\phi}_j$s can be replaced by true or approximated eigenimages and the corresponding image reconstruction can be compared. Fig. 14 and Fig. 15 show the reconstruction of a single image using the approximated eigenimages, which support the previous results obtained using the techniques in Section 3.2.2 and Section 3.2.3. These empirical results are explored in the next section using a special case of X that has a closed form SVD at both high and low resolution.

3.3 A Special Case with a Closed Form SVD

This section considers an image data matrix that has a closed form solution for the SVD at both high and low resolutions. This closed form solution along with its properties can

Table 2: Relative ρ for approximation by interpolating low-resolution eigenimages

	Reduction index					
Object	(1)	(2)	(3)	(4)	(5)	(6)
1	0.9955	0.9836	0.9565	0.9213	0.8877	0.7217
2	0.9926	0.9759	0.9344	0.8624	0.7094	0.3151
3	0.9905	0.9666	0.9125	0.8417	0.8013	0.5702
4	0.9953	0.9817	0.9517	0.9009	0.8298	0.5024
5	0.9909	0.9746	0.9379	0.8946	0.7555	0.4130
6	0.9943	0.9780	0.9238	0.8031	0.6929	0.4542
7	0.9813	0.9602	0.9251	0.9088	0.8689	0.5370
8	0.9893	0.9722	0.9384	0.8618	0.7134	0.3591
9	0.9853	0.9642	0.9176	0.8355	0.7223	0.4246
10	0.9957	0.9876	0.9708	0.9469	0.9026	0.7293
11	0.9957	0.9848	0.9551	0.9009	0.7840	0.3360
12	0.9968	0.9863	0.9598	0.9129	0.8641	0.4624
13	0.9954	0.9807	0.9428	0.8852	0.7818	0.5394
14	0.9935	0.9774	0.9413	0.8848	0.8075	0.4849
15	0.9951	0.9839	0.9621	0.9323	0.8968	0.5358
16	0.9950	0.9768	0.9300	0.8521	0.7814	0.6669
17	0.9946	0.9797	0.9444	0.8929	0.8574	0.6391
18	0.9951	0.9778	0.9366	0.8601	0.7756	0.5090
19	0.9959	0.9840	0.9506	0.8943	0.8421	0.4812
20	0.9915	0.9689	0.9075	0.8397	0.7664	0.4281
Mean	0.9930	0.9773	0.9400	0.8816	0.8020	0.5055
Maximum	0.9968	0.9876	0.9708	0.9469	0.9026	0.7293
Minimum	0.9813	0.9602	0.9075	0.8031	0.6929	0.3151

Table 3: Relative ρ for approximation using low-resolution right singular vectors

	Reduction index					
Object	(1)	(2)	(3)	(4)	(5)	(6)
1	1.0000	0.9999	0.9995	0.9994	0.9975	0.9804
2	1.0000	0.9998	0.9990	0.9983	0.9955	0.9843
3	1.0000	0.9999	0.9996	0.9976	0.9956	0.9936
4	1.0000	0.9999	0.9996	0.9984	0.9974	0.9737
5	0.9999	0.9997	0.9993	0.9991	0.9975	0.9996
6	1.0000	1.0000	0.9990	0.9968	0.9878	0.9688
7	1.0000	0.9999	0.9999	0.9995	0.9975	0.9944
8	0.9999	0.9995	0.9985	0.9977	0.9844	0.9878
9	0.9999	0.9998	0.9995	0.9980	0.9942	0.9855
10	1.0000	0.9998	0.9994	0.9985	0.9977	0.9984
11	1.0000	0.9999	0.9995	0.9973	0.9947	0.9894
12	1.0000	1.0000	0.9999	0.9994	0.9985	0.9987
13	1.0000	1.0000	0.9998	0.9997	0.9968	0.9909
14	1.0000	1.0000	0.9998	0.9995	0.9974	0.9892
15	1.0000	1.0000	0.9999	0.9996	0.9987	0.9947
16	1.0000	1.0000	0.9995	0.9982	0.9927	0.9932
17	1.0000	1.0000	0.9998	0.9995	0.9965	0.9931
18	1.0000	1.0000	0.9998	0.9992	0.9955	0.9736
19	1.0000	0.9999	0.9996	0.9993	0.9961	0.9855
20	1.0000	0.9999	0.9997	0.9994	0.9968	0.9891
Mean	1.0000	0.9999	0.9995	0.9987	0.9954	0.9882
Maximum	1.0000	1.0000	0.9999	0.9997	0.9987	0.9996
Minimum	0.9999	0.9995	0.9985	0.9968	0.9844	0.9688

Table 4: Relative ρ for approximation by interpolating low-resolution eigenimages

	Reduction index						
Video	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	0.9866	0.9665	0.9350	0.9080	0.8857	0.8692	0.8457
2	0.9794	0.9680	0.9597	0.9515	0.9425	0.9304	0.9298
3	0.9921	0.9791	0.9588	0.9299	0.9005	0.8627	0.8932
4	0.9845	0.9609	0.9186	0.8737	0.8321	0.7996	0.8141
5	0.9986	0.9938	0.9835	0.9651	0.9358	0.8831	0.8077
6	0.9978	0.9943	0.9908	0.9857	0.9800	0.9744	0.9810
7	0.9966	0.9877	0.9717	0.9478	0.9208	0.8895	0.8857
8	0.9987	0.9947	0.9887	0.9803	0.9684	0.9475	0.9408
9	0.9894	0.9678	0.9181	0.8470	0.7663	0.6482	
10	0.9976	0.9929	0.9845	0.9697	0.9416	0.9219	
11	0.9919	0.9773	0.9570	0.9331	0.8921	0.8749	
12	0.9994	0.9978	0.9936	0.9830	0.9545	0.9383	
13	0.9916	0.9797	0.9637	0.9440	0.9274	0.8842	
14	0.9974	0.9926	0.9846	0.9698	0.9485	0.9237	0.9049
15	0.9969	0.9900	0.9765	0.9551	0.9258	0.9027	0.8850
16	0.9823	0.9631	0.9368	0.9117	0.8843	0.8703	
17	0.9983	0.9961	0.9916	0.9840	0.9751	0.9611	
18	0.9967	0.9923	0.9870	0.9816	0.9725	0.9664	
Mean	0.9931	0.9830	0.9667	0.9456	0.9197	0.8916	0.8888
Maximum	0.9994	0.9978	0.9936	0.9857	0.9800	0.9744	0.9810
Minimum	0.9794	0.9609	0.9181	0.8470	0.7663	0.6482	0.8077

Table 5: Relative ρ for approximation using low-resolution right singular vectors

	Reduction index						
Video	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	1.0000	0.9999	0.9997	0.9987	0.9978	0.9956	0.9958
2	0.9999	0.9997	0.9993	0.9989	0.9978	0.9974	0.9989
3	0.9999	0.9995	0.9990	0.9980	0.9968	0.9909	0.9863
4	1.0000	0.9998	0.9993	0.9983	0.9976	0.9957	0.9847
5	1.0000	1.0000	0.9999	0.9998	0.9993	0.9973	0.9990
6	1.0000	1.0000	1.0000	0.9999	0.9993	0.9987	0.9985
7	1.0000	0.9999	0.9998	0.9997	0.9989	0.9959	0.9932
8	1.0000	1.0000	1.0000	0.9999	0.9994	0.9979	0.9959
9	1.0000	0.9997	0.9975	0.9953	0.9867	0.9907	
10	1.0000	0.9999	0.9997	0.9987	0.9961	0.9945	
11	1.0000	0.9999	0.9998	0.9992	0.9982	0.9956	
12	1.0000	1.0000	1.0000	0.9993	0.9970	0.9956	
13	1.0000	1.0000	0.9998	0.9996	0.9991	0.9985	
14	1.0000	0.9999	0.9996	0.9993	0.9990	0.9978	0.9972
15	1.0000	1.0000	0.9999	0.9997	0.9992	0.9974	0.9929
16	1.0000	0.9996	0.9992	0.9991	0.9986	0.9975	
17	1.0000	0.9999	0.9998	0.9993	0.9979	0.9969	
18	1.0000	1.0000	1.0000	0.9997	0.9988	0.9980	
Mean	1.0000	0.9999	0.9996	0.9990	0.9976	0.9962	0.9942
Maximum	1.0000	1.0000	1.0000	0.9999	0.9994	0.9987	0.9990
Minimum	0.9999	0.9995	0.9975	0.9953	0.9867	0.9907	0.9847

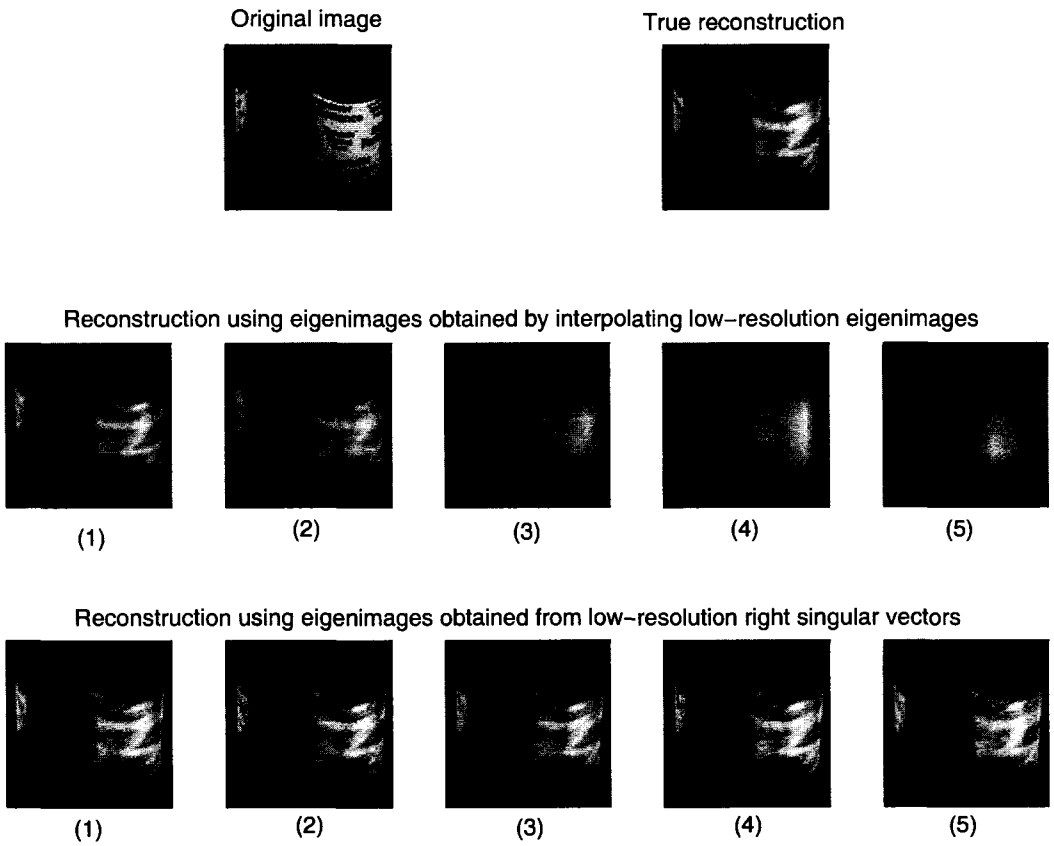


Figure 14: This figure shows the reconstruction of the first image for object #1 in Fig. 2. The first row shows the original image and its reconstruction using the first 20 true eigenimages. The second row shows the reconstruction using interpolated eigenimages, while the third row shows the reconstruction using eigenimages obtained from low-resolution right singular vectors. The numbers below the reconstructed images give the reduction index.

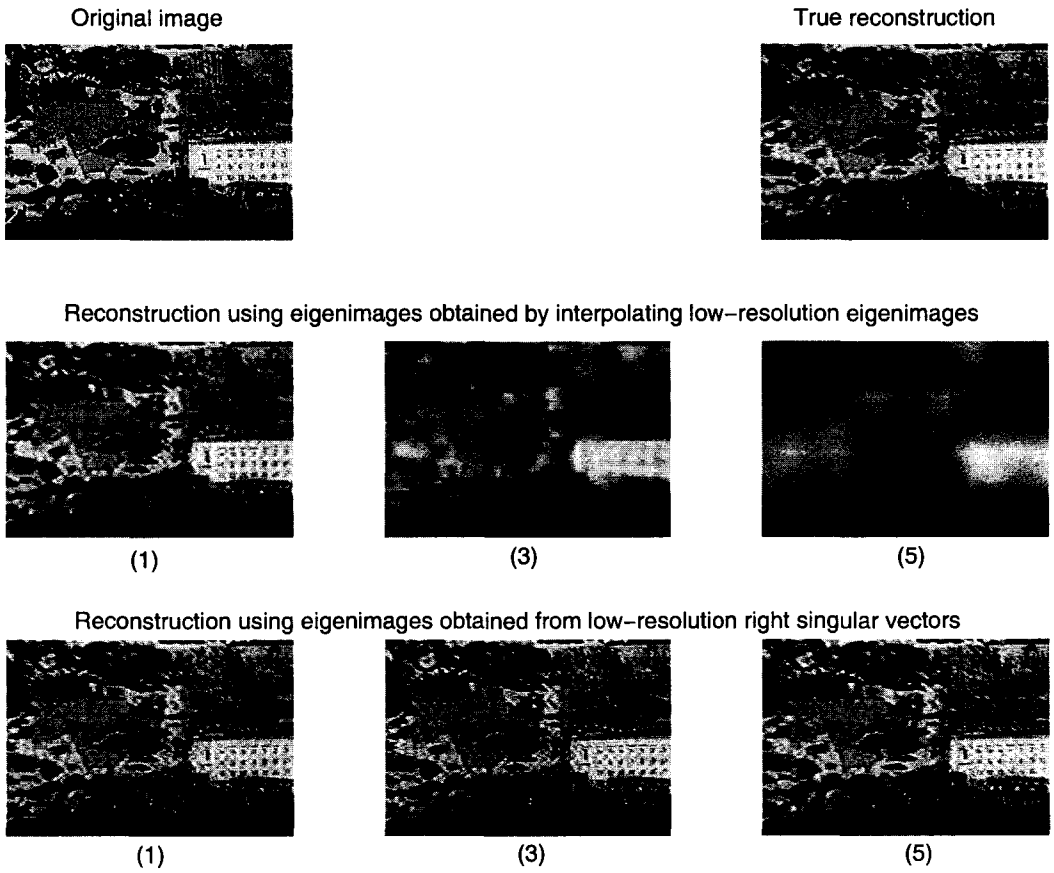


Figure 15: This figure shows the reconstruction of the first image for video sequence #1 in Fig. 3. The first row shows the original image and its reconstruction using the first 20 true eigenimages. The second row shows the reconstruction using interpolated eigenimages, while the third row shows the reconstruction using eigenimages obtained from low-resolution right singular vectors. The numbers below the reconstructed images give the reduction index.

then be used to support the empirical results in Section 3.2.4.

Consider two images with m pixels that have been row-scanned and normalized to unit norm. The $m \times 2$ high-resolution image data matrix, X_h, is given by

$$X_h = \begin{bmatrix} \hat{\mathbf{x}}_{h_1} & \hat{\mathbf{x}}_{h_2} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{m1} & x_{m2} \end{bmatrix} \tag{41}$$

where the $\hat{}$ notation indicates that the corresponding vectors are normalized to unit norm. The pixels in $\hat{\mathbf{x}}_{h_1}$ and $\hat{\mathbf{x}}_{h_2}$ are ordered so that a pixel in the low-resolution image vectors, $\mathbf{x}_{l_1}$ and $\mathbf{x}_{l_2}$, can be obtained by box-filtering the consecutive pixels in $\hat{\mathbf{x}}_{h_1}$ and $\hat{\mathbf{x}}_{h_2}$, respectively. Thus, with the integer reduction factor r, the low-resolution image data matrix X_l is given by

$$X_l = \frac{1}{r} \begin{bmatrix} x_{11} + \cdots + x_{r1} & x_{12} + \cdots + x_{r2} \\ \vdots & \vdots \\ x_{d1} + \cdots + x_{m1} & x_{d2} + \cdots + x_{m2} \end{bmatrix} \tag{42}$$

where $d = m - r + 1$. The critical step in calculating the SVD of X is to determine an orthogonal matrix V that will orthogonalize the columns of X. This matrix can be formed as a Givens rotation that is designed to orthogonalize two columns and results in the following V and U matrices for X_h:

$$\begin{aligned} V_h &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -\text{sgn}(\hat{\mathbf{x}}_{h_1}^T \hat{\mathbf{x}}_{h_2}) \\ \text{sgn}(\hat{\mathbf{x}}_{h_1}^T \hat{\mathbf{x}}_{h_2}) & 1 \end{bmatrix} \\ U_h &= \begin{bmatrix} \frac{\hat{\mathbf{x}}_{h_1} \pm \hat{\mathbf{x}}_{h_2}}{\|\hat{\mathbf{x}}_{h_1} \pm \hat{\mathbf{x}}_{h_2}\|} & \frac{\mp \hat{\mathbf{x}}_{h_1} + \hat{\mathbf{x}}_{h_2}}{\|\mp \hat{\mathbf{x}}_{h_1} + \hat{\mathbf{x}}_{h_2}\|} \end{bmatrix} \end{aligned} \tag{43}$$

where the subscript h denotes that these matrices correspond to X_h and sgn denotes a signum function. The upper sign in the $\pm$ and $\mp$ notations in the U_h matrix occurs when the dot product between the high-resolution image vectors is positive. Note that such closed form solutions can be obtained for these matrices because the image vectors in X_h have equal norms. The column vectors in X_l, however, do not necessarily have equal norms,

hence U and V matrices for X_l do not have the same closed form solution. To find these matrices, one can make use of the formulas for the SVD algorithm that relies on Givens rotations [54]. These formulas are based on the quantities,

$$y_l = \mathbf{x}_{l_1}^T \mathbf{x}_{l_2} \tag{44}$$

$$z_l = \mathbf{x}_{l_1}^T \mathbf{x}_{l_1} - \mathbf{x}_{l_2}^T \mathbf{x}_{l_2} \tag{45}$$

$$w_l = \sqrt{4y_l^2 + z_l^2} \tag{46}$$

so that

$$\begin{aligned} \cos\theta &= \sqrt{\frac{w_l + z_l}{2w_l}} \\ \sin\theta &= \frac{y_l}{w_l \cos\theta} \end{aligned} \tag{47}$$

if $z_l \geq 0$ and

$$\begin{aligned} \sin\theta &= \mathrm{sgn}(y_l)\sqrt{\frac{w_l - z_l}{2w_l}} \\ \cos\theta &= \frac{y_l}{w_l \sin\theta} \end{aligned} \tag{48}$$

if $z_l < 0$. Then the V and U matrices for X_l can be given by

$$\begin{aligned} V_l &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \\ U_l &= \begin{bmatrix} \frac{\cos\theta \mathbf{x}_{l_1} + \sin\theta \mathbf{x}_{l_2}}{\|\cos\theta \mathbf{x}_{l_1} + \sin\theta \mathbf{x}_{l_2}\|} & \frac{-\sin\theta \mathbf{x}_{l_1} + \cos\theta \mathbf{x}_{l_2}}{\|-\sin\theta \mathbf{x}_{l_1} + \cos\theta \mathbf{x}_{l_2}\|} \end{bmatrix} \end{aligned} \tag{49}$$

where the subscript l denotes that these matrices correspond to X_l. The left and right singular vectors at high and low resolutions can be compared against each other by using the difference measures given in Section 2.4. However, the dot product of the difference between the corresponding vectors indicate that the bound of this error is in the range of [0, 2] for both the right singular vectors and the interpolated eigenimages. Hence it is necessary to study the approximations of the high-resolution eigenimages using the low-resolution SVD in more detail, which is the topic of the following subsections.

3.3.1 Limitations of Interpolated Low-Resolution Eigenimages

It is explained in Section 3.2.2 how the interpolated low-resolution eigenimages can be used as an approximation of their high-resolution counterparts. The results were acceptable as

long as the reduction in resolution is not too great. It is shown here why this is true using a simple example.

Consider X_h with $m = 4$. The first (unnormalized) eigenimage of X_h is given by

$$\mathbf{u}_{h_1} = \frac{1}{2}\begin{bmatrix} x_{11} + x_{12} \\ x_{21} + x_{22} \\ x_{31} + x_{32} \\ x_{41} + x_{42} \end{bmatrix} \tag{50}$$

where the dot product between the columns of X_h is assumed to be positive. For $r = 2$, the matrix V_l for the corresponding X_l can be generated using the quantities in (44), (45), and (46). Then the first (unnormalized) eigenimage of X_l is given by

$$\begin{aligned} \mathbf{u}_{l_1} &= X_l \mathbf{v}_{l_1} \\ &= \frac{1}{2}\left(\cos\theta \begin{bmatrix} x_{11} + x_{21} \\ x_{31} + x_{41} \end{bmatrix} + \sin\theta \begin{bmatrix} x_{12} + x_{22} \\ x_{32} + x_{42} \end{bmatrix}\right). \end{aligned}$$

The linear interpolation[2] of $\mathbf{u}_{l_1}$ to the size of $\mathbf{u}_{h_1}$ gives

$$\tilde{\mathbf{u}}_{h_1} = L\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 5 & -1 \\ 3 & 1 \\ 1 & 3 \\ -1 & 5 \end{bmatrix}\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \tag{51}$$

where L gives the linear interpolation model, $r_1 = 0.5(\cos\theta(x_{11} + x_{21}) + \sin\theta(x_{12} + x_{22}))$, and $r_2 = 0.5(\cos\theta(x_{31} + x_{41}) + \sin\theta(x_{32} + x_{42}))$. Note that the columns of the following matrix form an orthonormal basis for the space perpendicular to the column space of L:

$$N_L = \begin{bmatrix} 0.5 & 0.2236 \\ -0.5 & -0.6708 \\ -0.5 & 0.6708 \\ 0.5 & -0.2236 \end{bmatrix}. \tag{52}$$

Now consider the family of all 4×2 matrix X_h with the following properties:

[2]Bicubic interpolation is used in Section 3.2.4 for better accuracy, while linear interpolation is used here for mathematical simplicity.

1. The column space of X_h is perpendicular to the column space of L.

2. The columns of X_h have unit norm.

3. The angle between the columns of X_h is α.

This family, denoted by $X_h(\phi)$, can be parameterized by ϕ in the following way:

$$X_h(\phi) = N_L \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & \cos\alpha \\ 0 & \pm\sin\alpha \end{bmatrix} \tag{53}$$

where ϕ is any angle. For a particular case of X_h, when $\phi = 60^\circ$ and $\alpha = 10^\circ$, we can see that,

$$\hat{\mathbf{u}}_{h_i}^T \tilde{\hat{\mathbf{u}}}_{h_j} = 0, \quad \text{for } i, j = 1, 2. \tag{54}$$

Thus the column space of the approximated eigenimages is orthogonal to the column space of the true eigenimages, giving the worst possible approximation even for simple X_h. Similar behavior was observed in Section 3.2.4 where the interpolated eigenimages gave a very poor approximation of the true eigenimages at high resolution, when the image data matrices consist of images at very low resolution.

3.3.2 Advantages of Using Low-Resolution Right Singular Vectors

It was shown in Section 3.2.3 that the low-resolution right singular vectors can be multiplied with the high-resolution image data matrix to get an approximate basis for the high-resolution eigenimages. Thus, in the current example, V_l can be used to approximate U_h (denoted $\tilde{U}_h$) and Σ_h (denoted $\tilde{\Sigma}_h$), i.e.,

$$\begin{aligned} \tilde{U}_h\tilde{\Sigma}_h &= X_h V_l \\ &= \begin{bmatrix} \hat{\mathbf{x}}_{h_1} & \hat{\mathbf{x}}_{h_2} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mathbf{x}}_{h_1}\cos\theta + \hat{\mathbf{x}}_{h_2}\sin\theta & -\hat{\mathbf{x}}_{h_1}\sin\theta + \hat{\mathbf{x}}_{h_2}\cos\theta \end{bmatrix}. \end{aligned} \tag{55}$$

If the first and the second column of $\tilde{U}_h\tilde{\Sigma}_h$ are denoted $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively, Gram-Schmidt orthogonalization gives

$$\begin{aligned}\tilde{\hat{\mathbf{u}}}_{h_1} &= \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|} = \frac{\cos\theta\hat{\mathbf{x}}_{h_1} + \sin\theta\hat{\mathbf{x}}_{h_2}}{\sqrt{1+2y_h\sin\theta\cos\theta}} \\ \tilde{\hat{\mathbf{u}}}_{h_2} &= \frac{\mathbf{a}_2 - (\tilde{\hat{\mathbf{u}}}_{h_1}^T\mathbf{a}_2)\tilde{\hat{\mathbf{u}}}_{h_1}}{\|\mathbf{a}_2 - (\tilde{\hat{\mathbf{u}}}_{h_1}^T\mathbf{a}_2)\tilde{\hat{\mathbf{u}}}_{h_1}\|} = \frac{c_1\hat{\mathbf{x}}_{h_2} - c_2\hat{\mathbf{x}}_{h_1}}{\sqrt{c_1^2 - 2y_hc_1c_2 + c_2^2}}\end{aligned} \tag{56}$$

where $y_h = \hat{\mathbf{x}}_{h_1}^T\hat{\mathbf{x}}_{h_2}$, $c_1 = \cos\theta + y_h\sin\theta$, and $c_2 = \sin\theta + y_h\cos\theta$. The Gram-Schmidt orthogonalization is started with column $\mathbf{a}_1$, because it is likely to be of larger norm than $\mathbf{a}_2$.

