We approve the thesis of JinHyeong Park.

Date of Signature

______________________________
Rangachar Kasturi
Professor of Computer Science and Engineering
Thesis Co-Adviser
Co-Chair of Committee

______________________________
Hongyuan Zha
Professor of Computer Science and Engineering
Thesis Co-Adviser
Co-Chair of Committee

______________________________
Octavia I. Camps
Associate Professor of Computer Science and Engineering
and Electrical Engineering

______________________________
Richard L. Tutwiler
Senior Research Associate of Applied Research Laboratory

______________________________
Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering
Abstract

Appearance based learning has become very popular in the field of computer vision. In a particular system, a visual datum such as an image is usually treated as a vector by concatenating each row or column. The dimension of the image vector is very high, equal to the number of pixels of the image. When we consider a sequence of images, such video sequences or images capturing an object from different view points, it typically lies on a non-linear dimensional manifold, whose dimension is much lower than that of the original data. When we know the structure of the non-linear manifold, it can be very helpful in the field of computer vision for various applications such as dimensionality reduction, noise handling, etc.

In the first part of this thesis, we propose a method for outlier handling and noise reduction using weighted local linear smoothing for a set of noisy points sampled from a nonlinear manifold. This method can be used in conjunction with various manifold learning methods such as Isomap (Isometric Feature Map), LLE (Local Linear Embedding) and LTSA (Local Tangent Space Alignment) as a preprocessing step to obtain a more accurate reconstruction of the underlying nonlinear manifolds. Using Weighted PCA (Principal Component Analysis) as a foundation, we suggest an iterative weight selection scheme for robust local linear fitting together with an outlier detection method based on minimal spanning trees to further improve robustness. We also develop an efficient and effective bias-reduction method to deal with the “trim the peak and fill the valley” phenomenon in local linear smoothing. Synthetic examples along with several real
image data sets are presented to show that we can combine manifold learning methods with weighted local linear smoothing to produce more accurate results. The proposed local smoothing method has been applied to the image occlusion handling problem and to the noise reduction problem for point-based rendering.

The second part of this thesis focuses on image occlusion handling utilizing manifold learning. We propose an algorithm to handle the problem of image occlusion using the Least Angle Regression (LARS) algorithm. LARS, which was proposed recently in the area of statistics, is known as a less greedy version of the traditional forward model selection algorithm. In other words, the LARS algorithm provides a family of image denoising results from one updated pixel to all of the updated pixels. Using image thresholding and the statistical model selection criterion of Akaike Information Criterion (AIC), we propose a method for selecting an optimal solution among the family of solutions that the LARS algorithm provides. Three sets of experiments were performed. The first measured the stability of the optimal solution estimation method. The second set showed the effects of subblock computation on performance. The last set applied the occlusion handling algorithm to the noisy data cleaning problem, and compared it to two other methods: Orthogonal projection with Weighted PCA and Robust PCA. Experimental results showed that the proposed method yields better performance than the other methods.

In the third part of this thesis, we propose a robust motion segmentation method using the techniques of matrix factorization, subspace separation and spectral graph partitioning. We first show that the shape interaction matrix can be derived using QR decomposition rather than Singular Value Decomposition (SVD) which also leads to a
simple proof of the shape subspace separation theorem. Using the shape interaction matrix, we solve the motion segmentation problems using spectral clustering techniques. We exploit the multi-way Min-Max cut clustering method and provide a novel approach for cluster membership assignment. We further show that we can combine a cluster refinement method based on subspace separation with the graph clustering method which improves its robustness in the presence of noise. The proposed method yields very good performance for both synthetic and real image sequences.
Table of Contents

List of Tables ................................................................. ix

List of Figures ................................................................. x

Acknowledgments ............................................................. xv

Chapter 1. Introduction ..................................................... 1
  1.1 Organization of the Dissertation ..................................... 6