To check as to how much these approximated eigenimages differ from the correct ones, consider the difference

$$\begin{aligned}\Delta\mathbf{u}_{h_1} &= \hat{\mathbf{u}}_{h_1} - \tilde{\hat{\mathbf{u}}}_{h_1} \\ &= \frac{\hat{\mathbf{x}}_{h_1} \pm \hat{\mathbf{x}}_{h_2}}{\|\hat{\mathbf{x}}_{h_1} \pm \hat{\mathbf{x}}_{h_2}\|} - \frac{\cos\theta\hat{\mathbf{x}}_{h_1} + \sin\theta\hat{\mathbf{x}}_{h_2}}{\|\cos\theta\hat{\mathbf{x}}_{h_1} + \sin\theta\hat{\mathbf{x}}_{h_2}\|} \\ &= \frac{\hat{\mathbf{x}}_{h_1} \pm \hat{\mathbf{x}}_{h_2}}{\sqrt{2(1\pm y_h)}} - \frac{\cos\theta\hat{\mathbf{x}}_{h_1} + \sin\theta\hat{\mathbf{x}}_{h_2}}{\sqrt{1+2y_h\sin\theta\cos\theta}}.\end{aligned} \tag{57}$$

If the first and second term in the right hand side of (57) are denoted A and B, respectively, then the square of the 2-norm of $\Delta\mathbf{u}_{h_1}$ is given by

$$\begin{aligned}\Delta\mathbf{u}_{h_1}^T\Delta\mathbf{u}_{h_1} &= A^TA - A^TB - B^TA + B^TB \\ &= 1 - 2A^TB + 1 \\ &= 2 - \frac{2(\hat{\mathbf{x}}_{h_1}^T\hat{\mathbf{x}}_{h_1}\cos\theta + \hat{\mathbf{x}}_{h_2}^T\hat{\mathbf{x}}_{h_2}\sin\theta \pm \hat{\mathbf{x}}_{h_1}^T\hat{\mathbf{x}}_{h_2}\sin\theta \pm \hat{\mathbf{x}}_{h_2}^T\hat{\mathbf{x}}_{h_1}\cos\theta)}{\sqrt{2(1\pm y_h)}\sqrt{1+2y_h\sin\theta\cos\theta}} \\ &= 2 - \frac{2(1\pm y_h)(\cos\theta\pm\sin\theta)}{\sqrt{2(1\pm y_h)}\sqrt{1+2y_h\sin\theta\cos\theta}} \\ &= 2 - \sqrt{2(1\pm y_h)}\frac{\cos\theta\pm\sin\theta}{\sqrt{1+2y_h\sin\theta\cos\theta}}\end{aligned} \tag{58}$$

whose extremal θ values (denoted θ^*) will give the best case and the worst case conditions on X_h for the approximation of $\hat{\mathbf{u}}_{h_1}$. The problem of finding the θ^* values of (58) is equivalent to finding the θ^* values of

$$\begin{aligned}f(\theta) &= \frac{\cos\theta\pm\sin\theta}{\sqrt{1+2y_h\sin\theta\cos\theta}} \\ &= \mathrm{sgn}(\cos\theta\pm\sin\theta)\sqrt{\frac{1\pm\sin 2\theta}{1+y_h\sin 2\theta}}.\end{aligned} \tag{59}$$

Differentiating (59) with respect to θ (for $\cos\theta + \sin\theta \neq 0$, i.e., for $\theta \neq \frac{3\pi}{4} + n\pi$) gives

$$\begin{aligned} f'(\theta) &= \operatorname{sgn}(\cos\theta \pm \sin\theta)\frac{1}{2}\sqrt{\frac{1 + y_h \sin 2\theta}{1 \pm \sin 2\theta}}\frac{\pm 1 - y_h}{(1 + y_h \sin 2\theta)^2} 2\cos 2\theta \\ &= \operatorname{sgn}(\cos\theta \pm \sin\theta)\frac{(\pm 1 - y_h)\cos 2\theta}{\sqrt{1 \pm \sin 2\theta}(1 + y_h \sin 2\theta)^{3/2}}. \end{aligned} \tag{60}$$

Thus the candidate θ^* values are:

1. $\cos 2\theta^* = 0 \Longrightarrow 2\theta^* = \frac{\pi}{2} + n\pi = \frac{(2n+1)\pi}{2} \Longrightarrow \theta^* = \frac{(2n+1)\pi}{4}$
2. $\sin 2\theta^* = -1 \Longrightarrow 2\theta^* = \frac{3\pi}{2} + 2n\pi = \frac{(4n+3)\pi}{2} \Longrightarrow \theta^* = \frac{(4n+3)\pi}{4}$

   OR

   $\sin 2\theta^* = 1 \Longrightarrow 2\theta^* = \frac{\pi}{2} + n\pi = \frac{(2n+1)\pi}{2} \Longrightarrow \theta^* = \frac{(2n+1)\pi}{4}$
3. $\cos\theta^* + \sin\theta^* = 0 \Longrightarrow \theta^* = \frac{(4n+3)\pi}{4}$

Hence it can be concluded that the candidate θ^* values are $\theta^* = \frac{(2n+1)\pi}{4}$ (odd multiples of 45°). (Note that because the roles of $\cos\theta$ and $\sin\theta$ can be switched in (58), and because $\cos(\frac{\pi}{2} - \theta) = \sin\theta$ and $\sin(\frac{\pi}{2} - \theta) = \cos\theta$, it follows that $f(\theta) = f(\frac{\pi}{2} - \theta)$. Hence if θ^* is an extremal θ value, so is $\frac{\pi}{2} - \theta^*$.)

From (58), it can be observed that $\|\Delta\mathbf{u}_{h_1}\|^2 = 2 - 2\hat{\mathbf{u}}_{h_1}^T \tilde{\hat{\mathbf{u}}}_{h_1}$. Hence the extremal θ^* values for $\|\Delta\mathbf{u}_{h_1}\|$ are exactly the extremal values for $\hat{\mathbf{u}}_{h_1}^T \tilde{\hat{\mathbf{u}}}_{h_1}$. If the components of $\hat{\mathbf{x}}_{h_1}$ and $\hat{\mathbf{x}}_{h_2}$ are allowed to take positive as well as negative values, then $0 \leq \phi \leq \pi$ (recall that ϕ is the angle between the high-resolution image vectors). If $\pi/2 < \phi \leq \pi$, then

$$\hat{\mathbf{u}}_{h_1}^T \tilde{\hat{\mathbf{u}}}_{h_1} = \frac{(1 - y_h)(\cos\theta - \sin\theta)}{\sqrt{2(1 - y_h)(1 + y_h \sin 2\theta)}}. \tag{61}$$

Table 6 gives the best case and the worst case scenarios for different θ^* values for this case. If the components of $\hat{\mathbf{x}}_{h_1}$ and $\hat{\mathbf{x}}_{h_2}$ are all non-negative, then $0 \leq \phi \leq \frac{\pi}{2}$ and $0 \leq \theta \leq \frac{\pi}{2}$. In this case,

$$\hat{\mathbf{u}}_{h_1}^T \tilde{\hat{\mathbf{u}}}_{h_1} = \frac{(1 + y_h)(\cos\theta - \sin\theta)}{\sqrt{2(1 + y_h)(1 + y_h \sin 2\theta)}}. \tag{62}$$

Thus the only θ^* value in this case is 45°, which gives the best-case scenario as shown in Table 7. It is also required to check the boundary condition θ values (0 and $\frac{\pi}{2}$), which give

Table 6: $\hat{\mathbf{u}}_{h_1}^T \tilde{\mathbf{u}}_{h_1}$ and $\|\Delta \mathbf{u}_{h_1}\|$ for different θ^* values when $\pi/2 < \phi \leq \pi$

Case #	θ^*	$\hat{\mathbf{u}}_{h_1}^T \tilde{\mathbf{u}}_{h_1}$	$\|\Delta \mathbf{u}_{h_1}\|$	Comments
1	$\frac{\pi}{4}$	0	$\sqrt{2}$	worst case ($\mathbf{u}$'s are $\perp$)
2	$\frac{3\pi}{4}$	-1	2	$\mathbf{u}$'s in opposite directions
3	$\frac{5\pi}{4}$	0	$\sqrt{2}$	similar to the first case
4	$\frac{7\pi}{4}$	1	0	best case ($\mathbf{u}$'s line up)

the worst-case scenarios (refer to Table 7) for the degree of error depending on the value of y_h, with the results being worst when $y_h = 0$. However, note that as $y_h \to 0$, the problem of finding the high-resolution eigenimages becomes ill-defined. It is interesting to observe

Table 7: $\hat{\mathbf{u}}_{h_1}^T \tilde{\mathbf{u}}_{h_1}$ and $\|\Delta \mathbf{u}_{h_1}\|$ for different θ^* values when $0 \leq \phi \leq \frac{\pi}{2}$

Case #	θ^*	$\hat{\mathbf{u}}_{h_1}^T \tilde{\mathbf{u}}_{h_1}$	$\|\Delta \mathbf{u}_{h_1}\|$	Comments
1	0	$\sqrt{\frac{1+y_h}{2}}$	$2 - \sqrt{2(1+y_h)}$	worst case
2	$\frac{\pi}{4}$	1	0	best case ($\mathbf{u}$'s line up)
3	$\frac{\pi}{2}$	$\sqrt{\frac{1+y_h}{2}}$	$2 - \sqrt{2(1+y_h)}$	worst case

that the best and the worst case θ^* values in Tables 6 and 7 are interchanged. This occurs because at the boundary condition ϕ value of $\pi/2$, the problem of finding the right singular vectors of X_h becomes ill-defined and the corresponding right singular vectors at either side of $\phi = \pi/2$ are flipped. (Note that for image data matrices with only two images, the θ^* values for the approximation of the correct second eigenimage, $\tilde{\hat{\mathbf{u}}}_{h_2}$ remains the same.)

It is instructive to consider the worst case θ^* values in Table 7. Consider the case when X_h has all non-negative entries. Then the two extreme θ^* values are 0 and $\pi/2$. When $\theta^* = 0$,

$$\begin{aligned} U_l \Sigma_l &= X_l V_l \\ &= \begin{bmatrix} \mathbf{x}_{l_1} & \mathbf{x}_{l_2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mathbf{u}}_{l_1} & \hat{\mathbf{u}}_{l_2} \end{bmatrix} \begin{bmatrix} \|\mathbf{u}_{l_1}\| & 0 \\ 0 & \|\mathbf{u}_{l_2}\| \end{bmatrix} \end{aligned} \tag{63}$$

where $\mathbf{u}_{l_1} = \mathbf{x}_{l_1}$ and $\mathbf{u}_{l_2} = \mathbf{x}_{l_2}$. By definition, left singular vectors are orthonormal to each other. Thus,

$$\hat{\mathbf{u}}_{l_1} \cdot \hat{\mathbf{u}}_{l_2} = 0 \;\Rightarrow\; \frac{\mathbf{x}_{l_1}^T \mathbf{x}_{l_2}}{\|\mathbf{x}_{l_1}\| \cdot \|\mathbf{x}_{l_2}\|} = 0 \;\Rightarrow\; \mathbf{x}_{l_1}^T \mathbf{x}_{l_2} = 0. \tag{64}$$

Similarly, when $\theta^* = \pi/2$,

$$\begin{aligned} U_l \Sigma_l &= X_l V_l \\ &= \begin{bmatrix} \mathbf{x}_{l_1} & \mathbf{x}_{l_2} \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mathbf{u}}_{l_1} & \hat{\mathbf{u}}_{l_2} \end{bmatrix} \begin{bmatrix} \|\mathbf{u}_{l_1}\| & 0 \\ 0 & \|\mathbf{u}_{l_2}\| \end{bmatrix} \end{aligned} \tag{65}$$

where $\mathbf{u}_{l_1} = \mathbf{x}_{l_2}$ and $\mathbf{u}_{l_2} = -\mathbf{x}_{l_1}$. Again,

$$\hat{\mathbf{u}}_{l_1} \cdot \hat{\mathbf{u}}_{l_2} = 0 \;\Rightarrow\; \frac{-\mathbf{x}_{l_2}^T \mathbf{x}_{l_1}}{\|\mathbf{x}_{l_2}\| \cdot \| - \mathbf{x}_{l_1}\|} = 0 \;\Rightarrow\; \mathbf{x}_{l_1}^T \mathbf{x}_{l_2} = 0. \tag{66}$$

Thus, the worst-case scenarios occur only when there is no overlap between the reduced image vectors at low resolution. To check for the conditions on the corresponding high-resolution image vectors, let $m = 4$ and $r = 2$. Then, $\mathbf{x}_{l_1}^T \mathbf{x}_{l_2} = 0 \Rightarrow$

$$\hat{\mathbf{x}}_{h_1}^T \hat{\mathbf{x}}_{h_2} + x_{11}x_{22} + x_{21}x_{12} + x_{31}x_{42} + x_{41}x_{32} = 0. \tag{67}$$

With all the components of X_h non-negative, all the individual products on the LHS must be 0 to satisfy the condition in (67), which is highly unlikely for most images. Similar results can be obtained when X_h is allowed to have both positive and negative entries. Thus, it can be concluded that using low-resolution right singular vectors is a better method than interpolating low-resolution eigenimages for approximating high-resolution eigenimages of X_h.

While the above analysis cannot be easily extended to arbitrary X_h, one can experimentally evaluate the quality of the eigenimage approximation. It was shown in Section 3.2.4 that using the low-resolution right singular vectors give a much better approximation of the high-resolution eigenimages than those obtained using the interpolated low-resolution eigenimages. This motivates several changes to Chang's eigendecomposition algorithm, which is narrated in the next chapter.

CHAPTER IV

OVERVIEW OF CHANG'S ALGORITHM

As indicated earlier, Chang's eigendecomposition algorithm [45] is the fastest known algorithm for computing the first k approximate eigenimages of correlated images to the user-specified accuracy. This chapter gives an overview of that algorithm, along with its computational efficiency. In particular, the next section explains that the eigendecomposition of circulant matrices can be given in closed form. Section 4.2 shows that this eigendecomposition holds true for planar rotated images and can be generalized to arbitrary 3-dimensional image sequences. Finally, Section 4.3 uses these properties to outline the algorithm with its computational expense.

4.1 Eigendecomposition of Circulant Matrices

A circulant matrix [55] is defined as

$$\text{circ}(a_1, a_2, a_3, ..., a_n) = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ a_n & a_1 & \cdots & a_{n-1} \\ \vdots & \vdots & \ddots & \cdots \\ a_2 & a_3 & \cdots & a_1 \end{bmatrix}. \tag{68}$$

Thus row $i+1$ of a circulant matrix can be obtained by a right-circular shift of row i (the first row is a right-circular shift of the last one). It is useful to represent circulant matrices with the following fundamental permutation matrix:

$$\Pi = \text{circ}(0, 1, 0, \cdots, 0). \tag{69}$$

Then the matrix in (68) can be represented by

$$\text{circ}(a_1, a_2, ..., a_n) = a_1 \times I + a_2 \times \Pi + ... + a_n \times \Pi^{n-1}. \tag{70}$$

The Fourier matrix F of dimension n is defined as

$$F = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \tag{71}$$

where $\omega = \exp(\frac{-2\pi j}{n})$, $FF^* = F^*F = I$, and the k^{th} column of F is denoted $\mathbf{f}_k$, elements of which are given by

$$\mathbf{f}_k(x) = \frac{1}{\sqrt{n}} e^{-j2\pi kx/n} \tag{72}$$

with $0 \leq x \leq n-1$.

It is easy to show that the eigendecomposition of a circulant matrix can be written using the Fourier matrix. In particular, let $\Omega = diag(1, \omega, \omega^2, \cdots, \omega^{n-1})$, then $\Pi = F\Omega F^*$, and

$$\begin{aligned} \text{circ}(a_1, a_2, ..., a_n) &= a_1 I + a_2 \Pi + ... + a_n \Pi^{n-1} \\ &= a_1 FF^* + a_2 F\Omega F^* + ... + a_n (F\Omega F^*)^{n-1} \\ &= F(a_1 I + a_2 \Omega + ... + a_n \Omega^{n-1}) F^* \\ &= F\Lambda F^*, \end{aligned} \tag{73}$$

where $\Lambda = (a_1 I + a_2 \Omega + \cdots + a_n \Omega^{n-1})$ is a diagonal matrix with its k^{th} diagonal element given by

$$\begin{aligned} \lambda_k &= a_1 + a_2 \omega^{(k-1)} + a_3 \omega^{2(k-1)} + \cdots + a_n \omega^{(n-1)(k-1)} \\ &= \textstyle\sum_{j=1}^{n} a_j \omega^{(k-1)(j-1)}, \end{aligned} \tag{74}$$

which is the DFT of the sequence $a_1, a_2, \cdots, a_n$. Equation (73) is the expression of the eigendecomposition of the circulant matrix $\text{circ}(a_1, a_2, \cdots, a_n)$, where the eigenvectors are the columns of the Fourier matrix F and the eigenvalues are given by (74).

Consider $\text{circ}(a_1, a_2, \cdots, a_n)$, in which the p^{th} row is "circularly symmetric" with respect to the p^{th} element in the row. That is, if $\oplus$ denotes the circular shift operator, then for the p^{th} row, the $(p \oplus k)^{\text{th}}$ element equals to the $(p \oplus -k)^{\text{th}}$ element, for any integer k. In this case, it is easy to show that the eigenvalues of $\text{circ}(a_1, a_2, \cdots, a_n)$ are real. In particular,

evaluating the imaginary part of (74) results in

$$\begin{aligned} Im(\lambda_k) &= Im\left(\sum_{l=1}^{n} a_l \omega^{(k-1)(j-1)}\right) \\ &= \sum_{l=1}^{n} a_l \sin\left(\frac{(2\pi)(k-1)(l-1)}{n}\right) \\ &= 0 \end{aligned} \tag{75}$$

This also leads to the fact that the eigenvalues are repeated where $\lambda_{1+k} = \lambda_{n+1-k}$, for $k = 1, 2, \cdots, \frac{n}{2}$. By using the proper linear combination of the $(1+k)^{\text{th}}$ and $(n+1-k)^{\text{th}}$ columns of F, one can obtain a real eigendecomposition of $\text{circ}(a_1, a_2, \cdots, a_n)$, i.e.,

$$\text{circ}(a_1, a_2, \cdots, a_n) = HDH^T \tag{76}$$

where D is the $n \times n$ matrix given by

$$D = \text{diag}(\lambda_1, \lambda_2, \lambda_2, \lambda_3, \lambda_3, ...) \tag{77}$$

and H consists of the first n columns of

$$\sqrt{2}\left[\frac{1}{\sqrt{2}}\mathbf{f}_1 \;\; \Re\mathbf{f}_2 \;\; \Im\mathbf{f}_2 \;\; \Re\mathbf{f}_3 \;\; \Im\mathbf{f}_3 \;\; \ldots\right] =$$

$$\sqrt{\frac{2}{n}}\begin{bmatrix} \frac{1}{\sqrt{2}} & c_0 & -s_0 & c_0 & -s_0 & \cdots \\ \frac{1}{\sqrt{2}} & c_1 & -s_1 & c_2 & -s_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \frac{1}{\sqrt{2}} & c_{n-1} & -s_{n-1} & c_{2(n-1)} & -s_{2(n-1)} & \cdots \end{bmatrix} \tag{78}$$

where $\Re\mathbf{f}_k$ and $\Im\mathbf{f}_k$ denote the real and the imaginary part of $\mathbf{f}_k$, respectively, $c_k = \cos(k\frac{2\pi}{n})$, and $s_k = \sin(k\frac{2\pi}{n})$. It is clear that the columns in H are sinusoidal sequences, arranged from a DC signal to higher harmonics. If n, the dimension of H, is odd, then the eigenvalues and eigenvectors of $\text{circ}(a_1, a_2, \cdots, a_n)$ come in pairs except for the first one. If n is even, then the eigenvalue $\lambda_{\frac{n+2}{2}}$ does not repeat and its corresponding eigenvector is

$$\begin{aligned} \mathbf{h}_n &= \tfrac{1}{\sqrt{n}}[1 \quad \omega^{\frac{n}{2}} \quad \omega^{2\frac{n}{2}} \quad \cdots \quad \omega^{(n-1)\frac{n}{2}}]^T \\ &= \tfrac{1}{\sqrt{n}}[1 \quad \exp(\tfrac{-2\pi j}{n}\tfrac{n}{2}) \quad \exp(\tfrac{-2\pi j}{n}\tfrac{2n}{2}) \quad \cdots \quad \exp(\tfrac{-2\pi j}{n}\tfrac{(n-1)n}{2})]^T \\ &= \tfrac{1}{\sqrt{n}}[1 \quad -1 \quad 1 \quad \cdots \quad -1]^T \end{aligned} \tag{79}$$

4.2 *Prelude to Chang's Algorithm*

Recall that the image data matrix is in the form of $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$. Consider X where each $\mathbf{x}_{i+1}$ is obtained from $\mathbf{x}_i$ by a planar rotation[1] of $\theta = 2\pi/n$. The correlation matrix $X^T X$ is given by

$$X^T X = \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \cdots & \mathbf{x}_1^T\mathbf{x}_n \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \cdots & \mathbf{x}_2^T\mathbf{x}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_n^T\mathbf{x}_1 & \mathbf{x}_n^T\mathbf{x}_2 & \cdots & \mathbf{x}_n^T\mathbf{x}_n \end{bmatrix}. \tag{80}$$

It is easy to show that $X^T X$ is a *circulant matrix* and its rows are *circularly symmetric* [45]. Hence its eigendecomposition is given by Equations (76), (77), and (78).

The expression for $X^T X$ can alternatively be given in terms of the singular value decomposition of X, i.e.,

$$\begin{aligned} X^T X &= \left(U\Sigma V^T\right)^T U\Sigma V^T \\ &= V\Sigma^T U^T U\Sigma V^T \\ &= V\Sigma^2 V^T. \end{aligned} \tag{81}$$

This expression for $X^T X$ along with its eigendecomposition reveals that Σ and V corresponding to an unordered SVD of X can be computed in a closed form. In particular, the square roots of the diagonal entries of D are the singular values of X, and $V = H$.

To compute the left singular vectors of X, observe that

$$\begin{aligned} XV &= U\Sigma V^T V \\ &= U\Sigma \\ &= [\sigma_1\hat{\mathbf{u}}_1, \sigma_2\hat{\mathbf{u}}_2, ..., \sigma_n\hat{\mathbf{u}}_n]. \end{aligned} \tag{82}$$

The multiplication of X with V $(= H)$ can be preformed efficiently using Fast Fourier Transform (FFT) techniques. In particular, if Y is a matrix whose i^{th} row is the FFT of the i^{th} row of X, then $Y = \sqrt{n}XF$. Therefore, the matrix XH can be formed from the

[1] More precisely, the image $i+1$ is obtained by rotating the infinite-resolution image represented by the i^{th} image, and then sampling it.

first n columns of Y as

$$\begin{aligned} XH &= X\sqrt{2}\left[\tfrac{1}{\sqrt{2}}\mathbf{f}_1 \;\; \Re\mathbf{f}_2 \;\; \Im\mathbf{f}_2 \;\; \Re\mathbf{f}_3 \;\; \Im\mathbf{f}_3 \;\; \ldots\right] \\ &= \sqrt{2}\left[\tfrac{1}{\sqrt{2}}X\mathbf{f}_1 \;\; \Re\{X\mathbf{f}_2\} \;\; \Im\{X\mathbf{f}_2\} \;\; \Re\{X\mathbf{f}_3\} \;\; \Im\{X\mathbf{f}_3\} \;\; \ldots\right] \\ &= \sqrt{\tfrac{2}{n}}\left[\tfrac{1}{\sqrt{2}}\mathbf{y}_1 \;\; \Re\mathbf{y}_2 \;\; \Im\mathbf{y}_2 \;\; \Re\mathbf{y}_3 \;\; \Im\mathbf{y}_3 \;\; \ldots\right]. \end{aligned} \tag{83}$$

The above development means that an unordered SVD for planar rotated image sequence can be given in closed form. In particular, the right singular vectors are pure sinusoids of frequencies that are multiples of $2\pi/n$ radians and the dominant frequencies of the power spectra of the (ordered) right singular vectors increase linearly with their index. The true eigenimages can then be calculated using (82) and (83).

Although the above eigendecomposition analysis does not hold true for arbitrary image sequence, it can be shown that the analytical expressions for planar rotation can serve as good approximations for general 3-D cases. To explore the consequence of 3-D effects, the first video sequence in Fig. 3 is used here. The center object in this video sequence is a slow-moving toy train. Fig. 16 shows five of the $n = 150$ images that make up the image data matrix X, as well as the singular vectors of X. Though the results of planar rotation do not apply here, the following two properties are again immediately apparent:

1. The right singular vectors are well-approximated by sinusoids of frequencies that are multiples of $2\pi/n$ radians, and the magnitude-squared of the spectra, i.e., the "power spectra" of the right singular vectors consist of a narrow band around the corresponding dominant harmonics.

2. The dominant frequencies of the power spectra of the (ordered) singular vectors increase approximately linearly with their index.

These properties (assuming they hold) suggest an approach for reducing the expense in computing the eigendecomposition to within a prespecified accuracy. In particular, the first property means that the singular vectors are approximately spanned by a handful of harmonics. In addition, if the second property holds, then the frequencies of the dominant harmonics can be quickly identified by simply searching from low to high frequencies. Consequently, by projecting the row space of X to a smaller subspace spanned by a few

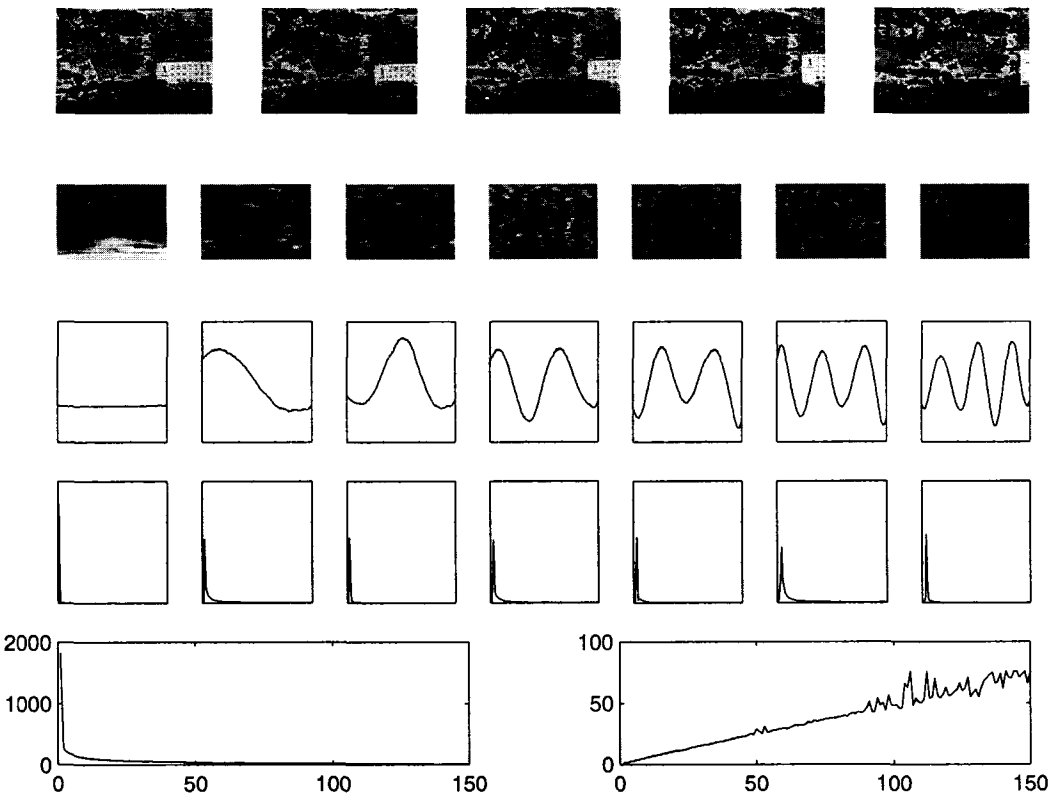


Figure 16: This figure shows the eigendecomposition of the image matrix X obtained from the video sequence that consists of images of slow-moving toy train. The first row shows five of the 150 images of the image data matrix X. The second row shows the first seven eigenimages (left singular vectors of X) using the same gray scale encoding, with white denoting the maximum positive pixel value and black denoting the maximum negative value. The third row shows the first seven right singular vectors of X. The fourth row shows the FFT magnitude-squared, i.e., the "power spectra" of these right singular vectors. It is apparent that though the right singular vectors of X are not pure sinusoids, their power spectra are concentrated in a narrow band around frequencies that are harmonics of $2\pi/n$. The plot on the left in the last row shows the singular values of X, while the plot on the right shows the frequency at which the power spectra of the corresponding right singular vectors achieves a maximum (i.e., the "dominant" frequencies). It can be seen that the dominant frequencies of the power spectra of the right singular vectors corresponding to nonzero singular values increase approximately linearly with their index.

of the harmonics, the computational expense associated with the SVD computation can be significantly reduced. Note that this approach can be used to generate the SVD to within any prespecified accuracy; the deviation of the actual singular vectors from pure harmonics only affects the computational savings that this approach offers. (Note that circulancy, by itself, only guarantees property (1); thus circulancy is not sufficient for this approach to work well. However, the discussion on circulant matrices provides a sound theoretical basis for this approach.) It is shown in [45] that the two properties discussed above hold true for sequences of images whose contents vary slowly, independent of the underlying transformation (rotation, translation, scaling, etc.) and these properties motivated Chang to propose a fast eigenspace decomposition technique, the details of which are provided in the next section.

4.3 Chang's Eigendecomposition Algorithm

The objective of Chang's algorithm is to determine the first k left singular vectors of X. Let p be such that the power spectra of the first k singular vectors are essentially restricted to the band $[0, 2\pi p/n]$. Owing to the properties of the singular vectors discussed in the previous section, p is typically not much larger than k. Let H_p denote the matrix comprising the first p columns of H (i.e., the first p columns of the matrix given in (78)). Then the first k singular values $\tilde{\sigma}_1, \cdots, \tilde{\sigma}_k$ and the corresponding left singular vectors $\tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k$ of XH_p serve as excellent estimates to those of X. (Note that XH_p typically has far fewer columns than X, so that its SVD can be computed much more quickly.) Moreover, the accuracy of the approximated singular vectors with power spectra concentrated around "lower" frequencies will tend to be better, i.e., the smaller i is, the better estimate $\tilde{\hat{\mathbf{u}}}_i$ is of $\hat{\mathbf{u}}_i$. Thus, it can be shown that for any p, the error $\rho(X, \hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_k) - \rho(X, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k)$ increases as k increases from 1 to p [45].

The ultimate goal is to guarantee, upon termination, that $\rho(X, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k)$ exceeds a user-specified threshold μ. While $\rho(X, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k)$ depends critically on k and $\tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k$, neither of which are available a priori, it has been shown [45] that

$$\rho(X, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_p) \geq \rho(X^T, \mathbf{h}_1, \cdots, \mathbf{h}_p) \tag{84}$$

where $\mathbf{h}_i$ denotes the i^{th} column of H.

In summary, when p is chosen so as to satisfy $\rho(X^T, \mathbf{h}_1, \cdots, \mathbf{h}_p) \geq \mu$, the quantity $\rho(X, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k)$ turns out to exceed μ for some $k \leq p$, with $\tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k$ being very good estimates for $\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_k$, and $\tilde{\sigma}_1, \cdots, \tilde{\sigma}_k$ being very good estimates for $\sigma_1, \cdots, \sigma_k$. The energy recovery ratio $\rho(X, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k)$ can be efficiently approximated by $\sum_{i=1}^{k} \tilde{\sigma}_i^2 / \|X\|_F^2$.

The entire algorithm for the fast computation of a partial SVD of X can be summarized as follows:

1. Form the matrix Y, whose i^{th} row is the FFT of the i^{th} row of X.

2. Determine the smallest number p such that $\rho(X^T, \mathbf{h}_1, \cdots, \mathbf{h}_p) > \mu$, where μ is the user-specified reconstruction ratio. The key observation here is that the matrix XH_p can be constructed as the first p columns of the matrix $\sqrt{\frac{2}{n}}[\frac{1}{\sqrt{2}}\mathbf{y}_0 \ \Re\mathbf{y}_1 \ \Im\mathbf{y}_1 \ \Re\mathbf{y}_2 \ \Im\mathbf{y}_2 \cdots]$, where $\mathbf{y}_i$ denotes the i^{th} column of Y.

3. With Z_p denoting the first p columns of the matrix $[\frac{1}{\sqrt{2}}\mathbf{y}_0 \ \Re\mathbf{y}_1 \ \Im\mathbf{y}_2 \ \Re\mathbf{y}_2 \ \Im\mathbf{y}_2 \cdots]$, compute the SVD $Z_p = \sum_{i=1}^{p} \tilde{\sigma}_i \tilde{\mathbf{u}}_i \tilde{\mathbf{v}}_i^T$.

4. Return $\tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k$ such that $\rho(X, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k) \geq \mu$.

The computational expense of Chang's algorithm is now analyzed. The cost incurred in Step 1, i.e., performing the FFT of each row of X, requires $O(mn \log_2 n)$ flops. Step 2, that of estimating p, requires of $O(mp)$ flops. In Step 3, the cost of computing the SVD of the matrix comprising the first p columns of $\sqrt{\frac{n}{2}}XH$ is of order $O(mp^2)$. Step 4, determining the needed dimension k, requires $O(mnk)$ flops. If $p \ll n$, then the total computation required is approximately $O(mn \log_2 n)$. This compares very favorably with the direct SVD approach which requires $O(mn^2)$ flops, and in most cases with the updating SVD method [42] as well, which requires $O(mnk^2)$ flops. The computational savings offered by Chang's algorithm are significant if the condition

$$\rho(X^T, \mathbf{h}_1, \cdots, \mathbf{h}_p) = \frac{\sum_{i=1}^{p} \|X\mathbf{h}_i\|^2}{\|X\|_F^2} \geq \mu$$

holds for $p \ll n$. As the vectors $\mathbf{h}_i$ are harmonics consisting of increasing frequencies, this condition simply means that the individual pixel values do not change rapidly across the

sequence of images. This is typically the case, as evidenced by the examples presented in [45].

In summary, Chang's eigendecomposition algorithm is motivated by the observation that the SVD of X can be determined in a closed form when the images are derived by a planar rotation of a single image, thus resulting in $X^T X$ being circulant. It is shown that the analytical expressions for the eigendecomposition, based on the theory of circulant matrices, can serve as a good approximation to the eigendecomposition of arbitrary video sequences. The algorithm shows better computational efficiency, because the SVD is computed on a much smaller matrix XH_p, which is formed by reducing X in the temporal dimension. However, one can still improve this computational efficiency by operating on lower resolution images using the techniques explained in the previous chapter. The appropriate manner of downsampling to achieve these low-resolution images is the topic of the next chapter.

CHAPTER V

FAST EIGENDECOMPOSITION OF CORRELATED IMAGES USING THEIR LOW-RESOLUTION PROPERTIES

Chang's eigendecomposition algorithm (refer to the previous chapter) reduces X in the temporal dimension and thus gives better computational efficiency than the direct SVD of correlated images. However, the first two steps in Chang's algorithm, i.e., the calculation of the value p and the computation of the SVD of XH_p, still requires a significant amount of time, because the algorithm always works with the full spatial resolution of the images. Hence it is desirable to reduce the images in the spatial dimension first. It was shown in Chapter 3 that the low-resolution SVD can be used to approximate high-resolution eigenimages. Hence these properties are used here to modify Chang's algorithm so that its computational efficiency is improved without sacrificing the accuracy of the resulting eigendecomposition.

Chapter 3 considered two image reduction techniques, i.e., box filtering and bicubic interpolation, and it was shown that box filtering gave better results than bicubic interpolation when low-resolution right singular vectors were used to determine approximate high-resolution eigenimages. Therefore, in Chang's algorithm, images can be reduced using simple box filtering and the corresponding right singular vectors can be used to approximate the eigenimages of XH_p. However, a reduced resolution version of an image can also be determined by randomly sampling pixels from X and an analysis in Section 5.1 shows that downsampling by random sampling can be more effective than any low-pass filtering technique. This analysis motivated a fast SVD algorithm, outlined in Section 5.2, to quickly compute the desired portion of the eigendecomposition based on a user-specified measure

of accuracy. The performance of the proposed algorithm is then evaluated in Section 5.3 using both the image data sets considered in Section 3.2.4.

5.1 *Effect of Spatial Reduction Techniques*

Two spatial reduction methods, i.e., low-pass filtering techniques and random sampling, are considered in this section. The reduction of X using box filtering was already studied in Section 3.2.4. Although this technique gives a good approximation of high-resolution eigenimages, it is not known how it affects the spatial as well as the temporal properties of X. Hence this section introduces one more reduction technique, i.e., random sampling and compares its effect on X against the effect of low-pass filtering techniques on X, using mathematical analysis and empirical results.

5.1.1 Illustrative Examples

This subsection presents two artificial examples to illustrate the behavior of low-pass filtering as compared to random sampling. For these examples, box filtering was again used to simplify the illustration. Fig. 17 shows the two "high-resolution" image sequences with images of size 4×4 each. The corresponding box-filtered image frames (of size 2×2 each) are shown in Fig. 18.

The sequence in Fig. 17(a) was selected to illustrate the worst case for box filtering. It consists of image frames corresponding to a checkerboard with all pixels changing their intensity between the two extreme values from one frame to the next. Hence all four temporal frequencies are required to attain any user-specified reconstruction ratio for this sequence ($p = 4$). However, for the corresponding box-filtered sequence in Fig. 18(a), only the zero frequency is required to attain any reconstruction ratio ($p_b = 1$),[1] thus giving the worst possible approximation of the original temporal properties. Now consider reducing the original sequence by randomly sampling 4 out of 16 pixels in all the images. Because the same permutation of four pixels is used over all four image frames, the reduced sequence will always give $p_r = 4$.

[1]The values of p for box-filtered, Gaussian-filtered and randomly sampled sequences are referred to as p_b, p_g, and p_r, respectively.

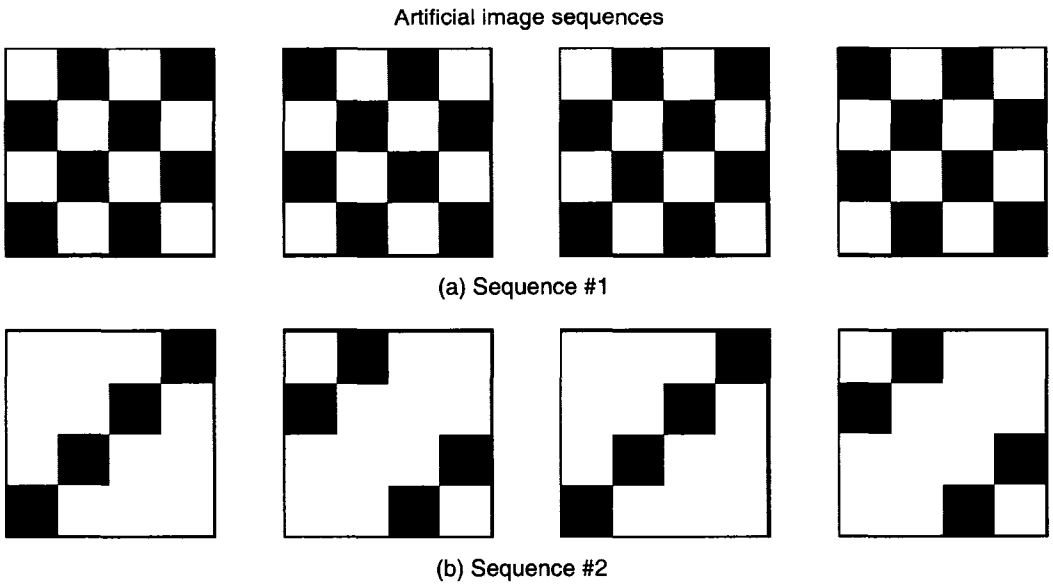


Figure 17: This figure shows two artificial image sequences that are used to compare box filtering with random sampling.

Now consider the sequence in Fig. 17(b) which was selected to illustrate the possible overestimation of true p value by box filtering. Fig. 19(a) shows the corresponding values of the true p when μ is varied from 0.8 to 0.99 in steps of 0.01. It is easy to see that the corresponding box-filtered image frames in Fig. 18(b) give $p_b = 4$ for any reconstruction ratio and thus unnecessarily overestimates the true p value by 3 for $\mu \leq 0.83$. Now consider reducing this sequence using random sampling. There are $\binom{16}{4} = 1820$ such realizations and the corresponding p_r values are plotted in Fig. 19(b). It is easy to see that the median p_r equals the true p for all values of μ. Thus random sampling performs very well even in this case. As the following subsection will show, this is true in general, even when applied to real video sequences.

5.1.2 Special Case with Empirical Results

Consider an $m \times n$ image data matrix X such that $X^T X$ is circulant. Then the eigendecomposition of X follows from Section 4.2. Consider reducing X using box filtering and random sampling to obtain the corresponding low-resolution image data matrices (X_b and

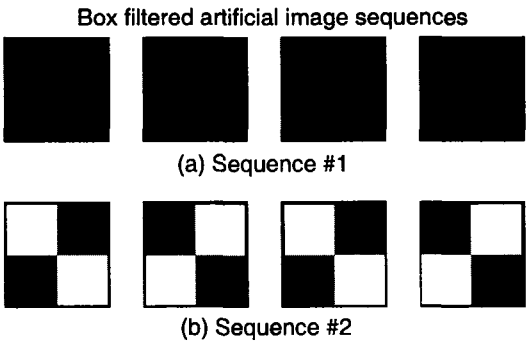


Figure 18: This figure shows the box filtered image sequences in Fig. 17 with a reduction factor of 2 in both directions.

X_r, respectively), with the reduction factor r. To determine how much these two reduction techniques affect the resulting eigendecomposition, consider the following assumptions:

1. m is an integer multiple of n.

2. the first set of n components of $\mathbf{x}_1$ is repeated throughout that image.

3. $r = n$.

With these assumptions, note that the correlation matrices, $X_b^T X_b$ and $X_r^T X_r$ are also circulant. Hence their eigendecomposition will yield the same expressions as for X. Although the eigenvectors of all the correlation matrices (i.e., right singular vectors of the original matrices) remain the same, their corresponding eigenvalues (i.e., squares of the singular values of the original matrices) will be different and hence, the importance of the singular vectors will depend on the ratio of the corresponding singular values. A simple example is considered here that explains how both the reduction methods affect the ratios between the eigenvalues.

Consider the following high-resolution image data matrix:

$$X = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_1 \\ x_1 & x_2 \\ x_2 & x_1 \end{bmatrix}. \tag{85}$$

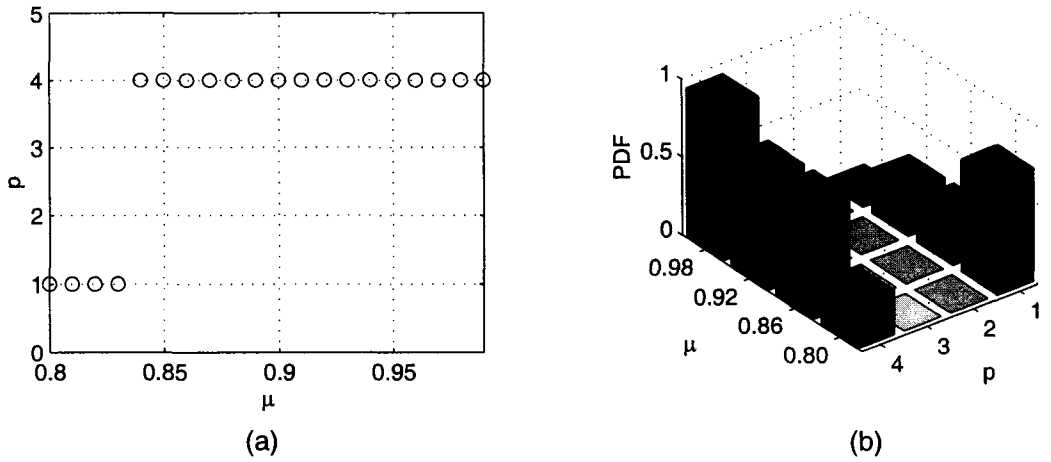


Figure 19: This figure shows the results for the image sequence shown in Fig. 17(b). Part (a) plots the true p values against μ, while part (b) plots the values of p_r for different values of μ for all $\binom{16}{4} = 1820$ realizations using random sampling. (The value of p_b for the corresponding box filtered sequence in Fig. 18(b) always remains 4 for all values of μ.)

Because $X^T X$ is circulant, its eigenvalues can be calculated using (77) and are given by

$$\begin{aligned} e_1 &= \sum_{i=0}^{1} \mathbf{x}_1^T \mathbf{x}_{i+1} = 2(x_1 + x_2)^2, \\ e_2 &= \sum_{i=0}^{1} \mathbf{x}_1^T \mathbf{x}_{i+1} e^{-j\pi i} = 2(x_1 - x_2)^2. \end{aligned} \tag{86}$$

Now if the images in X are reduced using box filtering and random sampling with $r = 2$, the corresponding low-resolution image data matrices are given by

$$\begin{aligned} X_b &= \frac{1}{2} \begin{bmatrix} x_1 + x_2 & x_1 + x_2 \\ x_1 + x_2 & x_1 + x_2 \end{bmatrix}, \\ X_r &= \begin{bmatrix} x_1 & x_2 \\ x_2 & x_1 \end{bmatrix}. \end{aligned} \tag{87}$$

Note that the correlation matrices, $X_b^T X_b$ and $X_r^T X_r$ are also circulant and (77) can be used to calculate their eigenvalues. The eigenvalues of $X_b^T X_b$ are given by

$$\begin{aligned} e_{b_1} &= (x_1 + x_2)^2, \\ e_{b_2} &= 0, \end{aligned} \tag{88}$$

while the eigenvalues of $X_r^T X_r$ are given by

$$\begin{aligned} e_{r_1} &= (x_1 + x_2)^2, \\ e_{r_2} &= (x_1 - x_2)^2. \end{aligned} \tag{89}$$

Note that $\frac{e_1}{e_2} = \frac{e_{r_1}}{e_{r_2}}$, while $\frac{e_{b_1}}{e_{b_2}} = \infty$. Hence, it can be observed that box filtering makes the first singular vector more dominant than it actually is for the original image data matrix X, while the random pixel selection keeps the importance of both the singular vectors the same.

One can argue that, for the specific X used in (85), if X_r consists of the first and the third row of X, then the ratio of the eigenvalues of $X_r^T X_r$ will also become ∞.[2] To solve this problem, the following procedure can be performed:

1. Generate an empty matrix E.
2. Generate X_r from X.
3. Compute the eigenvalues (e_{r_1}, e_{r_2}) and the eigenvectors $(\hat{\mathbf{v}}_{r_1}, \hat{\mathbf{v}}_{r_2})$ of $X_r^T X_r$.
4. Scale both the eigenvectors using

$$\mathbf{v}_{r_i} = \frac{e_{r_i}}{\sqrt{e_{r_1}^2 + e_{r_2}^2}} \hat{\mathbf{v}}_{r_i}, \tag{90}$$

   for $i = 1, 2$.
5. Update the matrix $E = [E, \mathbf{v}_{r_1}, \mathbf{v}_{r_2}]$.
6. Repeat Steps 2 through 5 and compute the reduced SVD of $E_{(2\times P)}$, i.e., $E_{(2\times P)} = (U_e)_{(2\times 2)}(S_e)_{(2\times P)}(V_e)^T_{(P\times P)}$, where P denotes the number of columns in E.
7. Repeat Step 6 until the ratio of the two singular values in S_e converges to some value.

Although it is difficult to show the symbolic calculations, it can be observed that after a few iterations, $\frac{e_{r_1}}{e_{r_2}} \rightarrow \frac{e_1}{e_2}$. Thus it can be concluded that the above procedure gives a good

[2] $X_r^T X_r$ will not be circulant in this case, but it is still easy to solve for its eigenvalues.

approximation of the right singular vectors of X and their relative importance remains the same as long as "enough" random samplings are considered.

One can also argue that the specific X used in (85) gives a singular matrix X_b, hence it affects the further analysis of finding the relative importance of the corresponding right singular vectors. However, the empirical results shown in Fig. 20 for arbitrary video sequences also depicts similar behavior. For all the video sequences, the p_b values decrease rapidly even for small reduction factors, indicating that box filtering makes the first few frequency components in the low-resolution images more dominant than they actually are in the high-resolution images. On the other hand, the p_r values are (almost) always above the "true" p values for all the video sequences, indicating that random pixel selection does not significantly alter the temporal properties of high-resolution images. This behavior is quite intuitive due to working with the pixels themselves rather than their averages. The mathematical analysis given in the following subsection explains why spatial reduction of image frames using any low-pass filtering technique will have an undesired effect on the temporal frequencies of the original video sequence.