Chapter 2. Local Smoothing for Manifold Learning ..................... 8
  2.1 Nonlinear Dimensionality Reduction using Manifold Learning .... 9
    2.1.1 Isometric Feature Map (Isomap) ............................. 10
    2.1.2 Local Linear Embedding (LLE) ............................ 11
    2.1.3 Curvilinear Distance Analysis (CDA) ....................... 13
    2.1.4 Local Tangent Space Alignment (LTSA) .................... 14
    2.1.5 Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering .................................. 16
  2.2 Weighted PCA ....................................................... 17
  2.3 Selecting Weights for WPCA ........................................ 20
  2.4 Further Improvement for Outlier Handling using MST .............. 23
  2.5 Bias Reduction ....................................................... 27
  2.6 Experimental Results ............................................... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.1</td>
<td>2D Spiral Curve Data</td>
<td>29</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Experiments using 3D Swiss Roll Data</td>
<td>34</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Smoothing 3D Points for Rendering</td>
<td>37</td>
</tr>
<tr>
<td>2.6.4</td>
<td>Face Images and Two Video Sequences (Human Walking and Ballet)</td>
<td>44</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Image Denoising Using LARS Algorithm</td>
<td>49</td>
</tr>
<tr>
<td>3.1</td>
<td>Problem Definition</td>
<td>51</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Image Denoising using $\ell_1$ constraints</td>
<td>56</td>
</tr>
<tr>
<td>3.2</td>
<td>Image Denoising using LARS</td>
<td>59</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Relationship between Image Denoising Objective Function and Lasso</td>
<td>60</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Overview of LARS algorithm</td>
<td>61</td>
</tr>
<tr>
<td>3.3</td>
<td>The Optimal Solution Computation Using Thresholding and AIC</td>
<td>68</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Statistical Model Selection Criteria</td>
<td>68</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Selecting the Optimal Solution Using Thresholding and AIC</td>
<td>69</td>
</tr>
<tr>
<td>3.4</td>
<td>Improvement of the Computation Time</td>
<td>73</td>
</tr>
<tr>
<td>3.5</td>
<td>Sub-Block Computation Using Wavelets decomposition</td>
<td>76</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Basic concept of Haar Wavelet Analysis</td>
<td>76</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Decomposition of Images using Modified Haar Wavelet Analysis</td>
<td>79</td>
</tr>
<tr>
<td>3.6</td>
<td>Experiments</td>
<td>81</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Performance of Optimal Solution Estimation</td>
<td>81</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Performance for Sub-Block Computation</td>
<td>82</td>
</tr>
</tbody>
</table>
3.6.3 Occlusion Handling for Noisy Data .......................... 85

Chapter 4. Spectral Clustering for Robust Motion Segmentation .......... 90

4.1 Matrix Factorization Method for Shape Recovery and Motion Seg-
mentation ........................................................................ 91

4.2 Multibody Motion Segmentation Algorithms .......................... 95

4.3 Constructing the Shape Interaction Matrix Using QR Decomposition 97

4.4 Motion Segmentation ...................................................... 99

4.4.1 Spectral Multi-way clustering ..................................... 99

4.4.2 Refinement of cluster assignment for motion segmentation ... 104

4.5 Experimental Results .................................................... 105

4.5.1 Synthetic Data .......................................................... 105

4.5.2 Real Video Sequences ................................................ 107

Chapter 5. Summary and Conclusions .................................... 110

5.0.3 Contributions .......................................................... 113

5.0.4 Future Works .......................................................... 114

Appendix A. Computation of the Equiangular vector $u_A, w$ and $A_A$ .... 116

Appendix B. 2D Haar Wavelet Analysis .................................. 118

References ................................................................. 120
List of Tables

3.1 Reconstruction Error of the test images from the second row to the seventh row of Fig. 3.11. \( \| \cdot \|_2 \) represents \( \ell_2 \) norm. ............................... 84

3.2 Average reconstruction errors based on \( \ell_2 \) norm and their standard deviation (inside the parenthesis). ......................................................... 87

4.1 An example of the matrix \( \hat{R} \). There are 10 points extracted from 3 objects. The last row shows the assigned cluster ................................. 103

4.2 Misclassification rate (%) for the two synthetic sequences. The values in parentheses are the standard deviation values. **Method 1** is the \( k \)-way Min-Max cut clustering discussed in Section 4.4.1 and **Method 2** is the \( k \)-way Min-Max cut clustering plus the clustering refinement using subspace projection discussed in Section 4.4.2. **Multi-Stage** is the Multi-stage optimization proposed in [33]. ................................. 106

4.3 Misclassification rate (%) for the real video sequences. The values in parentheses are the standard deviation values. ................................. 108
List of Figures

2.1 WPCA results for various $\gamma$'s. Blue line denotes the first principal component direction of the original PCA, and the other color lines denote those of WPCA with different $\gamma$'s (cyan:$\gamma=0.1$, magenta:$\gamma=0.5$ and green:$\gamma=1$). Magenta line is overlapped with green line in (a), and cyan line is overlapped with blue line in (b). “X” mark denotes the weighted center after applying WPCA.  26

2.2 An example of Minimum Spanning Tree (MST)  26

2.3 Shrinking Example. Green dots, red dots and blue dots are the original input data points, the smoothing results at the iteration and the target manifold, respectively. Parameters: $k=30$, $\gamma=0.1$; (a) Original input image. (b)&(c) are the results without bias correction. (d)&(e) are results with bias correction. .  29

2.4 Smoothing results for Gaussian noise data with different $\gamma$'s. Green dots are the input data, and red dots are smoothing results. The first row shows the results without MST outlier handling, and the second row shows those with MST outlier handling. The parameter $k$ is set to 30 and the number of iteration to 10. .  32

2.5 Smoothing results for outlier-overlaid data with different $\gamma$'s. The parameter $k$ is set to 30 and the number of iteration to 10. .  33

2.6 Smoothing results with different $k$s. $\gamma$ is set to 0.1 and the number of iteration to 10. .  33

2.7 Plots of Swiss roll data .  36
2.8 Original Gnome data ......................................................... 39
2.9 Original Santa data .......................................................... 39
2.10 Noisy Gnome data with Gaussian random noise, \( \mathcal{N}(0, 2) \) .......................................... 40
2.11 Smoothing results of noisy Gnome data in Fig. 2.10 ................. 40
2.12 Noisy Gnome data with Gaussian random Noise, \( \mathcal{N}(0, 4) \) .......................................... 41
2.13 Smoothing results of noisy Gnome data in Fig. 2.12 .................. 41
2.14 Noisy Santa data with Gaussian random Noise, \( \mathcal{N}(0, 3) \) .......................................... 42
2.15 Smoothing results of noisy Santa data in Fig. 2.14 ................. 42
2.16 Noisy Santa data with Gaussian random Noise, \( \mathcal{N}(0, 5) \) .......................................... 43
2.17 Smoothing results of noisy Santa data in Fig. 2.16 .................. 43
2.18 Residual variances for each dimension after applying Isomap to face images. (a) shows residual variances for original images without noise. \( k_{\text{Isomap}} \) in (a) stands for the number of neighbors for Isomap. (b)-(e) show the residual variance for occlusion-overlaid cases (green line) and their smoothing results (blue line). \( s \times s, r\% \) means a \( s \times s \) size of noise patch is overlaid to \( r\% \) of the data .......... 46
2.19 (a) shows occlusion-overlaid face images in the size of 20 \( \times \) 20 to 20\% of the data set. (b) shows the smoothing results of (a) ........................................ 46
2.20 Residual variances for each dimension after applying Isomap. (a) shows residual variances for original images without noise. (b)-(e) show the residual variances for occlusion-overlaid cases. ................................................. 47
2.21 (a) shows occlusion-overlaid human walking images in the size of 10 \( \times \) 10 to 40\% of the data set. (b) shows the smoothing results of (a) ....................... 47
2.22 Residual variances for each dimension after applying Isomap. (a) show residual variances for original images without noise. (b)-(e) show the residual variance for noise contamination cases.  
2.23 (a) shows occlusion-overlaid images in the size of 15 x 15 to 30% of the data set. (b) shows the smoothing results of (a).  
3.1 This curve represents a manifold of images including, x̂. The vertical axis and the horizontal axis represent the occlusion pixels and the non-occlusion pixels of the image x respectively.  
3.2 Boat test image and its reconstruction images  
3.3 Fourteen Training boat images  
3.4 Plot of the boat images into 2D space.  
3.5 (a) |α| images computed by applying the LARS algorithm. (b) Binary images of |α|’s. The white pixels represent a non-zero value. (c) Binary images after applying the thresholding algorithm to (a). (d) Reconstruction images by adding α to the occlusion image.  
3.6 (a) The blue line and red dashed line represent the number of foreground pixels and the sum of the absolute values of the foreground pixels, respectively, after applying the thresholding algorithm (b) ℓ2 norm reconstruction error.  
3.7 Haar wavelet decomposition of an image.  
3.8 Modified Haar wavelet decomposition  
3.9 Occlusion images in the first row and their reconstruction Results in the second row.
3.10 The first row plots $|F_i G_i|$’s and the second row plots their reconstruction error through the iteration of the 5 test images in Fig. 3.9. The red dot indicates the position of the selected optimal solution.