5.1.3 Mathematical Analysis

Objects in a video sequence typically experience relatively simple motions, e.g., translations, rotations, scaling, etc. Consequently, low pass filtering the images of a video sequence tends to attenuate temporal as well as spatial high frequency information content. This observation will be illustrated with a mathematical example of a video sequence of an object experiencing a constant linear motion. For simplicity, assume that a video sequence consists of a single object in a black background moving in the negative x-direction. To further simplify the equations, assume without loss of generality that the object is moving with a velocity of one pixel per frame and that the video runs at a rate of one frame per second. If the object is represented by a function $f(x, y)$, the image at time t is given by $f(x, y, t) = f(x + t, y)$. The 3-dimensional discrete Fourier transform (DFT) of the video sequence $f(x, y, t)$ can be determined directly from the 2-dimensional DFT representation

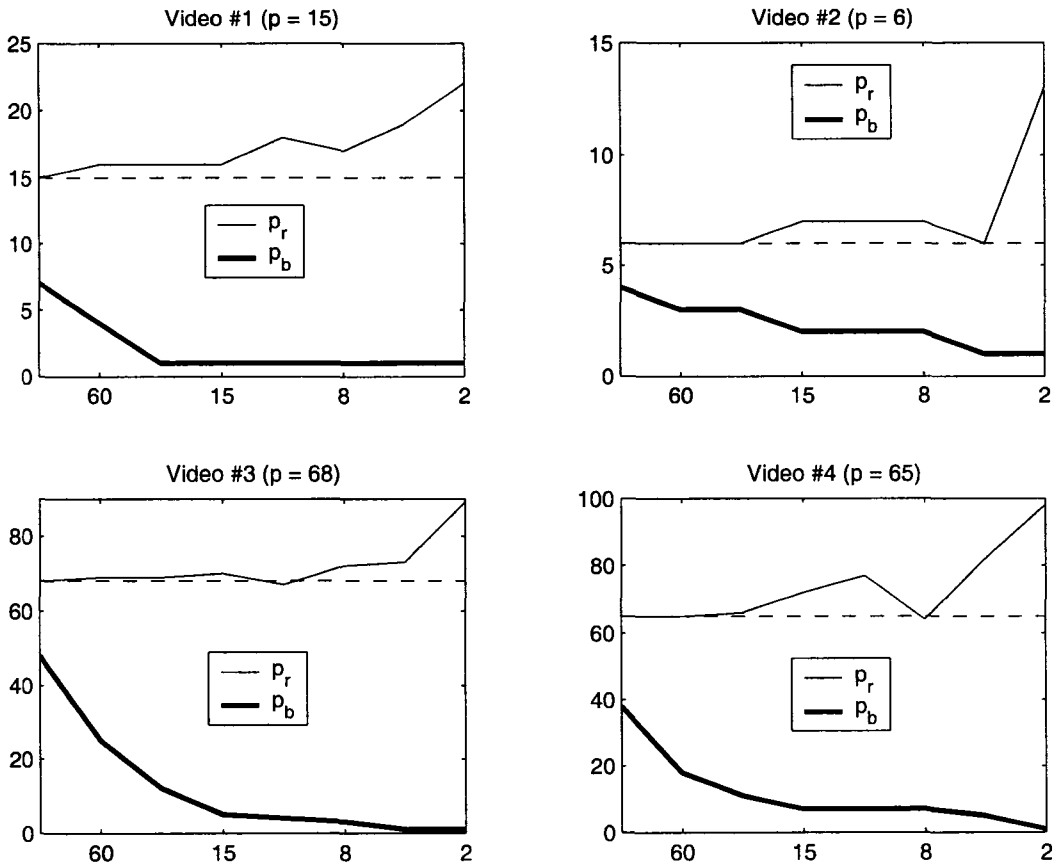


Figure 20: This figure shows the plots for the first four video sequences in Fig. 3. The image data matrices at different resolutions are formed after reducing the original images from $m = 240 \times 352$ to the lower resolutions of $120 \times 176, 60 \times 88, 30 \times 44, 15 \times 22, 10 \times 15, 8 \times 12, 4 \times 6$, and 2×3. Each subplot title gives the video number and its corresponding p value (plotted with a horizontal dashed line) at the highest resolution, where p is the smallest number of frequency harmonics required to obtain $\rho > 0.95$ in (26). The horizontal axis gives the resolution of low-resolution images in one dimension, while the plots p_r and p_b give the p values at the lower resolutions when the images are reduced using random pixel selection and box filtering, respectively. For the random pixel selection technique, the images are reduced for four different times to calculate four different p values and the maximum p value is assigned to p_r.

of $f(x, y)$, which is given by

$$f(x,y) = \frac{1}{N_x N_y} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} F(k_x, k_y) e^{j2\pi(x\frac{k_x}{N_x} + y\frac{k_y}{N_y})} \tag{91}$$

where N_x and N_y represent the numbers of rows and columns in an image, respectively and the $F(k_x, k_y)$ terms are the corresponding Fourier coefficients. More specifically,

$$\begin{aligned} f(x,y,t) &= \frac{1}{N_x N_y} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} F(k_x,k_y) e^{j2\pi((x+t)\frac{k_x}{N_x}+y\frac{k_y}{N_y})} \\ &= \frac{1}{N_x N_y N_t} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} \sum_{k_t=0}^{N_t-1} N_t F(k_x,k_y)\delta_{k_x k_t} e^{j2\pi(x\frac{k_x}{N_x}+y\frac{k_y}{N_y}+t\frac{k_t}{N_t})} \end{aligned} \quad (92)$$

where the number of frames $N_t = N_x$ and δ_{jk} is the Kronecker delta function with the property that $a_j = \sum_{k=1}^{n} \delta_{jk} a_k$. Thus the 3-dimensional DFT representation of the video sequence is

$$f(x,y,t) = \frac{1}{N_x N_y N_t} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} \sum_{k_t=0}^{N_t-1} F(k_x,k_y,k_t) e^{j2\pi(x\frac{k_x}{N_x}+y\frac{k_y}{N_y}+t\frac{k_t}{N_t})} \quad (93)$$

where

$$F(k_x,k_y,k_t) = \begin{cases} N_t F(k_x,k_y) & \text{if } k_t = k_x \\ 0 & \text{otherwise.} \end{cases} \quad (94)$$

Applying an ideal low pass spatial filter to eliminate higher spatial frequencies in the individual frames, (93) becomes

$$\begin{aligned} \tilde{f}(x,y,t) &= \frac{1}{N_x N_y N_t} \sum_{k_x=0}^{N_1} \sum_{k_y=0}^{N_2} \sum_{k_t=0}^{N_t-1} F(k_x,k_y,k_t) e^{j2\pi(x\frac{k_x}{N_x}+y\frac{k_y}{N_t}+t\frac{k_t}{N_t})} \\ &= \frac{1}{N_x N_y} \sum_{k_x=0}^{N_1} \sum_{k_y=0}^{N_2} F(k_x,k_y) e^{j2\pi(x\frac{k_x}{N_x}+y\frac{k_y}{N_x}+t\frac{k_x}{N_t})} \end{aligned} \quad (95)$$

where N_1 and N_2 are smaller than $N_x - 1$, $N_y - 1$, and $N_t - 1$. Note that (94) was used to reduce the number of summations from three to two in the representation of $\tilde{f}(x,y,t)$ so that the label k_t was replaced with k_x. From (95), it is clear that not only are the higher spatial frequency components corresponding to $k_x > N_1$ and $k_y > N_2$ in the individual frames lost, but so are the higher temporal frequency components $k_t > N_1$ in the video sequence. This is because the linear motion coupled the temporal and spatial information as exhibited in this case by the $\delta_{k_x k_t}$ term in (92) (cf., (94)). This argument also holds for video sequences of an object experiencing an arbitrary constant linear motion. Similar results hold for objects experiencing planar rotations or scaling, although the equations

are somewhat more complicated. This inherent loss of important high frequency temporal information makes low pass filtering an undesirable option for reducing the resolution of a video sequence. Random sampling on the other hand does not suffer from this phenomenon as it does not tend to selectively eliminate temporal frequencies. This motivates a modified version of Chang's algorithm, which is the topic of the next section.

5.2 *Fast Eigendecomposition Algorithm*

The objective here is to determine the first k left singular vectors of X. Using the analysis of the resolution reduction techniques in the previous section, one can now make the appropriate modifications to Chang's algorithm to improve its computational efficiency. Random sampling is used to reduce X in the spatial dimension and then Chang's method is used to reduce it further in the temporal dimension. An overview of the algorithm is presented first with the details following.

1. Generate the Fourier matrix, $F_{(n\times n)}$, and its *real* counterpart, $H_{(n\times n)}$ for $X_{(m\times n)}$.

2. Randomly sample n pixels from each image in X to obtain the $n \times n$ reduced image data matrix X_r.[3]

3. Determine the smallest number p such that

$$\rho(X_r^T, \mathbf{h}_1, \cdots, \mathbf{h}_p) = \frac{\sum_{i=1}^{p} \|X_r \mathbf{h}_i\|_2^2}{\|X_r\|_F^2} > \mu \tag{96}$$

   where μ is the user-specified reconstruction ratio.

4. Compute the thin SVD of $(X_r H_p)_{(n\times p)} = (U_r)_{(n\times p)}(S_r)_{(p\times p)}(V_r)_{(p\times p)}^T$.[4]

5. Repeat Steps 2 through 4 for three more times and concatenate all $S_r V_r^T$ matrices to

[3] The same permutation of n pixels is used over all m images, however, the order of these randomly sampled pixels in the reduced images does not matter.

[4] $X_r H_p$ is readily available after Step 3.

form

$$A^T_{(P\times s)} = \begin{bmatrix} S_{r_1}V^T_{r_1} \\ S_{r_2}V^T_{r_2} \\ S_{r_3}V^T_{r_3} \\ S_{r_4}V^T_{r_4} \end{bmatrix}$$

where s is the maximum of the four values of p and P is the sum of all values of p.[5]

6. Compute the reduced SVD of $A_{(s\times P)} = (U_s)_{(s\times s)}(S_s)_{(s\times P)}(V_s)^T_{(P\times P)}$.

7. Compute $Z_{(n\times s)} = (H_s)_{(n\times s)}(U_s)_{(s\times s)}$ to get an initial estimate of right singular vectors of X.[6]

8. Perform Steps 2 and 3. If $p > s$, perform Step 4 and compute $Z^{\text{new}} = (H_p)(V_r)$. Then update Z using:

```
for i = 1, 2, ..., p
        w = [I − ZZ^T] ẑ_i^new;
        if ||w|| > ε
                Z = [Z, ŵ];
                s = s + 1;
        end
end
```

where I is an $n \times n$ identity matrix and ϵ is some user-specified threshold.

9. Repeat Step 8 until $p \leq s$ for four consecutive times.

10. Compute $\tilde{U}_{(m\times s)} = X_{(m\times n)}Z_{(n\times s)}$ that gives an approximate basis for the left singular vectors of X.

11. Find the orthonormal basis for $\tilde{U}_{(m\times s)}$ using the thin QR decomposition, i.e., $\tilde{U}_{(m\times s)} = \tilde{\tilde{U}}_{(m\times s)}R_{(s\times s)}$.

12. Return $\tilde{\tilde{U}}_{(m\times s)}$.

[5] All $S_r V_r^T$ matrices should be padded with the appropriate number of columns of zeros if necessary, so that each of them has s columns.

[6] H_s consists of the first s columns of H in (78).

13. Optionally, check if $k < s$ by finding $\tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k$ such that $\rho(X, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k) > \mu$.

The above steps will now be explained in more detail. Steps 2 through 4 compute the p value for X_r and the SVD of the $X_r H_p$ matrix. Here it is described why n pixels are selected as the resolution of the downsampled version of an image. Recall that it is always true that $p \leq n$ (typically with $p \ll n$) for any image data matrix [45], therefore more than n pixels in each column in the reduced image data matrix are never needed to preserve the rank of X_r at p. However, if the resolution is selected at a value less than n, one always runs the risk of artificially reducing the rank below p. If n is large, then one may want to incrementally determine an appropriate number of rows for X_r.

Once the SVD of $X_r H_p$ is calculated, $(H_p)(V_r)$ gives the right singular vectors of X_r. These right singular vectors can be considered as a "good" approximation of their high-resolution counterparts [56]. However, different X_r's require different numbers of harmonics to satisfy the user-specified reconstruction ratio, because the random pixels used to create a specific X_r may not accurately represent the temporal properties of the entire X. Hence Steps 2 through 4 are performed four times to improve the probability of accurately representing the high-resolution image data matrix. The number of times to repeat Steps $2-4$ was empirically determined (on average) to optimize computational efficiency.[7]

Step 5 concatenates all $S_r V_r^T$ matrices to form the matrix A whose range will approximately span the dominant right singular vectors of X [56]. The SVD of A is computed in Step 6 to find its range, given by U_s. Thus, $(H_s)(U_s)$ computed in Step 7 can be considered as a good initial estimate of the right singular vectors of X. Note that if V_r's are used instead of $S_r V_r$'s to form A, then Step 7 will result in an unordered estimate of the right singular vectors of X. To obtain an ordered estimate, the right singular vectors in each V_r are scaled by their corresponding singular values before being concatenated in Step 5.

Fig. 20 shows that even after performing Steps 2 and 3 four different times, the maximum p values for Videos 3 and 4 were below the "true" p values. Hence Steps 8 and 9 are performed to check if there is any new information available in additional samplings of X.

[7] Using more than four iterations may be unnecessary and using fewer than four may require more iterations of the more computationally expensive Step 8.

If the new information in any $\hat{\mathbf{z}}_i^{\text{new}}$ is above a threshold, the Z matrix is updated. When no columns are added to the Z matrix for four consecutive times, the algorithm assumes that the final Z matrix provides a "good" basis for the right singular vectors of X.[8] In short, Steps 2 through 9 are performed to find the approximate right singular vectors of X.

Step 10 computes the approximate basis for the left singular vectors of X, while Step 11 computes the corresponding orthonormal basis using the QR decomposition. Step 13 optionally computes the minimum subspace that will satisfy the user-specified reconstruction ratio.

The computational expense of the proposed algorithm is now briefly analyzed. The cost incurred in Step 2, i.e., constructing X_r from X requires n^2 flops, while the estimation of the smallest number p in Step 3 requires $O(n^2p)$ flops. In Step 4, the cost of computing the SVD of the $n \times p$ matrix X_rH_p requires $O(np^2)$ flops. Step 5 performs Steps 2 through 4 four times. In Step 6, the cost of computing the SVD of the $s \times P$ matrix A requires $O(sP^2)$ flops, while finding the initial estimate of the right singular vectors of X in Step 7 requires ns^2 flops. Steps 8 and 9 that check if any new information should be added to Z requires $(n^2 + n^2p + np^2)$ flops and is repeated an unknown, but typically small, number of times. In Step 10, multiplication of X with Z requires ms^2 flops and the QR decomposition of $\tilde{U}$ in Step 11 requires $O(ms^2)$ flops. Finally, determination of the minimum dimension k in Step 13 requires $O(mnk)$ flops. If $s \ll n \ll m$, then the total computation required is $O(ms^2)$, which is essentially the cost of the QR decomposition.

5.3 Experimental Results

The proposed eigendecomposition algorithm was evaluated using all 20 objects in Fig. 2 and all 18 video sequences in Fig. 3. In particular, the algorithm was used to calculate the partial SVD of X for each set, with $\mu = 0.95$ and $\epsilon = 10^{-6}$. Tables 8 and 9 show a breakdown of the average time required for the different steps in the proposed algorithm for objects and video sequences, respectively. Both the tables show that in the typical cases, when $s \ll n$,[9]

[8] The value four was empirically determined to make it highly unlikely that the number of columns in Z is far from the true value of p.

[9] Recall that for rotated objects, $n = 360$, and for video sequences, $n = 150$

Table 8: Time required for the proposed algorithm for rotated objects (all times are in seconds)

			Time required for different steps							
Object	k	s	$2-5$	$6,7$	$8,9$	10	11	Part	13	Total
1	12	13	0.4500	0	0.4510	0.3200	0.3210	1.5420	1.7220	3.2640
2	13	15	0.5010	0	0.4800	0.3410	0.4300	1.7520	1.8230	3.5750
3	27	30	0.8310	0.01	0.8120	0.5100	1.6930	3.8560	3.5550	7.4110
4	7	9	0.3610	0	0.3500	0.2800	0.1510	1.1420	1.0810	2.2230
5	4	5	0.3000	0	0.2810	0.2400	0.0500	0.8710	0.7110	1.5820
6	17	19	0.5910	0.01	0.5810	0.3800	0.6810	2.2430	2.3240	4.5670
7	5	5	0.2900	0	0.2910	0.2400	0.0500	0.8710	0.8310	1.7020
8	6	8	0.3410	0	0.3400	0.2600	0.1210	1.0620	0.9510	2.0130
9	15	16	0.5210	0.01	0.5010	0.3500	0.4910	1.8730	2.0730	3.9460
10	4	9	0.3700	0	0.3510	0.2800	0.1500	1.1510	0.7310	1.8820
11	8	12	0.4110	0	0.4000	0.3110	0.2700	1.3920	1.2120	2.6040
12	6	7	0.3310	0	0.3200	0.2610	0.0900	1.0020	0.9610	1.9630
13	7	8	0.3500	0	0.3510	0.2700	0.1100	1.0810	1.0920	2.1730
14	6	7	0.3200	0	0.3210	0.2600	0.0900	0.9910	0.9620	1.9530
15	5	6	0.3000	0	0.3010	0.2500	0.0700	0.9210	0.8410	1.7620
16	16	18	0.5310	0	0.5110	0.3700	0.6210	2.0330	2.2730	4.3060
17	12	13	0.4410	0.01	0.4400	0.3110	0.3300	1.5320	1.7130	3.2450
18	9	10	0.3900	0	0.3810	0.2900	0.1900	1.2510	1.3420	2.5930
19	10	13	0.4400	0	0.4310	0.3200	0.3210	1.5120	1.4620	2.9740
20	12	14	0.4800	0	0.4710	0.3300	0.3710	1.6520	1.7120	3.3640

the algorithm is computationally very efficient with all the steps sharing the computational expense equally. However, as the value of s increases, the computational time required for Step 11 that computes the orthonormal basis for $\tilde{U}_{(m\times s)}$ using QR decomposition dominates the first ten steps combined. This agrees with the computational expense analysis of our algorithm provided in the previous section. The tables also show that the calculation of the first k eigenimages in Step 13 requires a significant amount of time. Thus a user may prefer to stop after Step 12 and simply use all s eigenimages. The total time for Steps $1-12$ is given in the column labelled "Part". If one indeed needs to know the minimum subspace, then the total time required for the algorithm is given in the "Total" column.

The proposed algorithm was run ten different times for each object as well as video sequence and the mean values for "Time", s, and k were calculated. Tables 10 and 11 summarize the performance of the algorithm, showing k^*, k, p, s, and the computation

Table 9: Time required for the proposed algorithm for video sequences (all times are in seconds)

			Time required for different steps							
Video	k	s	$2-5$	$6,7$	$8,9$	10	11	Part	13	Total
1	15	18	0.0800	0	0.0700	0.6710	1.4420	2.2630	4.2060	6.4690
2	4	7	0.0400	0	0.0500	0.4310	0.2800	0.8010	1.4720	2.2730
3	66	71	0.3310	0.23	0.2300	2.1330	19.3680	22.2920	16.6240	38.9160
4	63	72	0.3100	0.19	0.2200	2.1530	19.8990	22.7730	15.8920	38.6650
5	4	7	0.0500	0	0.0500	0.3810	0.2500	0.7310	1.3920	2.1230
6	1	3	0.0400	0	0.0400	0.3000	0.0700	0.4500	0.6910	1.1410
7	10	13	0.0600	0	0.0500	0.5010	0.7410	1.3520	2.7940	4.1460
8	5	9	0.0500	0	0.0500	0.4210	0.3900	0.9110	1.6230	2.5340
9	39	47	0.1710	0.02	0.1300	0.3100	1.9130	2.5440	2.2030	4.7470
10	10	19	0.0700	0.01	0.0600	0.1610	0.3400	0.6410	0.6410	1.2820
11	4	6	0.0400	0	0.0400	0.0900	0.0410	0.2110	0.3300	0.5410
12	5	16	0.0700	0	0.0600	0.1400	0.2510	0.5210	0.3800	0.9010
13	2	3	0.0400	0	0.0400	0.0700	0.0100	0.1600	0.2210	0.3810
14	8	11	0.0500	0	0.0600	0.3810	0.4900	0.9810	1.9830	2.9640
15	15	21	0.0710	0.01	0.0700	0.5600	1.5630	2.2740	3.3550	5.6290
16	5	7	0.0500	0	0.0400	0.1900	0.1510	0.4310	0.8510	1.2820
17	5	9	0.0500	0	0.0500	0.3100	0.2910	0.7010	1.1310	1.8320
18	1	1	0.0300	0	0.0400	0.0900	0	0.1600	0.2400	0.4000

times for objects and video sequences, respectively. Compared to the direct SVD, the speedup factors with the proposed algorithm for the objects (refer to Table 10) are in the range of $31.60 - 135.11$, depending on the value of s. Because the original image frames for all the objects have the same resolution, the speedup factor here depends only on the value of s. This is evident from the table results, as the minimum and maximum speedup factors are obtained for objects 3 and 5, respectively. The difference between $\rho(X, \hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k^*})$ and $\rho(X, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k)$ for each set was less than 0.13%, with an average of 0.07%, which reveals that $\{\tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k\}$ provides a very good approximate basis for the first k^* eigenimages $\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k^*}\}$. The speedup factors, on the other hand, for the video sequences (refer to Table 11) are in the range of $3.15 - 154.26$. The original image frames for these video sequences do not have the same resolution, hence the speedup factor here depends both on the value of s and the original image resolution. However, one can still observe a strong correlation between the value of s and the speedup factor. Hence, in the typical cases when $s \ll n$, the speedup factors will generally be more than 50.

Tables 10 and 11 also show that the average s and k values obtained with the proposed algorithm are very close to the p and k values obtained with Chang's algorithm for both the data sets, while the "time" entries in both the tables show that the proposed algorithm is computationally more efficient than Chang's algorithm.

The quality of the resulting eigendecomposition was also evaluated using the error measures described in Section 2.4 with the two data sets in Figures 2 and 3. The first object and the first video sequence are again used as representative examples. Fig. 18 and Fig. 19 show the corresponding results. All these results reveal that the subspaces obtained using proposed algorithm are as good as those obtained using Chang's algorithm when a k-dimensional eigenspace is used. In particular, the maximum principal angle and subspace criterion plots show that both the algorithms give nearly the same quality eigenspaces when the dimension of k is used. The energy recovery ratio plots, on the other hand, reveal that both the algorithms give virtually perfect image reconstruction.

Table 10: Comparison of different algorithms for rotated objects (all times are in seconds)

	Proposed algorithm			Chang's algorithm			MATLAB	
Object	Time	s	k	Time	p	k	Time	k^*
1	1.5581	13.1	12.0	7.0210	13	12	118.7350	12
2	1.6695	14.4	13.0	6.9110	13	13	120.5500	13
3	3.7303	29.4	27.0	8.6230	28	27	117.8670	27
4	1.1358	8.7	7.0	6.5900	8	7	117.7370	7
5	0.8713	5.0	4.0	6.4600	5	4	117.7170	4
6	2.3102	19.8	16.5	7.4610	19	17	117.6860	15
7	0.8872	5.4	5.0	6.4800	5	5	118.0170	4
8	1.1248	8.7	5.1	6.5690	8	5	117.9360	5
9	1.9169	16.4	15.0	7.0710	15	15	117.8170	15
10	1.1286	8.5	4.0	6.6200	8	4	118.1570	4
11	1.3299	11.2	8.1	6.6600	9	8	118.1770	8
12	0.9770	6.7	5.9	6.5400	6	6	117.9170	5
13	1.0926	8.1	7.0	6.6000	8	7	117.9770	7
14	0.9890	7.0	6.0	6.6000	7	6	117.9770	6
15	0.8873	5.3	5.0	6.4800	5	5	118.1670	5
16	2.0330	17.8	16.0	7.3010	17	16	117.8360	16
17	1.4910	12.7	12.0	6.8710	12	12	118.1870	11
18	1.2607	10.2	9.0	6.7200	10	9	118.0370	9
19	1.4541	12.2	10.1	6.8010	11	11	118.1470	9
20	1.5801	13.5	12.0	6.8500	12	12	118.0170	12

Table 11: Comparison of different algorithms for video sequences (all times are in seconds)

	Proposed algorithm			Chang's algorithm			MATLAB	
Video	Time	s	k	Time	p	k	Time	k^*
1	2.1351	17.1	15.0	14.6720	15	15	74.1670	15
2	0.7961	6.9	4.0	12.5590	6	4	71.8530	4
3	22.1549	70.5	66.1	57.2870	68	66	72.4240	63
4	22.9549	71.9	63.0	47.1010	65	63	72.4150	60
5	0.7933	7.7	4.0	12.2390	6	4	64.3630	4
6	0.4136	2.3	1.0	11.8080	2	1	63.8020	1
7	1.2656	12.2	10.0	13.1090	11	10	64.9830	9
8	0.7961	7.7	5.0	12.3690	7	5	64.0920	5
9	2.6498	47.9	39.2	6.0990	44	39	15.3820	36
10	0.5748	17.4	10.0	3.0150	16	10	15.2920	9
11	0.2129	5.9	4.0	2.4840	5	4	15.3520	4
12	0.5408	16.7	5.0	2.8340	13	5	15.4020	4
13	0.1623	3.0	2.0	2.4030	3	2	15.3920	2
14	1.0085	11.2	8.0	4.5570	10	8	55.9200	7
15	1.9939	19.1	15.0	6.4100	18	15	55.9410	15
16	0.4757	8.0	5.2	2.5040	7	5	35.2110	5
17	0.6619	8.4	5.0	7.6520	7	5	47.0780	5
18	0.1683	1.0	1.0	4.1560	1	1	23.1130	1

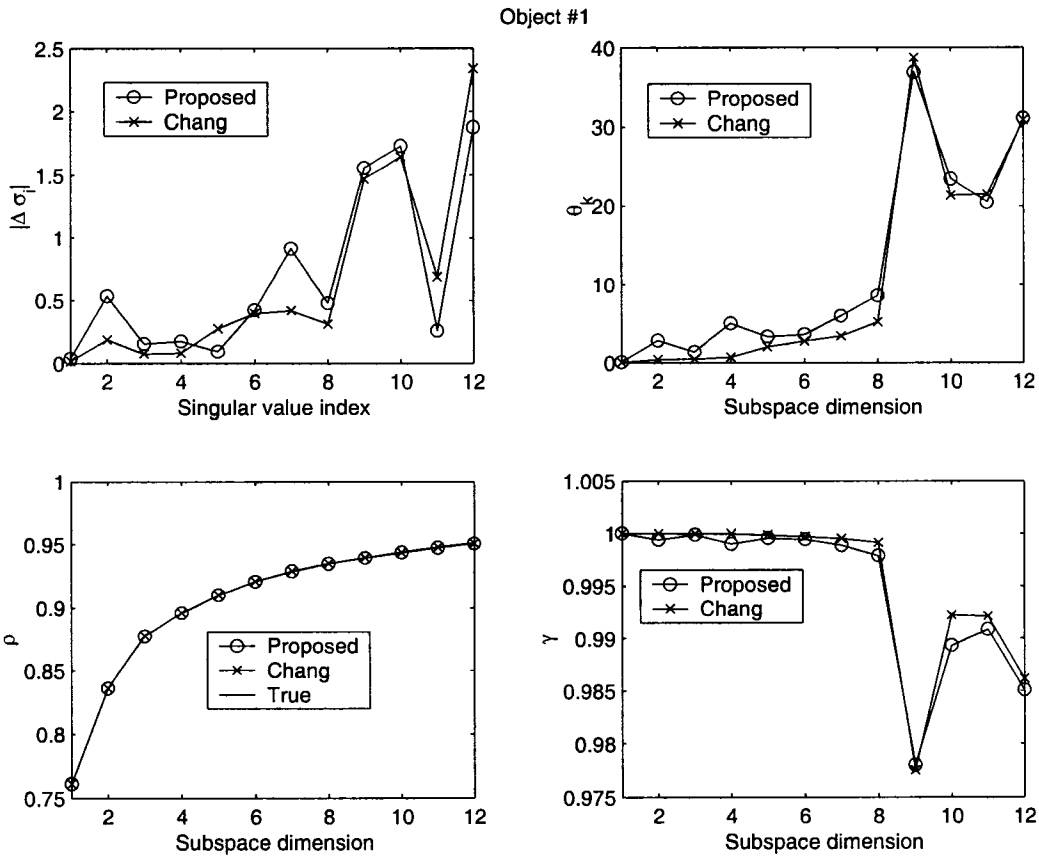


Figure 21: This figure shows a comparison of the approximated SVDs with the true SVD for object #1 from Fig. 2. The plots with $\circ$ and $\times$ give the results of the proposed and Chang's algorithm, respectively. The top left plot shows the difference between the true and approximated singular values. The top right plot shows the maximum principal angles in degrees between the U subspaces, when the subspace dimension is varied from 1 to $k = k^* = 12$. The bottom left plot shows the energy recovery ratio using true and approximated eigenimages, while the remaining plot shows the subspace criterion between the true and approximated eigenimages, when the subspace dimension is varied from 1 to k.

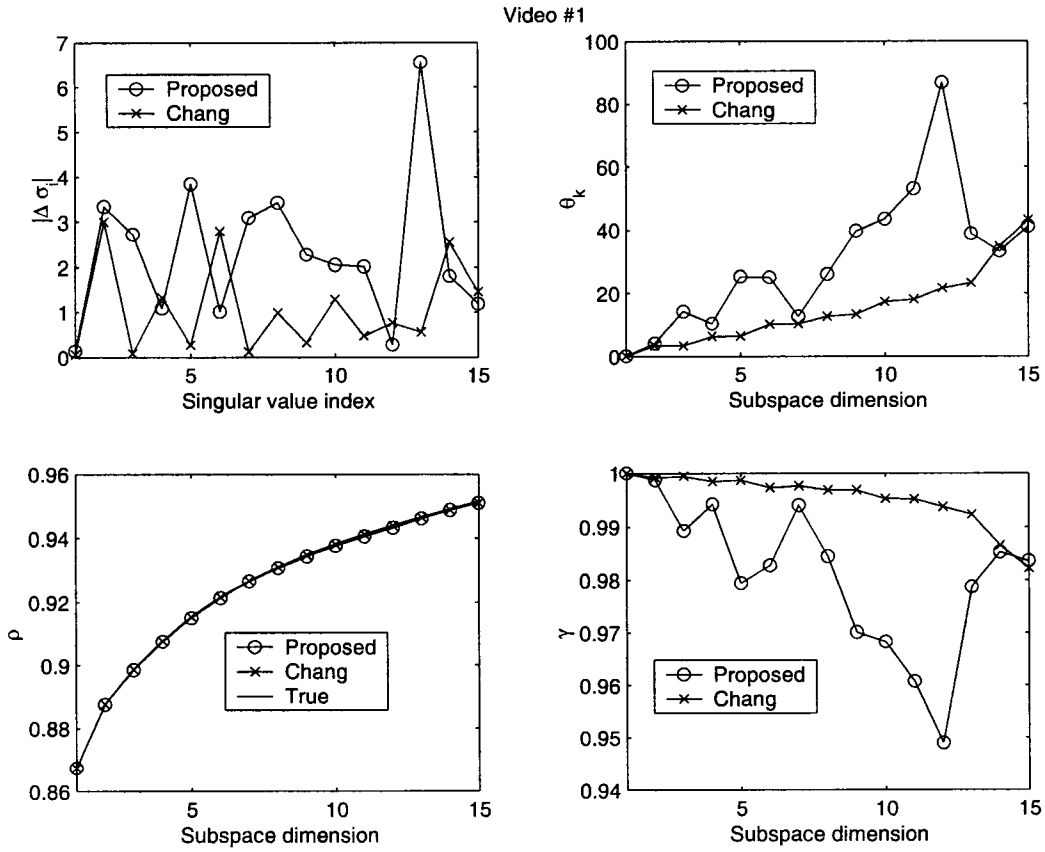


Figure 22: This figure shows a comparison of the approximated SVDs with the true SVD for video #1 from Fig. 3. The plots with $\circ$ and $\times$ give the results of the proposed and Chang's algorithm, respectively. The top left plot shows the difference between the true and approximated singular values. The top right plot shows the maximum principal angles in degrees between the U subspaces, when the subspace dimension is varied from 1 to $k = k^* = 15$. The bottom left plot shows the energy recovery ratio using true and approximated eigenimages, while the remaining plot shows the subspace criterion between the true and approximated eigenimages, when the subspace dimension is varied from 1 to k.

CHAPTER VI

FAST EIGENDECOMPOSITION OF 3D IMAGE SETS

Previous computationally efficient eigendecomposition algorithms considered exploiting the correlation between the successive images by putting the i^{th} image between $(i-1)^{\text{th}}$ and $(i+1)^{\text{th}}$ images; these image data sets will be referred to as one-dimensional (1D) image data sets here. However, certain pattern recognition applications require that different views of an object taken from different three-dimensional (3D) spatial camera locations be considered in the training data set. Because they are characterized by three parameters, such image data sets will be referred to as 3D image data sets. The existing algorithms do not consider the correlation along more than one dimension and hence cannot be applied to such 3D image data sets directly. The goal of this chapter is to extend Chang's algorithm (refer to Chapter 4) to effectively compute the partial SVD of such 3D image data sets.

This chapter is organized as follows. Section 6.1 explains the experimental setup for generation of fully general 3D image data sets, while Section 6.2 explains the frequency representation of these 3D image data sets. Section 6.3 explains how to efficiently compute and use the real 3D DFT of these sets, while Section 6.4 addresses the issue of ordering the frequencies for these 3D image data sets. Finally, the proposed algorithm (extension of Chang's algorithm) is outlined in Section 6.5, while supporting results are analyzed in Section 6.6.

6.1 Generation of 3D Image Sets

Figure 23 illustrates the experimental setup for generating fully general 3D image data sets, in which the correlated images are characterized by three parameters instead of one. In this setup, camera locations are defined in a spherical patch above the object with two consecutive camera locations separated by an equiangular distance in that patch. The

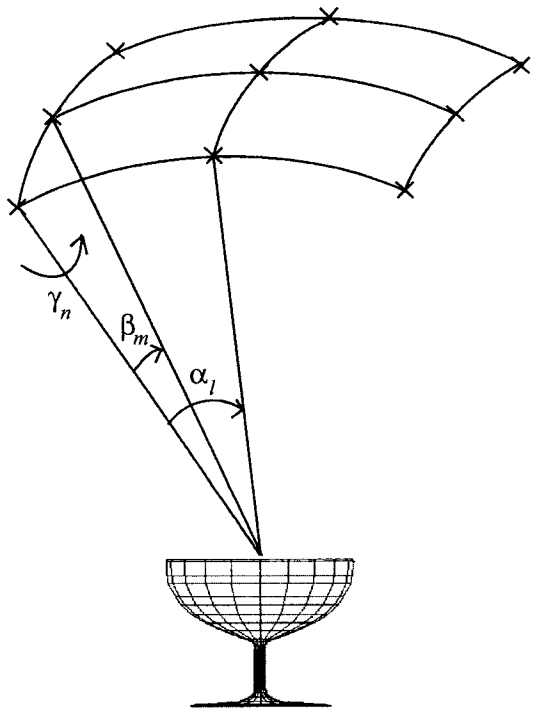


Figure 23: This figure shows the experimental setup for generating 3D image data sets, in which the images are characterized by three parameters, i.e., α_l, β_m, and γ_n. The crosses (x) denote the simulated camera locations that are placed in the spherical patch above the object. The range of the parameters α_l and β_m can be varied with respect to nadir (-40^o to 40^o was typically used for the experimental results presented here), whereas the range of γ_n is typically unrestricted.

range of these camera locations is characterized by two parameters, i.e., α_l and β_m, while the third parameter γ_n characterizes image plane rotation to capture different views of the object in equal increments. In practice, the required images can be captured using a video camera attached to a robot end-effector. The robot movement can be controlled to position the camera in one of the specified locations in the spherical patch and then the robot end-effector can be rotated to rotate the image plane of the camera for capturing different orientations of the object from the same location.

The images of an object captured using the experimental setup in Fig. 23 can be row-scanned and put into one four-dimensional (4D) image array $X_{m\times L\times M\times N}$ in such a way

that the entries along the first dimension of $X(:,l,m,n)$[1] correspond to the row-scanned image of an object taken from camera location (l,m) at the image plane rotation n, where $1 \leq l \leq L$, $1 \leq m \leq M$, and $1 \leq n \leq N$. The entries of X can be rearranged to obtain the following 3D image data matrix:

$$\breve{X} = [\mathbf{x}_{111}, \cdots, \mathbf{x}_{L11}, \mathbf{x}_{121}, \cdots \mathbf{x}_{L21}, \cdots, \mathbf{x}_{LM1}, \mathbf{x}_{112}, \cdots, \mathbf{x}_{LM2}, \cdots, \mathbf{x}_{LMN}] \tag{97}$$

where an image vector $\mathbf{x}_{lmn}$ again corresponds to the row-scanned image of an object taken from camera location (l,m) at image plane rotation n. The next section explains the frequency analysis of these 3D image data matrices.

6.2 Frequency Analysis of 3D Image Sets

Consider a three-dimensional signal $g(x,y,z)$ containing L, M, and N samples in the x, y, and z dimensions, respectively. The corresponding frequency representation using the 3D DFT can be given by

$$G(\alpha,\beta,\gamma) = \frac{1}{\sqrt{LMN}} \sum_{x=0}^{L-1} \sum_{y=0}^{M-1} \sum_{z=0}^{N-1} g(x,y,z) \omega_L^{\alpha x} \omega_M^{\beta y} \omega_N^{\gamma z} \tag{98}$$

where $0 \leq \alpha \leq L-1$, $0 \leq \beta \leq M-1$, and $0 \leq \gamma \leq N-1$, while $\omega_L = e^{-j2\pi/L}$, $\omega_M = e^{-j2\pi/M}$, and $\omega_N = e^{-j2\pi/N}$. Thus, similar to 1D image data sets, an orthonormal basis for the image data matrix $\breve{X}$ can be generated using the basis for the 3D DFT. In particular, the following three-dimensional array represents one 3D frequency:

$$F_{\alpha\beta\gamma}(x,y,z) = \frac{1}{\sqrt{LMN}} \omega_L^{\alpha x} \omega_M^{\beta y} \omega_N^{\gamma z} \tag{99}$$

where α, β, and γ denote the desired frequency components in three dimensions with $0 \leq x \leq L-1$, $0 \leq y \leq M-1$, and $0 \leq z \leq N-1$. All $F_{\alpha\beta\gamma}$ arrays can be lexicographically ordered (similar to the ordering of $\mathbf{x}_{lmn}$ vectors in $\breve{X}$) into their respective column vectors (denoted $\mathbf{f}_{\alpha\beta\gamma}$) so that the corresponding $(L \times M \times N) \times (L \times M \times N)$ "3D" Fourier matrix is given by

$$\breve{F} = [\mathbf{f}_{000} | \cdots | \mathbf{f}_{\alpha\beta\gamma} | \cdots | \mathbf{f}_{(L-1)(M-1)(N-1)}]. \tag{100}$$

[1] A colon (:) in an array argument is used here to specify all entries in the corresponding dimension of that array.

Note that the columns of $\breve{F}$ give the 3D DFT basis for complex matrices. However, for real images in 3D image data sets, the matrix $\breve{X}$ in (97) will contain all real values and hence, similar to the basis given by columns of H in (78) for 1D image data sets, $\breve{X}$ will have a real basis. To find this real basis, Euler's formula ($e^{-jx} = \cos x - j \sin x$) can be used to rewrite (99) as follows:

$$\begin{aligned} F_{\alpha\beta\gamma}(x,y,z) = &\frac{1}{\sqrt{LMN}}[(c_{\alpha x}c_{\beta y}c_{\gamma z} - c_{\alpha x}s_{\beta y}s_{\gamma z} - s_{\alpha x}c_{\beta y}s_{\gamma z} - s_{\alpha x}s_{\beta y}c_{\gamma z}) \\ &- j(c_{\alpha x}c_{\beta y}s_{\gamma z} + c_{\alpha x}s_{\beta y}c_{\gamma z} + s_{\alpha x}c_{\beta y}c_{\gamma z} - s_{\alpha x}s_{\beta y}s_{\gamma z})] \end{aligned} \tag{101}$$

where $c_{\alpha x} = \cos(\frac{2\pi\alpha x}{L})$, $c_{\beta y} = \cos(\frac{2\pi\beta y}{M})$, and $c_{\gamma z} = \cos(\frac{2\pi\gamma z}{N})$, while $s_{\alpha x}$, $s_{\beta y}$, and $s_{\gamma z}$ are the corresponding sine components. Note that there will be 1, 2, 4, and 8 different sine-cosine combinations for $F_{\alpha\beta\gamma}$ with 0, 1, 2, and 3 total number of non-zero α, β, and γ frequencies, respectively. Let r denote this number of non-zero α, β, and γ frequencies. If these sine-cosine combinations are lexicographically ordered and are scaled by $\sqrt{2^r}$ to give orthonormal columns of $H_{\alpha\beta\gamma}$, then the *real* 3D Fourier matrix of size $(L \times M \times N) \times (L \times M \times N)$ can be given by

$$\breve{H} = \begin{bmatrix} \mathbf{f}_{000} & \cdots & H_{\alpha\beta\gamma} & \cdots \end{bmatrix} \tag{102}$$

where the first column, $\mathbf{f}_{000}$, of $\breve{H}$ refers to the zero frequency component corresponding to $r = 0$. Note that if any of the three dimensions, e.g. L, is even, then only the cosine (real) component of the corresponding maximum "real" frequency, i.e., $\frac{L}{2}+1$, is considered, otherwise both cosine (real) and sine (imaginary) components of $\frac{L+1}{2}$ are considered while generating the corresponding orthonormal columns in $\breve{H}$.

The resulting matrix $\breve{H}$, which is generated for a given $\breve{X}$, can be used to extend Chang's algorithm to compute the approximate SVD of $\breve{X}$. In particular, the row space of $\breve{X}$ can be projected to the first few columns of $\breve{H}$ and the SVD of $\breve{X}\breve{H}_p$ can be used to approximate the SVD of $\breve{X}$, where $\breve{H}_p$ denotes the matrix containing the first p columns of $\breve{H}$. The computation of $\breve{X}\breve{H}$ can be performed efficiently using DFT techniques. However, the implementation and use of a real 3D basis in $\breve{H}$ is not as trivial as in the 1D case and it is discussed in detail in the next section.

6.3 *Efficient Computation of $\breve{X}\breve{H}$ Using FFT Techniques*

This section explains how to compute $\breve{X}\breve{H}$ efficiently using the FFT of the original four-dimensional (4D) image array $X_{m\times L\times M\times N}$. Before getting into actual implementation details, the frequency properties of one pixel varying along three dimensions are analyzed here. In particular, consider the 3D DFT representation of a three-dimensional signal $g(x,y,z)$ given by (98), which can be rewritten as

$$G(\alpha,\beta,\gamma)=\frac{1}{\sqrt{LMN}}\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}\left[\sum_{z=0}^{N-1}g(x,y,z)(c_{\gamma z}-js_{\gamma z})\right](c_{\beta y}-js_{\beta y})(c_{\alpha x}-js_{\alpha x}). \quad (103)$$

Now let

$$\begin{aligned} R_z(g) &= \sum_{z=0}^{N-1}g(x,y,z)c_{\gamma z}, \\ I_z(g) &= \sum_{z=0}^{N-1}g(x,y,z)s_{\gamma z}. \end{aligned} \quad (104)$$

Then the 3D DFT $G(\alpha,\beta,\gamma)$ becomes

$$G(\alpha,\beta,\gamma)=\frac{1}{\sqrt{LMN}}\sum_{x=0}^{L-1}\left[\sum_{y=0}^{M-1}(R_z(g)-jI_z(g))(c_{\beta y}-js_{\beta y})\right](c_{\alpha x}-js_{\alpha x}). \quad (105)$$

With

$$\begin{aligned} R_y(R_z(g)) &= \sum_{y=0}^{M-1}R_z(g)c_{\beta y}, \\ R_y(I_z(g)) &= \sum_{y=0}^{M-1}I_z(g)c_{\beta y}, \\ I_y(R_z(g)) &= \sum_{y=0}^{M-1}R_z(g)s_{\beta y}, \\ I_y(I_z(g)) &= \sum_{y=0}^{M-1}I_z(g)s_{\beta y}, \end{aligned} \quad (106)$$

$G(\alpha,\beta,\gamma)$ can be rewritten as

$$G(\alpha,\beta,\gamma)=\frac{1}{\sqrt{LMN}}\sum_{x=0}^{L-1}\left(R_y(R_z(g))-jR_y(I_z(g))-jI_y(R_z(g))-I_y(I_z(g))\right)(c_{\alpha x}-js_{\alpha x}). \quad (107)$$

Finally, with

$$
\begin{aligned}
R_x(R_y(R_z(g))) &= \sum_{x=0}^{L-1} R_y(R_z(g))c_{\alpha x}, \\
R_x(R_y(I_z(g))) &= \sum_{x=0}^{L-1} R_y(I_z(g))c_{\alpha x}, \\
R_x(I_y(R_z(g))) &= \sum_{x=0}^{L-1} I_y(R_z(g))c_{\alpha x}, \\
R_x(I_y(I_z(g))) &= \sum_{x=0}^{L-1} I_y(I_z(g))c_{\alpha x}, \\
I_x(R_y(R_z(g))) &= \sum_{x=0}^{L-1} R_y(R_z(g))s_{\alpha x}, \\
I_x(R_y(I_z(g))) &= \sum_{x=0}^{L-1} R_y(I_z(g))s_{\alpha x}, \\
I_x(I_y(R_z(g))) &= \sum_{x=0}^{L-1} I_y(R_z(g))s_{\alpha x}, \\
I_x(I_y(I_z(g))) &= \sum_{x=0}^{L-1} I_y(I_z(g))s_{\alpha x},
\end{aligned} \tag{108}
$$

$$
\begin{aligned}
G(\alpha,\beta,\gamma) = &\frac{1}{\sqrt{LMN}}(R_x(R_y(R_z(g))) - jR_x(R_y(I_z(g))) - jR_x(I_y(R_z(g))) - R_x(I_y(I_z(g))) \\
&- jI_x(R_y(R_z(g))) - I_x(R_y(I_z(g))) - I_x(I_y(R_z(g))) + jI_x(I_y(I_z(g)))).
\end{aligned} \tag{109}
$$

Note that for a given frequency combination of (α, β, γ), $R_x(\cdot)$, $R_y(\cdot)$, and $R_z(\cdot)$ denote real parts of the DFT of $g(x, y, z)$, computed along x, y, and z dimension, respectively, while $I_x(\cdot)$, $I_y(\cdot)$, and $I_z(\cdot)$ denote the corresponding imaginary parts. This frequency analysis agrees with the fact that there can be a maximum of $2^3 = 8$ sine-cosine combinations for a given frequency combination.

Now consider the four-dimensional (4D) image array $X_{m\times L\times M\times N}$, from which the image data matrix $\breve{X}$ is generated. Recall that $X(:, l, m, n)$ corresponds to the row-scanned image of an object taken from camera location (l, m) at the image plane rotation n, where $1 \leq l \leq L$, $1 \leq m \leq M$, and $1 \leq n \leq N$. Due to the above frequency analysis of $g(x, y, z)$, multiplication of X and its real basis can be computed using fast Fourier transform (FFT)

techniques to realize the following eight 4D arrays:

$$\begin{aligned}
XF_1 &= \Re(\mathrm{fft}_l(\Re(\mathrm{fft}_m(\Re(\mathrm{fft}_n(X)))))), \\
XF_2 &= \Im(\mathrm{fft}_l(\Re(\mathrm{fft}_m(\Re(\mathrm{fft}_n(X)))))), \\
XF_3 &= \Re(\mathrm{fft}_l(\Im(\mathrm{fft}_m(\Re(\mathrm{fft}_n(X)))))), \\
XF_4 &= \Im(\mathrm{fft}_l(\Im(\mathrm{fft}_m(\Re(\mathrm{fft}_n(X)))))), \\
XF_5 &= \Re(\mathrm{fft}_l(\Re(\mathrm{fft}_m(\Im(\mathrm{fft}_n(X)))))), \\
XF_6 &= \Im(\mathrm{fft}_l(\Re(\mathrm{fft}_m(\Im(\mathrm{fft}_n(X)))))), \\
XF_7 &= \Re(\mathrm{fft}_l(\Im(\mathrm{fft}_m(\Im(\mathrm{fft}_n(X)))))), \\
XF_8 &= \Im(\mathrm{fft}_l(\Im(\mathrm{fft}_m(\Im(\mathrm{fft}_n(X))))))
\end{aligned} \tag{110}$$

where fft_l, fft_m, and fft_n denote the FFT of a 4D array computed along the α_l, β_m, and γ_n dimension, respectively.

Notice that the arrays XF_1 through XF_8 will each have $m \times L \times M \times N$ elements (i.e., the same size as that of X), which suggests that these "multiplication" arrays have a significant amount of duplicated information. In particular, consider the FFT computation of X along its γ_n dimension. The corresponding Fourier basis vectors will come in complex conjugate pairs.[2] However, because X contains all real entries, the real and imaginary components of these complex basis vectors will form the corresponding "real" basis, which will result in two sets of duplicate basis vectors. Note that only one of these two sets is necessary and sufficient to proceed with the FFT computation along the β_m dimension. In short, the complete computation of XF_1 through XF_8 using (110) results in a significant amount of wasted time in processing unnecessary data. Thus the arrays XF_1 through XF_8 for X (with odd L, M, and N) can be computed efficiently using the following steps:

1. Compute the FFT of $X_{m \times L \times M \times N}$ along the γ_n dimension to obtain a 4D array $F_n = \mathrm{fft}_n(X)$, which will be of the same size as X.

[2]Recall that if the dimension, e.g. N, is odd, there will be one real basis vector and $\frac{N-1}{2}$ complex conjugate pairs of basis vectors, while if N is even, there will be two real basis vectors and $\frac{N}{2} - 1$ complex conjugate pairs of basis vectors.

2. Use the first $\frac{N+1}{2}$ entries along the γ_n dimension of F_n to compute

$$\begin{aligned} F_{m1} &= \text{fft}_m(\Re(F_n(:,:,:,1:\frac{N+1}{2}))) \\ F_{m2} &= \text{fft}_m(\Im(F_n(:,:,:,1:\frac{N+1}{2}))) \end{aligned} \tag{111}$$

such that F_{m1} and F_{m2} will both have $m \times L \times M \times \frac{N+1}{2}$ elements.[3]

3. Use the first $\frac{M+1}{2}$ entries along the β_m dimension of F_{m1} and F_{m2} to compute

$$\begin{aligned} F_{l1} &= \text{fft}_l(\Re(F_{m1}(:,:,1:\frac{M+1}{2},:))) \\ F_{l2} &= \text{fft}_l(\Im(F_{m1}(:,:,1:\frac{M+1}{2},:))) \\ F_{l3} &= \text{fft}_l(\Re(F_{m2}(:,:,1:\frac{M+1}{2},:))) \\ F_{l4} &= \text{fft}_l(\Im(F_{m2}(:,:,1:\frac{M+1}{2},:))) \end{aligned} \tag{112}$$

such that F_{l1} through F_{l4} will each have $m \times L \times \frac{M+1}{2} \times \frac{N+1}{2}$ elements.

4. Finally, use the first $\frac{L+1}{2}$ entries along the α_l dimension of F_{l1} through F_{l4} to obtain

$$\begin{aligned} XF_1 &= \Re(F_{l1}(:,1:\frac{L+1}{2},:,:)) \\ XF_2 &= \Im(F_{l1}(:,1:\frac{L+1}{2},:,:)) \\ XF_3 &= \Re(F_{l2}(:,1:\frac{L+1}{2},:,:)) \\ XF_4 &= \Im(F_{l2}(:,1:\frac{L+1}{2},:,:)) \\ XF_5 &= \Re(F_{l3}(:,1:\frac{L+1}{2},:,:)) \\ XF_6 &= \Im(F_{l3}(:,1:\frac{L+1}{2},:,:)) \\ XF_7 &= \Re(F_{l4}(:,1:\frac{L+1}{2},:,:)) \\ XF_8 &= \Im(F_{l4}(:,1:\frac{L+1}{2},:,:)) \end{aligned} \tag{113}$$

such that the resulting 4D arrays will each have $m \times \frac{L+1}{2} \times \frac{M+1}{2} \times \frac{N+1}{2}$ elements. The columns in XF_i arrays that correspond to zero real frequency in the corresponding dimensions are discarded before further processing.

[3]The notation $(1:i)$ here refers to the first i entries in the corresponding dimension of an array.

The above XF_1 through XF_8 arrays can be rearranged into their respective 2D matrices, denoted XH_1 to XH_8. Without loss of generality, for each frequency combination, the corresponding columns in XH_i matrices can be given equal importance. However, the relative importance of one frequency combination with any other is not as trivial as in the 1D case and hence this issue needs to be addressed before combining the columns of the XH_i matrices to form the final XH matrix (which is essentially a multiplication of $\breve{X}$ in (97) and $\breve{H}$ in (102)) of size $m \times LMN$. This ordering of the columns of $\breve{X}\breve{H}$ in terms of their "importance" is addressed in the next section before extending Chang's algorithm to general 3D image data sets.

6.4 Proposed Ordering of 3D Frequency Components

This section explains the heuristics behind ordering different 3D frequency components in terms of their energy recovery ability for a given 3D image data set. For this study, several image data sets were generated by ray-tracing different objects as per the specifications of the experimental setup given in Fig. 23. Figure 24 shows eight such artificial objects that were considered in this study. For each object, a total of $L \times M \times N$ images were ray-traced and the corresponding 3D image data matrix, $\breve{X}$, of size $m \times LMN$ was generated, where $m = 128 \times 128 = 2^{14}$. Recall that LM denotes the total number of camera locations above the object, while N denotes the number of images that are captured (after rotating the image plane of the camera) at each camera location.

To propose a "good" ordering of 3D frequencies based on the given specifications of 3D image data sets, the first object in Fig. 24 is used as a representative example here. In particular, two ray-traced 3D image data sets of this object are evaluated in detail. These two image data sets have the same number of images with $L = M = N = 9$ and the parameters α_l and β_m in both these image data sets are allowed to span 80 degrees each. More specifically, the camera locations along the parameters α_l and β_m are placed from $-40°$ to $40°$ with $10°$ separation between the two consecutive camera locations along both α_l and β_m. The only difference in these two image data sets is the range of the parameter γ_n. In one image data set, it is allowed to span only 80 degrees (image plane rotation from

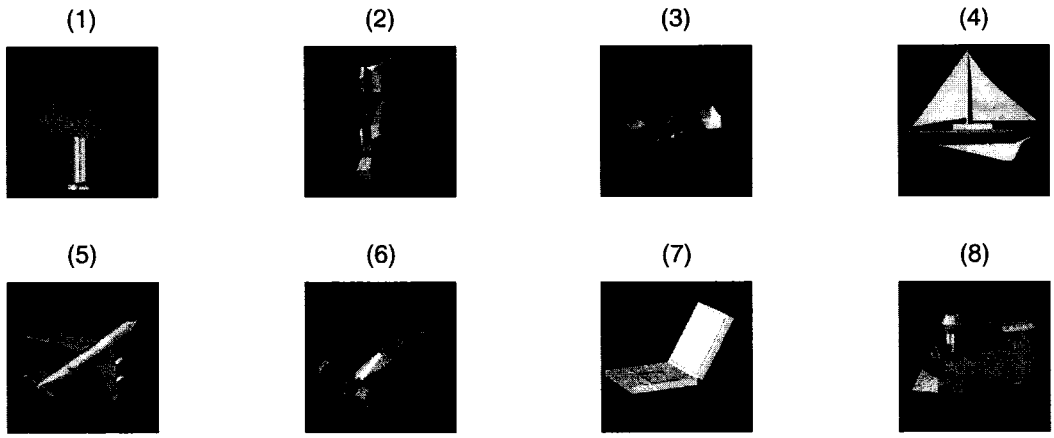


Figure 24: This figure shows eight artificial (ray-traced) objects that are used in this study. Each image of the object is of size 128×128, resulting in an image data matrix $\breve{X}$ of size $2^{14} \times LMN$ for each object.

0° to 80° in 10° increments), while in the other image data set, it spans a full 360 degrees (image plane rotation from 0° to 320° in 40° increments). Note that the first image data set has the same range for all three parameters with the same angular separation in the three parameters between images and hence it will be referred to as the "homogeneously sampled" image data set. On the other hand, the second image data set has a different range and a different angular separation in the three parameters between images. Hence, it will be referred to as the "non-homogeneously sampled" image data set.

Clearly, the above two example image data sets do not represent all possible combinations of resolutions in α_l, β_m, and γ_n parameters. Table 12 shows a much broader range of image data sets with different combinations of resolutions. The detailed analysis of the frequency properties of the homogeneously sampled and non-homogeneously sampled image data sets along with the general observations for the image data sets in Table 12 will be used here to propose frequency ordering for 3D image data sets based on their α_l, β_m, and γ_n parameter specifications. Note that the entries in Table 12 indicate the number of images in the image data set in $L \times M \times N$ format. The entries in the column "Same range (80,80,80), Different separation" indicate that the corresponding image data sets have a 80° range for the α_l, β_m, and γ_n parameters, but different angular separation between images

to accommodate the required number of images along all the parameters. Similarly, the entries in the column "Different range, Same separation (10,10,10)" indicate that the angular separation between images in all three dimensions is 10 degrees. Finally, the entries in the column "Different range, Different separation" refer to the cases where the images cover different ranges in all three parameters and have different equiangular separations between them.

Table 12: Image data sets with different specifications

	Same range (90,90,90) Different separation	Different range Same separation (10,10,10)	Different range Different separation
			Range - 90,90,360
Variation in γ_n	$9 \times 9 \times 36$	$9 \times 9 \times 36$	$9 \times 9 \times 36$
	$9 \times 9 \times 18$	$9 \times 9 \times 18$	$9 \times 9 \times 18$
	$9 \times 9 \times 9$	$9 \times 9 \times 9$	$9 \times 9 \times 9$
	$9 \times 9 \times 3$	$9 \times 9 \times 3$	$9 \times 9 \times 3$
	$9 \times 9 \times 2$	$9 \times 9 \times 2$	$9 \times 9 \times 2$
			Range - 90,360,90
Variation in β_m	$9 \times 36 \times 9$	$9 \times 36 \times 9$	$9 \times 36 \times 9$
	$9 \times 18 \times 9$	$9 \times 18 \times 9$	$9 \times 18 \times 9$
	$9 \times 9 \times 9$	$9 \times 9 \times 9$	$9 \times 9 \times 9$
	$9 \times 3 \times 9$	$9 \times 3 \times 9$	$9 \times 3 \times 9$
	$9 \times 2 \times 9$	$9 \times 2 \times 9$	$9 \times 2 \times 9$
			Range - 360,90,90
Variation in α_l	$36 \times 9 \times 9$	$36 \times 9 \times 9$	$36 \times 9 \times 9$
	$18 \times 9 \times 9$	$18 \times 9 \times 9$	$18 \times 9 \times 9$
	$9 \times 9 \times 9$	$9 \times 9 \times 9$	$9 \times 9 \times 9$
	$3 \times 9 \times 9$	$3 \times 9 \times 9$	$3 \times 9 \times 9$
	$2 \times 9 \times 9$	$2 \times 9 \times 9$	$2 \times 9 \times 9$

Consider a three-dimensional ($L \times M \times N$) array, which has N two-dimensional slices consisting of LM entries each. Let the i^{th} slice correspond to the i^{th} value of γ_n. In particular, the entries in the first slice denote images corresponding to the minimum value of γ_n, while the entries in the last slice denote images corresponding to the maximum value of γ_n. With this terminology, Fig. 25 shows the ray-traced images in the first and the last slice of the homogeneously sampled image data set, while Fig. 26 shows the images in the first and the last slice of the non-homogeneously sampled image data set. Because the minimum value of γ_n for both these image data sets is 0°, the first slice of the images in

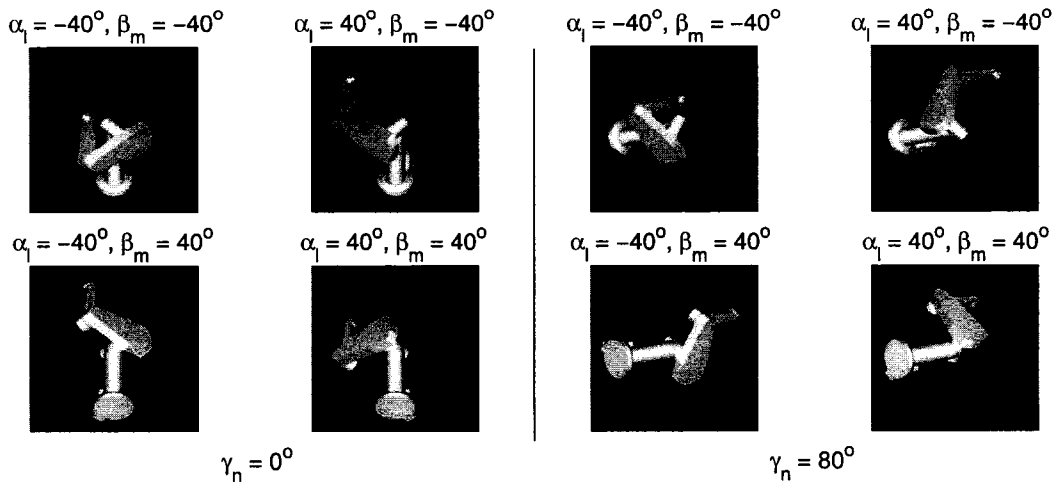


Figure 25: This figure shows ray-traced images for the homogeneously sampled image data set for object 1 in Fig. 24 ($L = M = N = 9$). The left half shows images at the camera locations placed at the extreme corners in the first slice of images (corresponding to $\gamma_n = 0^o$), while the right half shows the images at the same camera locations in the last slice of images (corresponding to $\gamma_n = 80^o$).

these sets will be identical, which is shown in Fig. 27.

Now consider the variation of images along the three parameters for the homogeneously sampled image data set. Images in the middle row and the middle column of Fig. 27 are used here as representative examples for evaluating the variation of images in the parameters α_l and β_m, respectively. The image in the center (corresponding to $\alpha_l = \beta_m = \gamma_n = 0^o$) of Fig. 27 is planar rotated from 0^o to 80^o to obtain nine images in 10^o increments and the resulting image set is used as a representative example for evaluating the variation of images in the parameter γ_n. Figure 28 shows the entries of the first nine right singular vectors of the homogeneously sampled image data set corresponding to the variation of images in the three parameters. Note that the image data set under consideration has $LMN = 729$ images in total. Hence, the corresponding right singular vectors will each have dimension of 729. However, only nine of those 729 entries can be used to monitor the variation of images along one parameter while keeping the other two parameters constant.

Similar to 1D image data sets, Fig. 28 shows that the right singular vector entries corresponding to variation of images along each parameter are well-approximated by sinusoids

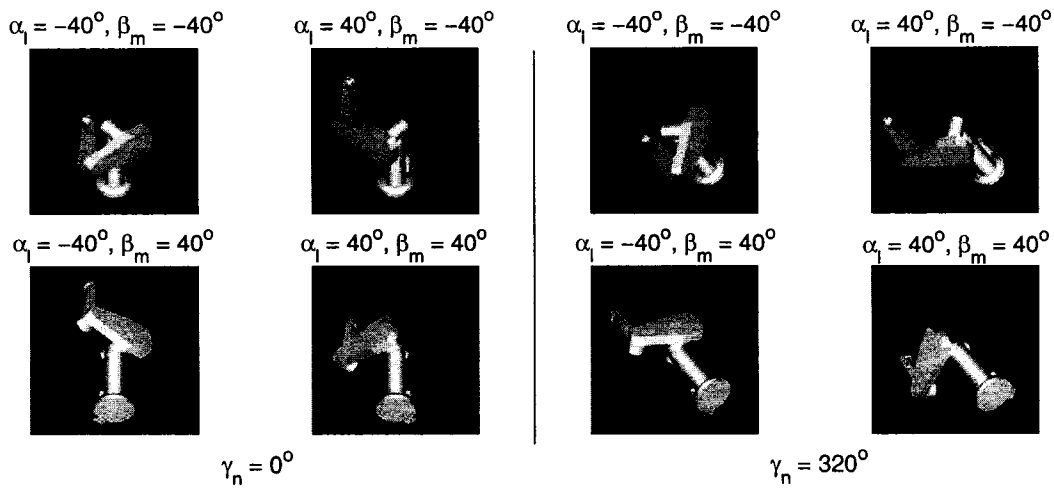


Figure 26: This figure shows ray-traced images for the non-homogeneously sampled image data set for object 1 in Fig. 24 ($L = M = N = 9$). The left half shows images at the camera locations placed at the extreme corners in the first slice of images (corresponding to $\gamma_n = 0^o$), while the right half shows the images at the same camera locations in the last slice of images (corresponding to $\gamma_n = 320^o$).

of increasing frequencies starting from zero frequencies. This suggests that the frequency combinations in $\breve{H}$ for such image data sets can be ordered by the increasing sum of their frequencies, i.e., $(\alpha + \beta + \gamma)$, from 0 to $\frac{L+M+N+6}{2}$.[4] Note that this is an intuitive extension of the 1D image data set case, because the parameters α_l, β_m, and γ_n in the homogeneously sampled image data set span the same range.

Even if it is decided to order the columns in $\breve{H}$ with increasing sum of frequencies in three dimensions, there will be many frequency combinations that will have the same sum. For example, $\alpha + \beta + \gamma = 2$ will result in six different frequency combinations, i.e., $(1,1,0)$, $(1,0,1)$, $(0,1,1)$, $(2,0,0)$, $(0,2,0)$, and $(0,0,2)$. (In general, $\alpha + \beta + \gamma = n$ will result in $\binom{n+2}{2}$ frequency combinations.) Therefore, it is desirable to have an effective ordering within the group of frequency combinations with the same sum of frequencies. Consider the right singular vector entries in Fig. 28 again. It shows that γ frequencies vary differently as opposed to α and β frequencies. This is due to the fact that the parameter γ_n is fundamentally different when compared to the other two parameters and should be treated

[4]Here it is assumed that L, M, and N are all even numbers.

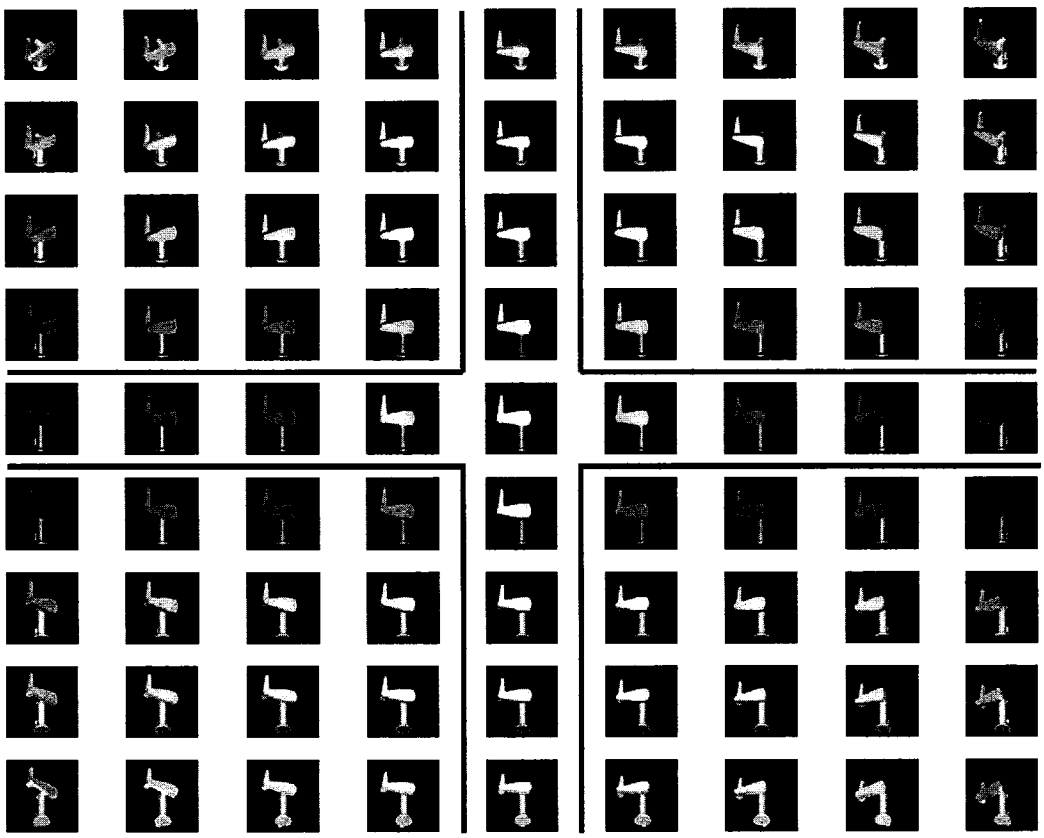


Figure 27: This figure shows the images in the first slice (corresponding to $\gamma_n = 0^\circ$) of both homogeneously sampled (refer to Fig. 25) and non-homogeneously sampled (refer to Fig. 26) image data sets. Within these images, the parameter α_l varies from left to right from -40° to 40° in 10° increments, while the parameter β_m varies from top to bottom from -40° to 40° in 10° increments. In particular, the images in the middle column (confined by the vertical solid lines) of this figure correspond to $\alpha_l = 0^\circ$, while the images in the middle row (confined by the horizontal solid lines) of this figure correspond to $\beta_l = 0^\circ$.

differently. In particular, variation of images along γ_n requires relatively higher frequencies than the variation along the other two parameters. A potential explanation for this is due to the fact that the images varying along the parameter γ_n are planar rotations of each other. Hence the right singular vectors of the corresponding 1D image data matrices would be pure sinusoids with increasing frequencies (due to the properties of circulant matrices [45]). On the other hand, the images varying along the parameters α_l and β_m are not planar rotations and hence the right singular vectors of the corresponding 1D image data matrices

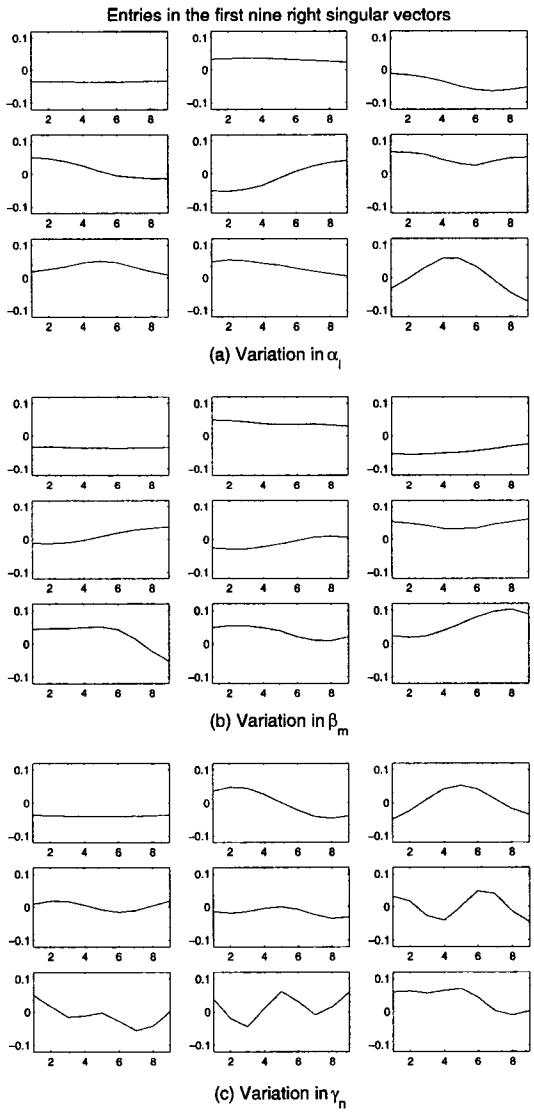


Figure 28: The plots in this figure show entries of the first nine right singular vectors of the homogeneously sampled image data set shown in Fig. 25. In particular, Part (a) shows entries of the first nine right singular vectors corresponding to variation of nine images along the α_l parameter with $\beta_m = \gamma_n = 0^o$. (The corresponding images are shown in the middle row of Fig. 27.) Part (b) shows entries of the same right singular vectors corresponding to variation of nine images along the β_m parameter with $\alpha_l = \gamma_n = 0^o$ (refer to the middle column of Fig. 27), while part (c) shows entries of the same nine right singular vectors corresponding to variation of nine images along the γ_n parameter with $\alpha_l = \beta_m = 0^o$. (The image in the center of Fig. 27 is planar rotated from 0^o to 80^o in 10^o increments to obtain the corresponding images.) Note that only nine (out of a total of $LMN = 729$) right singular vector entries are plotted in each plot here. In each part, the right singular vectors are numbered from 1 through 9 from left to right and top to bottom.

will typically not be pure sinusoids. A simple way of analyzing this is that the images varying along γ_n contain the "same" information in all of them, while the images varying along the other two parameters contain slightly different information in consecutive images. Therefore higher γ frequencies are required as compared to α and β frequencies (within the group of frequency combinations having the same sum of frequency components) to achieve the same level of "importance." Now refer to the first column of Table 12, which lists the image data sets with the same range (80 degrees) for all three parameters. It was observed that regardless of how many images are used along each parameter, the above frequency ordering heuristic for the homogeneously sampled image data set performs well.

The above observations motivate ordering the frequency combinations for the homogeneously sampled (or same-range) image data sets as follows:

1. Group frequency combinations in increasing order of the frequency sum $M_1 = \alpha+\beta+\gamma$.

2. Within a group of frequencies with the same frequency sum M_1, order these combinations in increasing value of $M_2 = \alpha + \beta$.

3. Within a subgroup of frequencies with equal values of M_2, order the combinations based on $M_3 = \alpha - \beta$ using the following ordering on the set of integers:

$$0, 1, -1, 2, -2, 3, -3, \cdots$$

This gives more importance to the combinations with lower individual α and β frequencies over combinations with higher α and β frequencies. Alternatively, one can obtain the same ordering with $M_3' = \max(\alpha, \beta)$ with a preference given to a smaller β over a smaller α in the case when two combinations have the same M_3' value.

Note that the above ordering is uniquely defined since there is a one-to-one correspondence between (M_1, M_2, M_3) and (α, β, γ).

Recall that there will be a maximum of eight sine-cosine combinations for the given frequency combination of (α, β, γ). All of these sine-cosine combinations are considered to be equally important and hence their ordering can be arbitrarily chosen. The analysis of the proposed frequency ordering considers the following ordering of the sine-cosine combinations for the given frequency combinations: (cos, cos, cos), (cos, cos, sin), (cos, sin, cos),

(cos, sin, sin), (sin, cos, cos), (sin, cos, sin), (sin, sin, cos), and (sin, sin, sin).

Using the above heuristics, the first few frequency combinations in the proposed ordering for the homogeneously sampled image data set are given in Table 13, while the entire ordering is shown in Fig. 29. Because $L = M = N = 9$, note that there are five "real" frequencies (0 through 4) along all three dimensions for this image data set resulting in a total of 125 frequency combinations with the maximum sum of three frequencies being 12. Figure 30 shows that the frequency combinations in this proposed ordering give a very good approximation as compared to that using the optimum ordering (both individually and as a group of sine-cosine combinations for each frequency combination) in terms of the energy recovery ratio.

Table 13: First few frequency combinations for homogeneously sampled image data set

α	0	0	1	0	0	1	0	1	2	0	0	1	0	1	2	0	2	1	3	0
β	0	0	0	1	0	0	1	1	0	2	0	0	1	1	0	2	1	2	0	3
γ	0	1	0	0	2	1	1	0	0	0	3	2	2	1	1	1	0	0	0	0

Now consider the variation of images along the three parameters for the non-homogeneously sampled image data set shown in Fig. 26. Because the variations along the parameters α_l and β_m remain the same as in the homogeneously sampled image data set, the only data that needs to be generated again are the images varying along the parameter γ_n. In particular, the image in the center (corresponding to $\alpha_l = \beta_m = \gamma_n = 0°$) of Fig. 27 is now planar rotated from 0° to 320° to obtain nine images in 40° increments and the resulting image set is used as a representative example for evaluating the variation of images in the parameter γ_n. Figure 31 shows the entries of the first nine right singular vectors of the non-homogeneously sampled image data set corresponding to the variation of images in three parameters. Again, note that the image data set under consideration has $LMN = 729$ images in total. Hence, the corresponding right singular vectors will each have dimension of 729. However, only nine of those 729 entries are used to monitor the variation of images along one parameter keeping the other two parameters constant.

Figure 31 also shows that the right singular vector entries corresponding to variation of

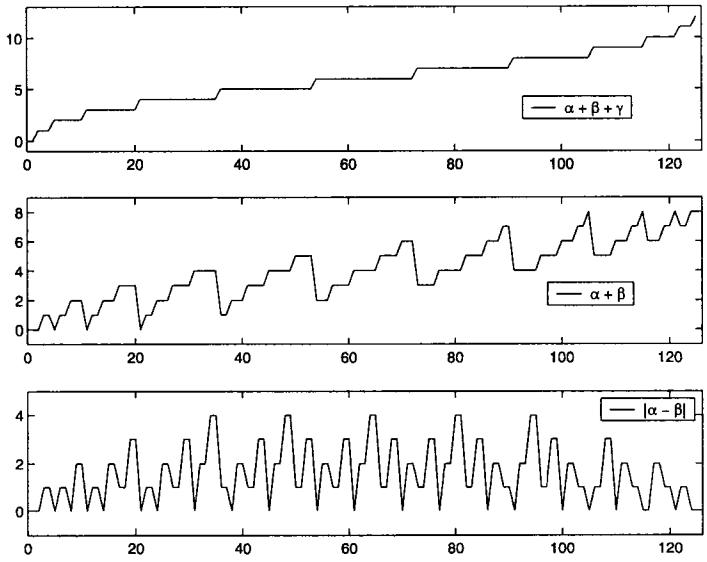


Figure 29: The plots in this figure give a pictorial representation of the proposed ordering of all frequency combinations for the homogeneously sampled image data set shown in Fig. 25. These plots are shown as a function of the frequency combination index and follow the measures M_1, M_2, and M_3 outlined for ordering these combinations for the homogeneously sampled image data set.

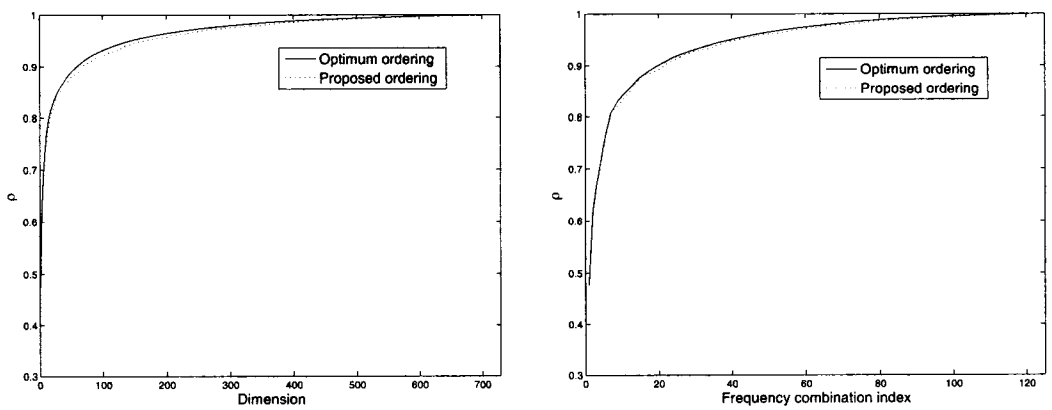


Figure 30: The plots in this figure show the comparison of the proposed ordering with the optimum ordering of the frequency combinations in terms of their energy recovery ability in $\breve{X}$ for the homogeneously sampled image data set shown in Fig. 25. The plot on the left shows the energy recovery ratio as a function of subspace dimension, while the plot on the right shows the energy recovery ratio as a function of frequency combination index. Recall that if r is the total number of non-zero α, β, and γ frequencies, then there will be 2^r sine-cosine combinations in $\breve{H}$.

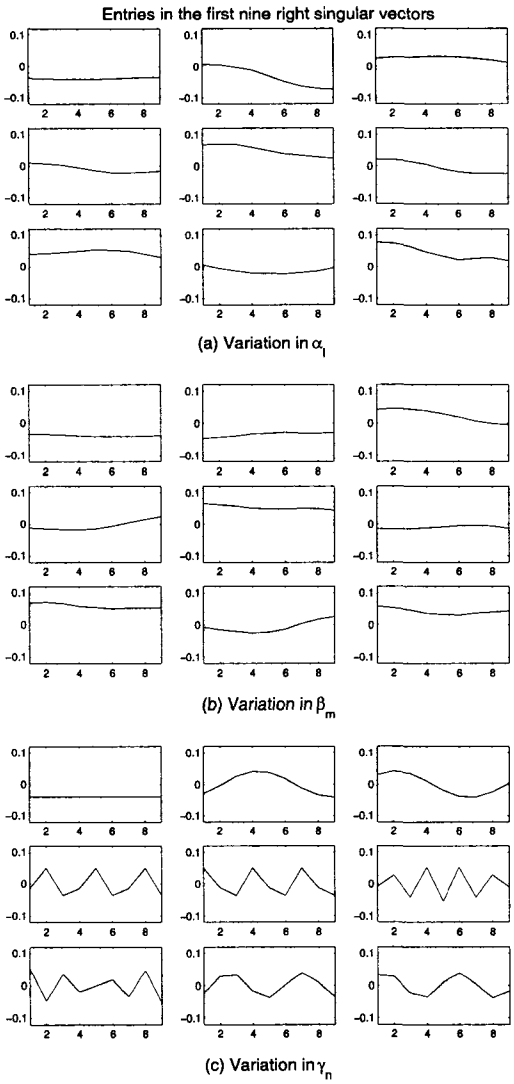


Figure 31: The plots in this figure show entries of the first nine right singular vectors of the non-homogeneously sampled image data set shown in Fig. 26. In particular, Part (a) shows entries of the first nine right singular vectors corresponding to variation of nine images along the α_l parameter with $\beta_m = \gamma_n = 0^\circ$. (The corresponding images are shown in the middle row of Fig. 27.) Part (b) shows entries of the same right singular vectors corresponding to variation of nine images along the β_m parameter with $\alpha_l = \gamma_n = 0^\circ$ (refer to the middle column of Fig. 27), while part (c) shows entries of the same nine right singular vectors corresponding to variation of nine images along the γ_n parameter with $\alpha_l = \beta_m = 0^\circ$. (The image in the center of Fig. 27 is planar rotated from 0° to 320° in 40° increments to obtain the corresponding images.) Note that only nine (out of a total of $LMN = 729$) right singular vector entries are plotted in each plot here. In each part, the right singular vectors are numbered from 1 through 9 from left to right and top to bottom.

images along each parameter are well-approximated by sinusoids of increasing frequencies starting from zero frequencies. However, the sum of frequencies does not seem to be a good starting measure to order the frequency components for this image data set, as even the first few right singular vectors seem to contain much higher γ frequencies. In particular, apart from the zero frequency, all four γ frequencies[5] are as important as the first α and β frequencies. This is due to the fact that the range of the parameter γ_n (360°) is four times larger than that of the parameters α_l and β_m. However, within the same γ frequency, it appears that α and β frequencies should be ordered in their increasing sum.

Now refer to the second column of Table 12, which lists the image data sets with the same separation (10 degrees) between adjacent images along all three parameters. For the $9 \times 9 \times 36$ and $9 \times 9 \times 18$ image data sets, it was observed that four and two γ frequencies were as important as one α or β frequencies, respectively. This indicates that when the range of the γ_n parameter is larger than for the other two parameters, the importance of γ frequencies as compared to α and β frequencies is directly proportional to the ratio of the corresponding parameter ranges. On the other hand, if the range of the γ_n parameter is smaller than the other two parameters, then it was noted that the relative importance of the three frequencies changed slightly. More specifically, even though the range of β_m is four times as much as that of γ_n in the $9 \times 36 \times 9$ image data set, four β frequencies were as important as two γ frequencies and one α frequency. Similarly for the $9 \times 18 \times 9$ image data set, two β frequencies were as important as two γ frequencies and one α frequency. This is because of the fact that γ frequencies are inherently more important than α and β frequencies. It was again observed that α and β frequencies were interchangeable. In particular, $9 \times 36 \times 9$ and $36 \times 9 \times 9$ image data sets indicated same frequency ordering with α and β frequencies interchanged.

Finally, it was observed that the image data sets in the last column of Table 12 derive from the image data sets in the other two columns. In particular, it was observed that for $9 \times 9 \times 36$, $9 \times 9 \times 18$, and $9 \times 9 \times 9$ image data sets, four γ frequencies were as important as

[5]Because $L = M = N = 9$, note that there are five "real" frequencies (0 through 4) along all three parameters for the different-resolution image data set.

one α and β frequency as the ranges of the three dimensions were the same. Similarly, for $9 \times 36 \times 9$, $9 \times 18 \times 9$, and $9 \times 9 \times 9$ image data sets, four β frequencies were as important as two γ frequencies and one α frequency.[6]

The above observations motivate ordering the frequency combinations for the non-homogeneously sampled image data sets as follows. Recall that α_l and β_m are essentially interchangeable, while the parameter γ_n is different in nature. This becomes more pronounced when the range for γ_n is different than for the other parameters. This motivates a non-homogeneous ordering where the measures are now given by:

- $M_1 = \alpha + \beta$
- $M_2 = \alpha - \beta$
- $M_3 = \gamma$

These measures put a higher priority on α and β, as compared to the homogeneous ordering case, so that higher frequencies in γ are considered sooner. However, like the homogeneous case, there is a preference for combinations of lower α and β frequencies as opposed to combinations with a high frequency.

Let η, τ, and δ represent the ratios between the ranges of α_l, β_m, and γ_n, i.e., $(\eta : \tau : \delta = \max(\alpha_l) - \min(\alpha_l) : \max(\beta_m) - \min(\beta_m) : \max(\gamma_n) - \min(\gamma_n))$, where η, τ, and δ must be integers with no common factors. It is also assumed that all parameters are sampled with equiangular increments. One can order the family of 3-tuples of non-negative integers in the following way:

- First, consider the set of 3-tuples of integers, (α, β, γ), such that $0 \leq \alpha \leq \eta$, $0 \leq \beta \leq \tau$, and $0 \leq \gamma \leq \delta$. There are $(\eta + 1)(\tau + 1)(\delta + 1)$ such combinations. Order these 3-tuples using the non-homogeneous ordering measures described above and denote the ordered set by S_1.

[6]Note that the $9 \times 9 \times 9$ image data set can also be classified as the homogeneously sampled image data set and hence it obeys the rule of the sum of frequencies followed by the higher γ frequencies being more important within the subgroups. For the $9 \times 9 \times 3$ and the $9 \times 9 \times 2$ image data sets, the sum of frequencies ordering heuristic is still near optimal, however, because more images are sampled along α and β, higher α and β frequencies should be placed ahead of higher γ frequencies within a group with the same sum.

- Next, starting with the set of 3-tuples of integers, (α, β, γ), such that $0 \leq \alpha \leq 2\eta$, $0 \leq \beta \leq 2\tau$, and $0 \leq \gamma \leq 2\delta$, remove all those 3-tuples that are in S_1. Order the remaining 3-tuples using non-homogeneous ordering, concatenate this ordered set with S_1, and denote the resulting ordered set as S_2.

- Continue the above process inductively by the following rule. Remove the 3-tuples in S_n from the set of 3-tuples (α, β, γ), such that $0 \leq \alpha \leq (n+1)\eta$, $0 \leq \beta \leq (n+1)\tau$, and $0 \leq \gamma \leq (n+1)\delta$. Order the remaining $[(n+1)\eta+1][(n+1)\tau+1][(n+1)\delta+1] - (n\eta+1)(n\tau+1)(n\delta+1)$ 3-tuples using non-homogeneous ordering and concatenate this ordered list to S_n to obtain S_{n+1}.

The above procedure is illustrated in Fig. 32 for $(\eta : \tau : \delta = 1 : 1 : 2)$.

Using the above heuristics, the first few frequency combinations in the proposed ordering for the non-homogeneously sampled image data set are given in Table 14, while the entire ordering is shown in Fig. 33. Because $L = M = N = 9$, note that there are five "real" frequencies (0 through 4) along all three dimensions for this image data set resulting in a total of 125 frequency combinations with the maximum sum of three frequencies being 12. Figure 34 shows that the frequency combinations in this proposed ordering give a very good approximation as compared to that using the optimum ordering (both individually and as a group of sine-cosine combinations for each frequency combination) in terms of the energy recovery ratio. The proposed ordering of frequency combinations for both homogeneously sampled and non-homogeneously sampled image data sets is also visually evaluated in Figure 35, which shows that the $\breve{X}\breve{H}$ images corresponding to the proposed ordering for both image data sets give a very good approximations of the true eigenimages.

Table 14: First few frequency combinations for the non-homogeneously sampled image data set

α	0	0	$\cdots$	0	1	0	1	0	$\cdots$	1	0	1	1	$\cdots$	1	2	0	2	0	$\cdots$	2	0
β	0	0	$\cdots$	0	0	1	0	1	$\cdots$	0	1	1	1	$\cdots$	1	0	2	0	2	$\cdots$	0	2
γ	0	1	$\cdots$	4	0	0	1	1	$\cdots$	4	4	0	1	$\cdots$	4	0	0	1	1	$\cdots$	4	4

$S_1 =$

α	0	0	0	1	0	1	0	1	0	1	1	1
β	0	0	0	0	1	0	1	0	1	1	1	1
γ	0	1	2	0	0	1	1	2	2	0	1	2

$S_2 - S_1 =$

α	0	0	1	0	1	0	1	1	2	$\cdots$	0	2	$\cdots$	1	2	$\cdots$	2
β	0	0	0	1	0	1	1	1	0	$\cdots$	2	1	$\cdots$	2	2	$\cdots$	2
γ	3	4	3	3	4	4	3	4	0	$\cdots$	4	0	$\cdots$	4	0	$\cdots$	4

Figure 32: This figure shows two ordered sets of 3-tuples (S_1 and $S_2 - S_1$) using non-homogeneous ordering measures for the case where $(\eta : \tau : \delta)$ is equal to $(1 : 1 : 2)$.

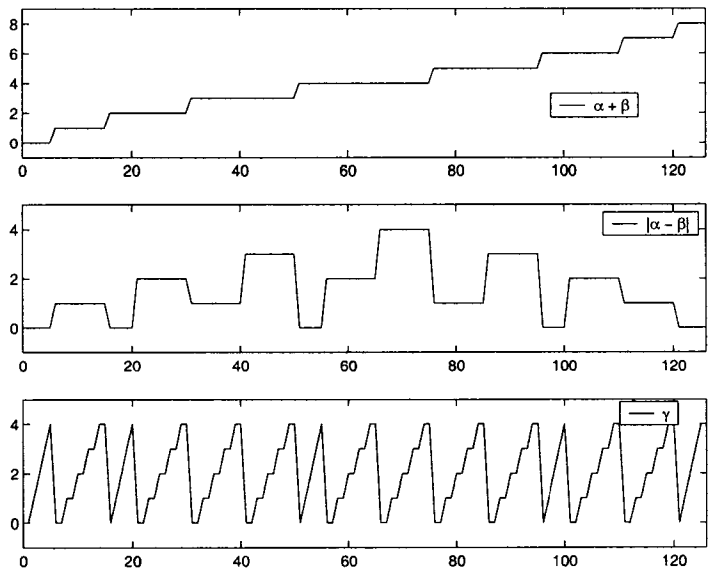


Figure 33: The plots in this figure give a pictorial representation of the proposed ordering of all frequency combinations for the non-homogeneously sampled image data set shown in Fig. 26. These plots are shown as a function of the frequency combination index and follow the measures M_1, M_2, and M_3 outlined for ordering these combinations for the non-homogeneously sampled image data set.

The proposed heuristics for frequency ordering for the given image data set can be summarized as follows. One should look at the range of all three parameters. If the range of all three parameters is the same, then frequency combinations should be ordered in exactly the same way as for the homogeneously sampled image data sets (refer to Table 13). If the range is different, then the relative importance of α, β, and γ frequencies need to be identified and the frequency ordering for the non-homogeneously sampled image data set

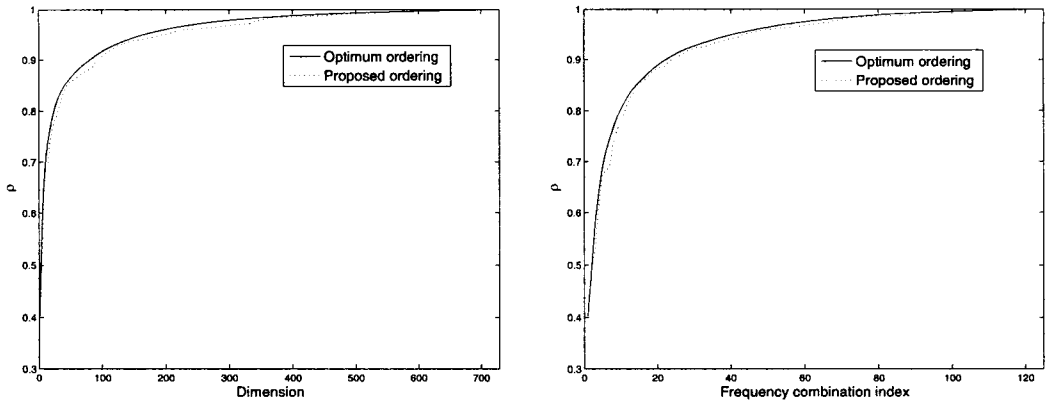


Figure 34: The plots in this figure show the comparison of the proposed ordering with the optimum ordering of the frequency combinations in terms of their energy recovery ability in $\breve{X}$ for the non-homogeneously sampled image data set shown in Fig. 26. The plot on the left shows the energy recovery ratio as a function of subspace dimension, while the plot on the right shows the energy recovery ratio as a function of frequency combination index. Recall that if r is the total number of non-zero α, β, and γ frequencies, then there will be 2^r sine-cosine combinations in $\breve{H}$.

(refer to Fig. 32) should be used.[7] These heuristics for ordering the frequency combinations for a given $\breve{X}$ along with the efficient computation of $\breve{X}\breve{H}$ using FFT techniques (refer to the previous section) can be used to extend Chang's eigendecomposition algorithm to compute the SVD of $\breve{X}$, which is the topic of the next section.

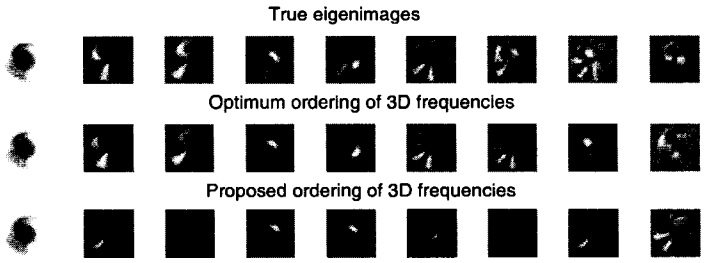
6.5 *A Fast Eigendecomposition Algorithm for 3D Image Data Sets*

The entire algorithm for the fast computation of a partial SVD of $\breve{X}_{m\times n}$, where $n = LMN$, can be summarized as follows:

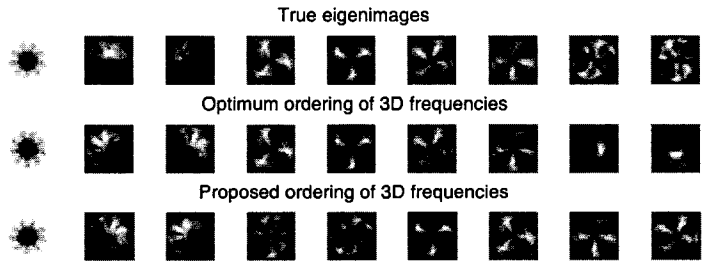
1. Form the matrix $Y = \breve{X}\breve{H}$ using fast Fourier transform techniques described in Section 6.3 such that the columns of Y are placed in the order discussed in Section 6.4 based on the range of the α_l, β_m, γ_n parameters and the angular separation between the images.

2. Determine the smallest number p such that $\rho(\breve{X}^T, \mathbf{h}_1, \cdots, \mathbf{h}_p) > \mu$, where μ is the

[7]Note that these heuristics were developed for the image data sets with equiangular separation between the images along all three parameters.



(a) Homogeneously sampled image data set



(b) Non–homogeneously sampled image data set

Figure 35: This figure shows true eigenimages of the homogeneously sampled and non-homogeneously sampled image data sets along with the multiplication $\breve{X}\breve{H}$ in the form of images using the optimum and proposed ordering of frequency combinations. Parts (a) and (b) show the results for the homogeneously sampled and the non-homogeneously sampled image data set, respectively. In particular, the first row of images in both parts show the first nine true eigenimages. The second row shows the $\breve{X}\breve{H}$ images using the first nine optimally ordered columns of $\breve{H}$ in terms of their ability to recover energy in $\breve{X}$, while the third row shows $\breve{X}\breve{H}$ images using the first nine columns that are placed in $\breve{H}$ using the proposed ordering.

user-specified reconstruction ratio. The key observation here is that the matrix $\breve{X}\breve{H}_p$ can be constructed directly from the first p columns of Y.

3. Compute the SVD of $\breve{X}\breve{H}_p = \sum_{i=1}^{p} \tilde{\sigma}_i \tilde{\hat{\mathbf{u}}}_i \tilde{\hat{\mathbf{v}}}_i^T$.

4. Return $\tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k$ such that $\rho(\breve{X}, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k) \geq \mu$.

The computational expense of the proposed algorithm is now analyzed. The cost incurred in Step 1, i.e., performing the FFT of the 3D image data matrix $\breve{X}$, requires $O(m(n+s)\log_2(n+s))$ flops, where $s = 2(LM + LN + MN) + 4(L + M + N) + 8$.[8]

[8]This computational analysis is for all L, M, and N being even. If any or all these numbers are odd, the computational efficiency improves slightly.

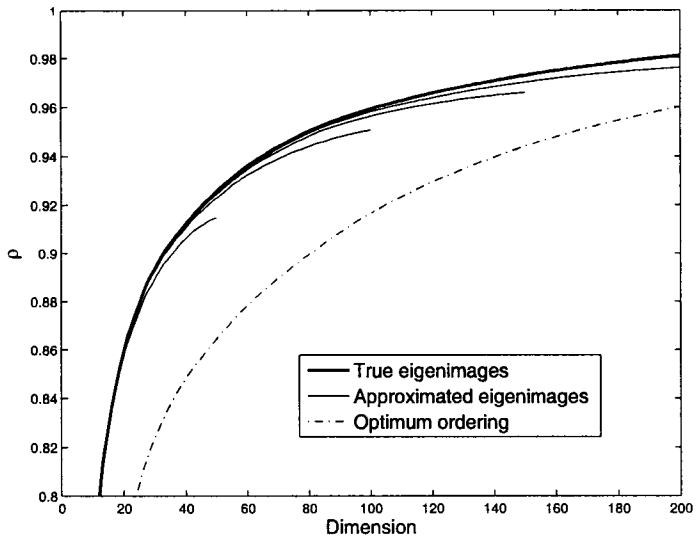


Figure 36: This figure shows the typical relationship between the true left singular vectors, the computed estimates (as a function of k, $1 \leq k \leq p$, for several fixed values of p), and ordered frequency combinations. The plots shown here are for the 3D image data matrix generated for the homogeneously sampled image data set shown in Fig. 25 with $L = M = N = 9$ and 90, 90, 360 degree range for α_l, β_m, and γ_n parameters.

Step 2, that of estimating p, requires $O(mp)$ flops. In Step 3, the cost of computing the SVD of the matrix comprising the first p columns of $\breve{X}\breve{H}$ is $O(mp^2)$. Step 4, determining the needed dimension k, requires $O(mnk)$ flops. If $p \ll n$, then the total computation required is approximately $O(m(n+s)\log_2(n+s))$, otherwise it is approximately $O(mp^2)$. This compares favorably with the direct SVD approach, which requires $O(mn^2)$ flops.

For 1D image data sets, it was observed that the individual pixel values do not change rapidly across the sequence of images resulting in $p \ll n$. However, for 3D image data sets, the value of p increases because the images are not as highly correlated and the right singular vectors are not as closely approximated by pure sinusoids. To illustrate this phenomenon, consider the homogeneously sampled image data set shown in Fig. 25 with $L = M = N = 9$. Let the columns of $\breve{H}$ be in the descending order of their ability to recover energy in $\breve{X}$ using $\rho(\breve{X}^T, \breve{\mathbf{h}}_1, \breve{\mathbf{h}}_2, \cdots)$. Then the dotted line in Fig. 36 shows $\rho(\breve{X}, \breve{\mathbf{h}}_1, \cdots, \breve{\mathbf{h}}_p)$ as a function of p, while the solid lines show $\rho(\breve{X}, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k)$ for $k = 1, 2, \cdots, p$ and $p = 50, 100, 150, 200$. It is evident that while the $\tilde{\mathbf{u}}_i$ give good estimates of the $\hat{\mathbf{u}}_i$, $\rho(\breve{X}^T, \breve{\mathbf{h}}_1, \breve{\mathbf{h}}_2, \cdots)$ does not give

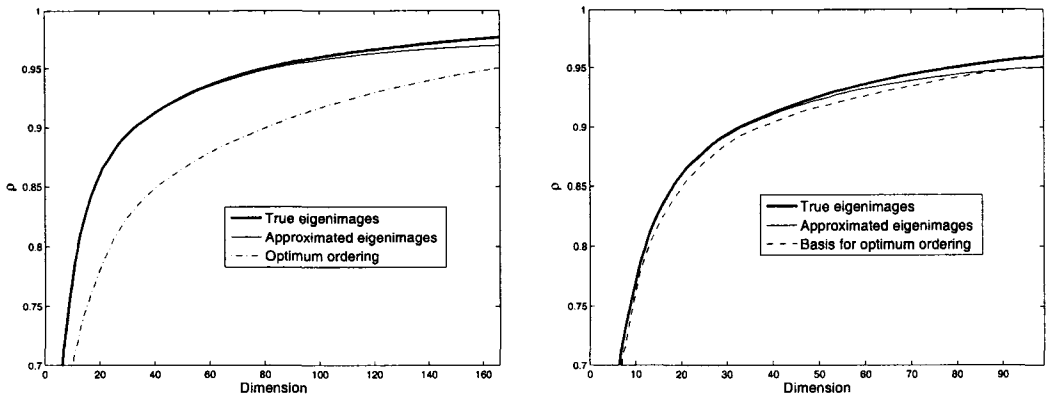


Figure 37: This figure shows the energy recovery ratio plots for $\breve{X}$ using the columns of the $\breve{X}\breve{H}$ matrix in part (a) for computing the approximated eigenimages versus an orthonormal basis for the $\breve{X}\breve{H}$ matrix in part (b).

as tight a lower bound on $\rho(\breve{X}, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_k)$ as in 1D case.

It is possible to provide a tighter bound on $\rho(\breve{X}, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_p)$ by using an orthonormal basis for the range of $\breve{X}\breve{H}_p$, however, the following example illustrates why this may not be desirable. Consider the case shown in Fig. 37 where the user-specified reconstruction ratio $\mu = 0.95$. The required dimension of the true eigenspace for the homogeneously sampled image data set is $k^* = 80$, however, the first $p = 166$ optimally ordered columns of $\breve{X}\breve{H}$ are required to attain $\rho(\breve{X}, \breve{\mathbf{h}}_1, \cdots, \breve{\mathbf{h}}_p) \geq \mu$. If all these p columns of $\breve{X}\breve{H}$ are used, then the approximated eigenimages result in $\rho(\breve{X}, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_p) = 0.9693$ (refer to Fig. 37 (a)) with $k = 83$. This indicates that a large number of frequency combinations are required to span the first k eigenimages. Now if the columns of $\breve{X}\breve{H}$ are orthonormalized using the QR decomposition ($\breve{X}\breve{H} = QR$), then $\rho(\breve{X}, \mathbf{q}_1, \cdots, \mathbf{q}_{99}) \geq \mu$, where $\mathbf{q}_i$ denotes the i^{th} column of the resulting Q matrix. Thus the value of p is reduced from 166 to 99, which potentially improves the offline computational efficiency of calculating the SVD of $\breve{X}$. The approximated eigenimages for $\breve{X}$ would now be given by the columns of $\breve{U} = QU_r$ where U_r denotes the matrix containing the left singular vectors of R. Figure 37 (b) shows that these approximated eigenimages result in $\rho(\breve{X}, \tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_{99}) = 0.9503$, which satisfies the user-specified accuracy very closely. However, the resulting dimension of the approximate eigenspace is now $k = 99$, which results in a significant increase in online computations,

which are proportional to the dimension of the subspace used in the approximation. Note that the difference between k and k^* increases drastically as the number of images in $\breve{X}$ increases. Also there is no simple method to determine *a priori* how many columns of $\breve{X}\breve{H}$ should be considered for computing its QR decomposition.

6.6 *Experimental Results*

The proposed extension of Chang's algorithm for computation of the partial SVD of 3D image data sets was evaluated using eight artificial (ray-traced) objects in Fig. 24 and sixteen real objects in Fig. 38. In particular, the algorithm was used to calculate the partial SVD of $\breve{X}$ with $\mu = 0.95$ for three image data sets for each artificial object and for two image data sets for each real object. Table 15 explains the specifications of the corresponding image data sets.[9]

Table 15: Image data sets generated for each object in Fig. 24 and 38

Artificial objects			
Image data set	Number of images	Range	Angular separation
1	$10 \times 10 \times 10$	81, 81, 81	9, 9, 9
2	$10 \times 10 \times 10$	81, 81, 324	9, 9, 36
3	$9 \times 9 \times 36$	80, 80, 360	10, 10, 10
Real objects			
Image data set	Number of images	Range	Angular separation
1	$9 \times 9 \times 9$	60, 60, 60	7.5, 7.5, 7.5
2	$9 \times 9 \times 33$	60, 60, 240	7.5, 7.5, 7.5

Tables 16, 17 and 18 summarize the performance of the algorithm, showing p, k, k^*, and the computation times for image data sets for artificial objects. Compared to the direct SVD, the speed-up factors with the proposed algorithm are in the range of $3.00 - 17.00$ with averages of 6.28, 5.93, and 7.31 for these three image data sets. Because the original image frames for all the objects have the same resolution, the speed-up factor here depends on the values of p and k. This is evident from all the table entries, as the minimum speed-up factors are obtained for objects 5 and 7 with the maximum obtained for object 2. Note

[9] Due to the physical limitations of the robot, the range for the α_l and β_m parameters for the real objects was restricted to 60° only.

Table 16: Performance of the proposed algorithm on image data set 1 for artificial objects (all times are in seconds)

				Time required for different steps					Speed-up	Subspace
Object	p	k	k^*	1	2	3	4	Total	factor	Criterion
1	238	105	99	6.53	0.024	3.23	3.73	**13.52**	7.81	0.9676
2	61	36	34	6.54	0.006	0.46	1.26	**8.27**	12.77	0.9679
3	485	183	161	6.53	0.049	10.54	6.50	**23.62**	4.47	0.9555
4	299	82	73	6.53	0.030	4.56	2.91	**14.02**	7.53	0.9508
5	625	248	222	6.53	0.063	17.43	8.77	**32.81**	3.22	0.9526
6	536	227	201	6.53	0.054	12.77	8.03	**27.39**	3.86	0.9508
7	638	209	192	6.53	0.065	17.60	7.37	**31.58**	3.34	0.9637
8	292	110	101	6.54	0.029	4.15	3.88	**14.63**	7.22	0.9556

Table 17: Performance of the proposed algorithm on image data set 2 for artificial objects (all times are in seconds)

				Time required for different steps					Speed-up	Subspace
Object	p	k	k^*	1	2	3	4	Total	factor	Criterion
1	259	126	121	6.53	0.026	3.49	4.46	**14.51**	7.28	0.9750
2	73	42	40	6.54	0.007	0.64	1.49	**8.68**	12.17	0.9815
3	494	240	215	6.53	0.050	10.84	8.50	**25.92**	4.07	0.9556
4	277	101	93	6.54	0.028	4.07	3.57	**14.21**	7.43	0.9536
5	641	309	277	6.54	0.065	17.56	10.95	**35.18**	3.00	0.9538
6	541	261	235	6.53	0.054	12.98	9.27	**28.92**	3.65	0.9532
7	646	250	228	6.54	0.065	17.84	8.94	**33.43**	3.16	0.9593
8	307	129	122	6.54	0.031	4.77	4.64	**16.01**	6.60	0.9686

Table 18: Performance of the proposed algorithm on image data set 3 for artificial objects (all times are in seconds)

				Time required for different steps					Speed-up	Subspace
Object	p	k	k^*	1	2	3	4	Total	factor	Criterion
1	626	194	184	38.20	0.06	16.77	64.20	**119.23**	7.23	0.9712
2	151	73	70	40.97	0.01	2.05	7.65	**50.70**	17.00	0.9871
3	898	351	320	39.40	0.33	85.66	37.00	**162.40**	5.31	0.9529
4	577	147	139	40.16	0.14	14.33	25.50	**80.15**	10.75	0.9727
5	1236	457	425	40.67	0.41	134.22	48.59	**223.91**	3.85	0.9631
6	1044	418	382	39.91	0.15	113.41	66.14	**219.62**	3.92	0.9566
7	1333	392	373	39.55	0.46	146.36	42.32	**248.71**	3.47	0.9786
8	736	185	175	40.37	0.08	23.02	36.97	**124.56**	6.92	0.9642

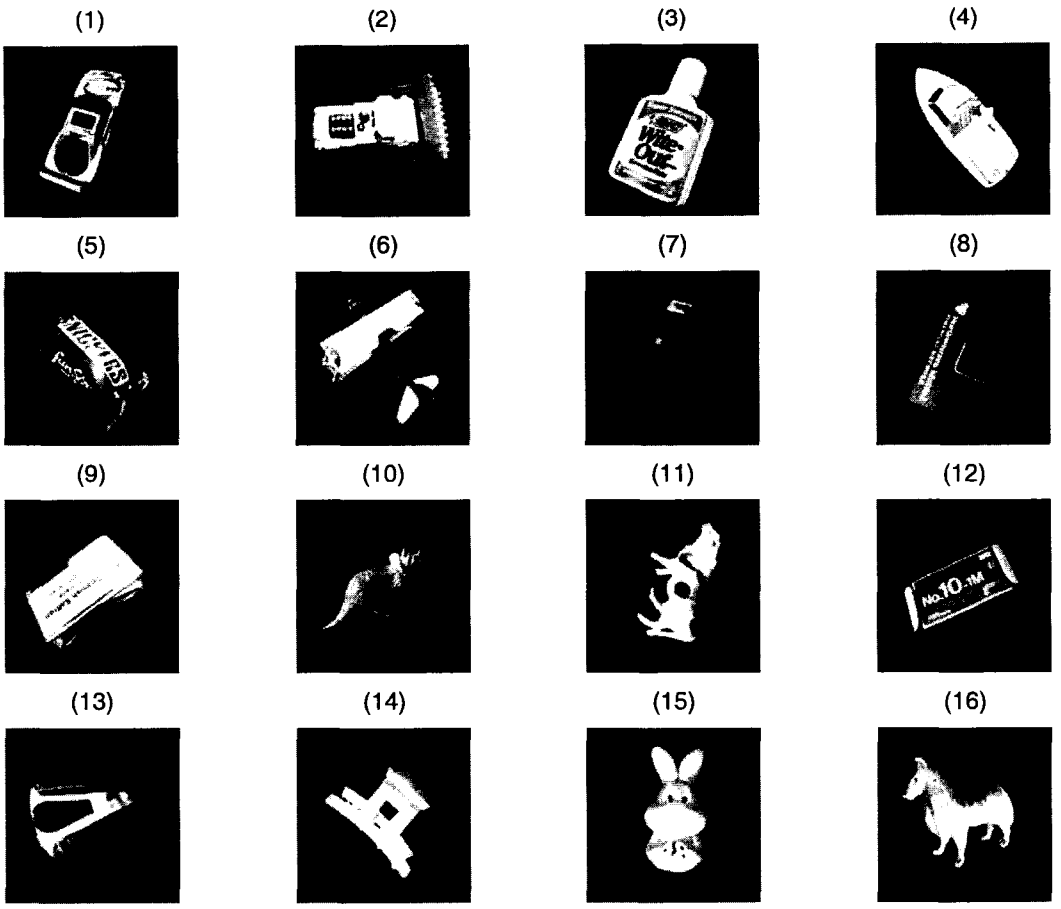


Figure 38: This figure shows sixteen real objects that are used in this study. Each image of the object is of size 128×128, resulting in an image data matrix $\breve{X}$ of size $2^{14} \times LMN$ for each object.

that despite the higher number of images in data set three that requires an increase in the time to perform step 4, the overall speed-up factors are comparable. Note that the value of p is much larger than k in most of the cases indicating that a large number of frequency combinations are required to span the first k eigenimages. It is also interesting to note that the value of k is also generally large (with respect to the number of images in the image data set) indicating that the 3D image data sets have far less correlation between images than in the 1D case.

Tables 19 and 20 summarize the performance of the algorithm, showing p, k, k^*, and

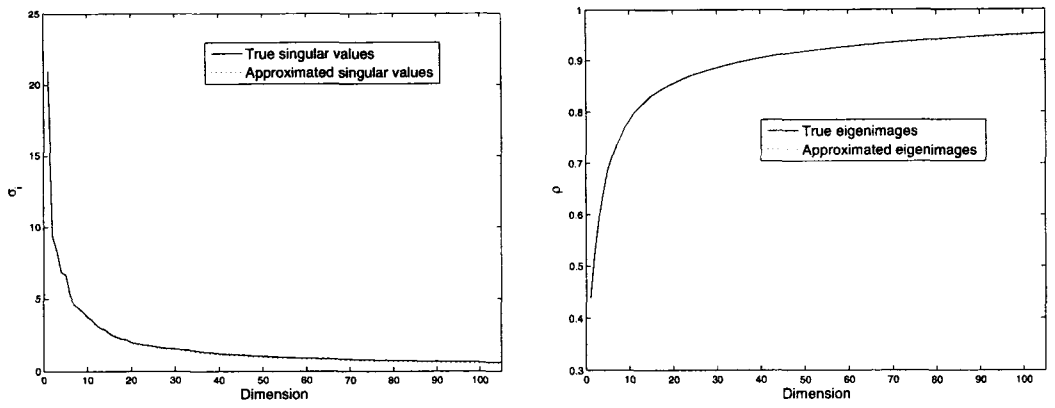


Figure 39: This figure shows a comparison of the approximated SVD with the true SVD for image data set 1 (refer to Table 15) for the artificial object 1 from Fig. 24. The left plot shows the difference between true and approximated singular values and the right plot shows the difference between energy recovery ratio using true and approximated eigenimages, when the subspace dimension is varied from 1 to k.

the computation times for image data sets for real objects. Compared to the direct SVD, the speed-up factors with the proposed algorithm for these real objects are in the range of $3.82 - 22.47$ with averages of 7.80 and 21.64 for the two image data sets. Note that the proposed algorithm performs better on real objects than on artificial objects. Additionally, if the user-specified accuracy μ is reduced to 0.90, then the range of speed-up factors of the proposed algorithm as compared to the direct SVD improves to $5.23 - 23.12$ with averages of 9.74, 9.19, and 14.35 for image data sets for artificial objects, and averages of 10.26 and 32.48 for image data sets for real objects.

The quality of the resulting eigendecomposition was also evaluated using the error measures described in Section 2.4 for both real and artificial objects. In particular, Tables 16 - 20 show that the subspace criterion measure between the true and approximated eigenimages for all image data sets is in the range of $0.9213 - 0.9930$. Also, the difference between $\rho(\breve{X}, \hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k^*})$ and $\rho(\breve{X}, \tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k)$ for each set was less than 0.17%, with an average of 0.09%, which reveals that $\{\tilde{\hat{\mathbf{u}}}_1, \cdots, \tilde{\hat{\mathbf{u}}}_k\}$ provides a very good approximate basis for the first k^* eigenimages $\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k^*}\}$. The energy recovery ratio plots in Figure 39 also show that the proposed algorithm give virtually perfect image reconstruction.

Table 19: Performance of the proposed algorithm on image data set 1 for real objects (all times are in seconds)

				Time required for different steps					Speed-up	Subspace
Object	p	k	k^*	1	2	3	4	Total	factor	Criterion
1	351	154	132	4.35	0.03	6.25	4.07	**14.72**	3.82	0.9320
2	259	108	91	4.35	0.02	3.51	2.86	**10.76**	5.23	0.9332
3	245	89	76	4.33	0.02	3.44	2.30	**10.11**	5.56	0.9220
4	56	12	11	4.35	0.00	0.38	0.31	**5.06**	11.12	0.9468
5	115	44	36	4.35	0.01	1.52	1.13	**7.03**	8.03	0.9262
6	288	93	79	4.35	0.02	4.45	2.41	**11.25**	5.00	0.9276
7	208	84	75	4.36	0.02	2.70	2.18	**9.27**	6.07	0.9261
8	134	28	26	4.35	0.01	1.55	0.73	**6.66**	8.44	0.9261
9	22	5	4	4.35	0.00	0.07	0.13	**4.56**	12.33	0.9930
10	94	13	13	4.36	0.01	1.03	0.35	**5.75**	9.78	0.9756
11	221	61	56	4.35	0.02	3.11	1.60	**9.10**	6.18	0.9449
12	207	58	50	4.36	0.05	2.71	1.50	**8.64**	6.51	0.9260
13	194	61	55	4.36	0.01	2.32	1.61	**8.33**	6.75	0.9300
14	67	14	14	4.36	0.00	0.54	0.37	**5.29**	10.63	0.9375
15	21	5	5	4.37	0.00	0.06	0.13	**4.57**	12.31	0.9812
16	184	41	38	4.35	0.01	2.43	1.10	**7.91**	7.11	0.9530
								Average	**7.80**	**0.9426**
								Minimum	**3.82**	**0.9220**
								Maximum	**12.33**	**0.9930**

Table 20: Performance of the proposed algorithm on image data set 2 for real objects (all times are in seconds)

				Time required for different steps					Speed-up	Subspace
Object	p	k	k^*	1	2	3	4	Total	factor	Criterion
1	1215	346	301	16.43	0.31	76.50	33.39	**126.72**	5.85	0.9338
2	874	205	182	16.42	0.22	35.67	19.79	**71.95**	10.29	0.9367
3	823	198	170	16.49	0.21	31.77	19.13	**67.59**	10.96	0.9257
4	159	25	24	16.38	0.04	2.25	2.43	**21.11**	35.10	0.9537
5	391	96	84	16.45	0.10	7.12	9.30	**32.99**	22.46	0.9300
6	975	169	147	16.55	0.25	45.31	16.31	**78.43**	9.45	0.9286
7	680	135	123	16.39	0.17	20.35	13.02	**49.94**	14.84	0.9423
8	422	64	57	16.39	0.10	8.05	6.18	**30.74**	24.11	0.9213
9	53	7	7	16.40	0.01	0.35	0.67	**17.43**	42.51	0.9778
10	296	26	25	16.39	0.07	4.24	2.51	**23.24**	31.88	0.9533
11	717	142	128	16.45	0.18	22.76	13.69	**53.09**	13.96	0.9449
12	706	140	120	16.40	0.18	21.94	13.54	**52.07**	14.03	0.9304
13	629	114	105	16.40	0.16	17.56	11.02	**45.15**	16.41	0.9358
14	184	28	25	16.48	0.04	2.41	2.70	**21.65**	34.23	0.9355
15	55	9	9	16.41	0.01	0.37	0.87	**17.68**	41.91	0.9909
16	596	89	84	16.43	0.15	15.32	8.59	**40.51**	18.29	0.9536
								Average	**21.64**	**0.9433**
								Minimum	**5.85**	**0.9213**
								Maximum	**42.51**	**0.9909**

CHAPTER VII

CONCLUSION

7.1 Summary

This work addressed two issues associated with the calculation of the eigendecomposition of correlated images, i.e., (1) the effect of spatial resolution reduction on the temporal properties of video sequences, and (2) the correlations associated with three-dimensional pose estimation.

In the first part of this thesis, a theoretical background was presented for quantifying the tradeoff associated with performing eigendecomposition of one-dimensional correlated images at lower resolutions in order to mediate the high computational expense of performing these calculations at high resolutions. While doing so, it was shown that the low-resolution right singular vectors can be effectively used to approximate the high-resolution eigenimages without introducing significant error. For reducing the original images to some lower resolutions, two different resolution reduction methods were considered, i.e., low-pass filtering techniques and random sampling. A mathematical analysis showed that downsampling using random sampling is more effective than any low-pass filtering technique as it does not affect the temporal properties of the original images. Finally, the similarity between the right singular vectors of correlated images at different resolutions was used to improve the computational efficiency of Chang's algorithm for calculating eigendecomposition. The proposed algorithm enjoys the advantage of making use of both spatial and temporal correlations between the one-dimensional image data sets. Examples showed that the algorithm performed very well on the images of objects rotated along a single axis and on arbitrary video sequences.

The second part of this thesis considered the efficient computation of the eigendecomposition of general three-dimensional image sets that are parameterized by three different parameters. It was shown that the three-dimensional frequency properties of these image data sets can be used to extend Chang's algorithm to such image data sets. In particular, two important extensions were discussed, i.e., (1) efficient computation of the multiplication of the image data matrix with three-dimensional frequency combinations using FFT techniques, and (2) optimum ordering of the frequency combinations based on the ranges of the three-dimensional parameters that parameterize the given image data sets. These extensions were successfully implemented to realize one computational efficient eigendecomposition algorithm, which performed very well on several three-dimensional image data sets with different parameterizations. This eigendecomposition algorithm can be used for computationally efficient three-dimensional pose estimation of objects.

7.2 Future Work

In the proposed one-dimensional algorithm, the images are reduced by randomly sampling pixels from the original images, so that the number of pixels in low-resolution images is on the order of the number of images. It is shown that this reduction technique works well for the proposed algorithm, however it still does not quantify the exact errors associated with performing the eigendecomposition at different resolutions. To predict these errors, it is important to have some reduction model based on the statistical properties of correlated images at high resolution; such error prediction would be useful in almost all computer vision applications. In particular, the high-resolution images can be reduced using such a model and the corresponding low-resolution SVD can be effectively used to approximate the high-resolution SVD. This can result in considerable time savings in offline as well as online calculations, thus making many applications realizable in real time.

Another interesting topic is related to superresolution reconstruction [57], which produces a high-resolution image using a set of low-resolution images. Most of the superresolution algorithms are computationally expensive and numerically unstable due to an underlying ill-conditioned and typically underdetermined large scale problem. To solve this

problem, a similar approach can be applied in the eigenspaces. In particular, if a specific resolution reduction model is assumed, then the low-resolution SVD can be computed and the corresponding high-resolution SVD can be effectively approximated. Such an approximation can then give a good estimation of the major features of the corresponding high-resolution image(s). This process can possibly be used to address the ill-conditioned and underdetermined system of equations that occurs with traditional superresolution reconstruction.

REFERENCES

[1] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *Computing Surveys*, vol. 18, no. 1, pp. 67–108, March 1986.

[2] A. M. Wallace, "Industrial applications of computer vision since 1982," *IEE Proceedings Computers and Digital Techniques*, vol. 135, no. 3, pp. 117–136, May 1988.

[3] E. R. Davies, *Machine Vision.* London: Academic Press, 1997.

[4] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* London: Academic Press, 1990.

[5] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Trans. PAMI*, vol. 23, no. 2, pp. 228–233, Feb. 2001.

[6] H. C. Andrews and C. L. Patterson, "Singular value decomposition (SVD) image coding," *IEEE Trans. Communications*, vol. COM-24, pp. 425–432, Apr. 1976.

[7] H. C. Andrews and C. L. Patterson, "Singular value decomposition and digital image processing," *IEEE Trans. ASSP*, vol. ASSP-24, no. 1, pp. 26–53, Feb. 1976.

[8] Y. S. Shim and Z. H. Cho, "SVD pseudoinversion image reconstruction," *IEEE Trans. ASSP*, vol. ASSP-29, no. 4, pp. 904–909, Aug. 1981.

[9] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *J. Opt. Soc. Amer.*, vol. 4, no. 3, pp. 519–524, March 1987.

[10] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," *IEEE Trans. PAMI*, vol. 12, no. 1, pp. 103–108, Jan. 1990.

[11] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, March 1991.

[12] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. PAMI*, vol. 19, no. 7, pp. 711–720, July 1997.

[13] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. PAMI*, vol. 15, no. 10, pp. 1042–1052, Oct. 1993.

[14] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*, Seattle, WA, USA, Jun 21-23 1994, pp. 84–91.

[15] M. H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. PAMI*, vol. 24, no. 1, pp. 34–58, Jan. 2002.

[16] H. Murase and R. Sakai, "Moving object recognition in eigenspace representation: Gait analysis and lip reading," *Pattern Recognit. Lett.*, vol. 17, no. 2, pp. 155–162, Feb. 1996.

[17] G. Chiou and J. N. Hwang, "Lipreading from color video," *IEEE Trans. Image Processing*, vol. 6, no. 8, pp. 1192–1195, Aug. 1997.

[18] H. Murase and S. K. Nayar, "Illumination planning for object recognition using parametric eigenspaces," *IEEE Trans. PAMI*, vol. 16, no. 12, pp. 1219–1227, Dec. 1994.

[19] C. Y. Huang, O. I. Camps, and T. Kanungo, "Object recognition using appearance-based parts and relations," in *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*, San Juan, PR, USA, Jun 17-19 1997, pp. 877–883.

[20] R. J. Campbell and P. J. Flynn, "Eigenshapes for 3D object recognition in range data," in *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*, Fort Collins, CO, USA, June 23-25 1999, pp. 505–510.

[21] M. Jogan and A. Leonardis, "Robust localization using eigenspace of spinning-images," in *Proc. IEEE Workshop Omnidirectional Vision*, Hilton Head Island, South Carolina, USA, June 2000, pp. 37–44.

[22] S. Yoshimura and T. Kanade, "Fast template matching based on the normalized correlation by using multiresolution eigenimages," in *IEEE Workshop Motion of Non-Rigid and Articulated Objects*, Austin, Texas, Nov. 11-12 1994, pp. 83–88.

[23] J. Winkeler, B. S. Manjunath, and S. Chandrasekaran, "Subset selection for active object recognition," in *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*, Fort Collins, Colorado, USA, June 23-25 1999, pp. 511–516.

[24] J. Ben-Arie and Z. Wang, "Pictorial recognition of objects employing affine invariance in the frequency domain," *IEEE Trans. PAMI*, vol. 20, no. 6, pp. 604 – 618, June 1998.

[25] S. Romdhani, S. Gong, and A. Psarrou, "A generic face appearance model of shape and texture under very large pose variations from profile to profile views," in *IEEE Int. Conf. Pattern Recognition*, Sept. 3 - 7 2000, pp. 1060 – 1063.

[26] J.-G. Wang, E. Sung, and R. Venkateswarlu, "Registration of infra-ref and visible-spectrum imagery for face recognition," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition (FGR04)*, May 17 - 19 2004, pp. 638 – 644.

[27] S. K. Nayar, H. Murase, and S. A. Nene, "Learning, positioning, and tracking visual appearance," in *Proc. IEEE Int. Conf. Robot. Automat.*, San Diego, CA, USA, May 8-13 1994, pp. 3237–3246.

[28] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[29] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *Int. J. Computer Vision*, vol. 14, no. 1, pp. 5–24, Jan. 1995.

[30] H. Murase and S. K. Nayar, "Detection of 3D objects in cluttered scenes using hierarchical eigenspace," *Pattern Recognit. Lett.*, vol. 18, no. 4, pp. 375–384, April 1997.

[31] S. K. Nayar, S. A. Nene, and H. Murase, "Subspace method for robot vision," *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, pp. 750–758, Oct. 1996.

[32] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. PAMI*, vol. 19, no. 7, pp. 696–710, July 1997.

[33] I. J. Good, "Some applications of the singular value decomposition of a matrix," vol. 11, pp. 823–831, Nov. 1969.

[34] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods.* New York: Wiley, 1966.

[35] G. W. Stewart, *Introduction to Matrix Computation.* New York: Academic, 1973.

[36] F. S. Action, *Numerical Methods that Work.* New York: Harper and Row, 1970.

[37] S. Shlien, "A method for computing the partial singular value decomposition," *IEEE Trans. PAMI*, vol. 4, no. 6, pp. 671–676, Nov. 1982.

[38] C. R. Vogel and J. G. Wade, "Iterative SVD-based methods for ill-posed problems," *SIAM J. Sci. Comput.*, vol. 15, no. 3, pp. 736–754, May 1994.

[39] R. Haimi-Cohen and A. Cohen, "Gradient-type algorithms for partial singular value decomposition," *IEEE Trans. PAMI*, vol. PAMI-9, no. 1, pp. 137–142, Jan. 1987.

[40] X. Yang, T. K. Sarkar, and E. Arvas, "A survey of conjugate gradient algorithms for solution of extreme eigen-problems for a symmetric matrix," *IEEE Trans. ASSP*, vol. 37, no. 10, pp. 1550–1556, Oct. 1989.

[41] L. N. Trefethen and D. Bau, *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 1997.

[42] H. Murakami and V. Kumar, "Efficient calculation of primary images from a set of images," *IEEE Trans. PAMI*, vol. 4, no. 5, pp. 511–515, Sept. 1982.

[43] S. Chandrasekaran, B. Manjunath, Y. Wang, J. Winkeler, and H. Zhang, "An eigenspace update algorithm for image analysis," *CVGIP: Graphic Models and Image Processing*, vol. 59, no. 5, pp. 321–332, Sept. 1997.

[44] H. Murase and M. Lindenbaum, "Partial eigenvalue decomposition of large images using the spatial temporal adaptive method," *IEEE Trans. Image Processing*, vol. 4, no. 5, pp. 620–629, May 1995.

[45] C. Y. Chang, A. A. Maciejewski, and V. Balakrishnan, "Fast eigenspace decomposition of correlated images," *IEEE Trans. Image Processing*, vol. 9, no. 11, pp. 1937–1949, Nov. 2000.

[46] M. Magnor, P. Ramanathan, and B. Girod, "Multi-view coding for image-based rendering using 3-D scene geometry," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 11, pp. 1092 – 1106, Nov. 2003.

[47] P. Eisert, E. Steinbach, and B. Girod, "Automatic reconstruction of stationary 3-D objects from multiple uncalibrated camera views," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 2, pp. 261 – 277, Mar. 2000.

[48] R. Strzodka, I. Ihrke, and M. Magnor, "A graphics hardware implementation of the generalized hough transform for fast object recognition, scale, and 3D pose detection," in *IEEE Int. Conf. Image Analysis and Processing (ICIAP03)*, Sept. 17 - 19 2003, pp. 188 – 193.

[49] H. Schneiderman and T. Kanade, "A histogram-based method for detection of faces and cars," in *IEEE Int. Conf. Image Processing (ICIP)*, Sept. 10 - 13 2000, pp. 504 – 507.

[50] A. Califano and R. Mohan, "Multidimensional indexing for recognizing visual shapes," *IEEE Trans. PAMI*, vol. 16, no. 4, pp. 373 – 392, Apr. 1994.

[51] Y. Lamdan, J. Schwartz, and H. J. Wolfson, "On recognition of 3-D objects from 2-D images," in *IEEE Int. Conf. Robot. Automat. (ICRA)*, Apr. 24 - 29 1988, pp. 1407 – 1413.

[52] G. H. Golub and C. F. V. Loan, *Matrix Computations.* Baltimore, Maryland: Johns Hopkins, 1996.

[53] K. Saitwal, A. A. Maciejewski, and R. G. Roberts, "A comparison of eigendecomposition for sets of correlated images at different resolutions," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Oct. 27-31 2003, pp. 1011–1017.

[54] J. C. Nash, "A one-sided transformation method for the singular value decomposition and algebraic eigenproblem," *Comp. J.*, vol. 18, no. 1, pp. 74–76, 1975.

[55] P. J. Davis, *Circulant Matrices.* John Wiley and Sons, Inc., 1979.

[56] K. Saitwal, A. A. Maciejewski, and R. G. Roberts, "Analysis of eigendecomposition for sets of correlated images at different resolutions," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, New Orleans, LA, USA, Apr. 26-May 1 2004, pp. 1393–1398.

[57] N. Nguyen, P. Milanfar, and G. Golub, "A computationally efficient superresolution image reconstruction algorithm," *IEEE Trans. Image Processing*, vol. 10, no. 4, pp. 573–583, Apr. 2001.