3.11 Image denoising results of face images. (a) Original images. (b) Occlusion overlaid images of (a). (c) PCA reconstruction (d) LARS reconstruction without Haar wavelet decomposition. (e) $\sim$ (g) LARS reconstruction by applying the proposed Haar wavelet decomposition of level 1 (4 subbands, size of $32 \times 32$), level 2 (16 subbands, size of $16 \times 16$) and level 3 (64 subbands, size of $8 \times 8$), respectively.

3.12 Selected reconstruction results of the face data, the human walking video sequence, and the ballet video sequence. (a) 10 selected occlusion images (b) Reconstruction results of Proposed LARS algorithm, (c) Reconstruction results of the Local Smoothing method proposed in [45], (d): Reconstruction results of Robust PCA [15].

3.13 (a) 20 selected occlusion overlaid images from video sequence. (b) LARS reconstruction. (c) Absolute images between (a) and (b). (d) RPCA reconstruction. (e) Absolute images between (a) and (d).

4.1 The two synthetic video sequences used in [32] and [33] respectively. (a). 20 green dots are background pixels and 9 foreground points (b). 24 background points and 14 foreground points. The foreground pixels are connected with lines.
4.2 Real video sequences with the feature points. 1\textsuperscript{st} row: \textit{video1}, 2\textsuperscript{nd} row: \textit{video2}, 3\textsuperscript{rd} row: \textit{video3}, 4\textsuperscript{th} row: \textit{video4} (foreground feature points in \textit{video1} are overlaid onto \textit{video2}). Red dots correspond to the background while green dots correspond to the foreground. The yellow cross marks in \textit{video4} represent the foreground feature points of \textit{video1}.

4.3 Graph for misclassification rate. Graph of \textit{video1} is not depicted here because all of the methods perform perfectly. \textbf{Method 1}: Dashed-dot blue line, \textbf{Method 2}: Red line and \textbf{Multi-Stage}: Dashed green line.
Acknowledgments

First of all, I praise and give my thank from deep inside my heart to God who has guided me through my study. He listened to my cry and he was always with me whenever I was wandering in the desert without hope. When I fall on the ground, he raised me. When I couldn’t feel any hope he blew a breeze of help to me. Thank you, God.

I would like to express my heartfelt and sincere thanks to my co-advisors and the co-chairman of my thesis committee, Prof. Rangachar Kasturi and Prof. Hongyuan Zha, for their guidance and support over the course of my research. I also appreciate the other committee members, Dr. Octavia I. Camps and Dr. Richard L. Tuwiler, their useful suggestions and valuable comments on the thesis. I would like to thank Prof. Jesse L. Barlow and Dr. Robert T Collins for their encouragement and interest in my work.

My thanks also goes to my colleagues in Computer Vision group and other former graduate students in the group. It was thank the Korean graduate students in my departments for their concern and encouragement. I would like to thank the members of PSU Korean Soccer Club. My life in State College was enjoyable because I could play soccer with great friends every Saturday. It was very helpful to keep my body strong enough to pursue my research.

I would like to thank Pastor Sang-kee Eun and Pastor Ju-Young Lee at the State College Korean Church for teaching and guiding me to know the love of the Christ Jesus. I also thank all the brothers and sisters in the church who encouraged me whenever I
was depressed, and prayed for me whenever I need help. My thanks also goes to Pastor David Alas for his help in editing my thesis.

My thank also goes to my mother-in-law and father-in-law for there love and support. Especially, I appreciate my mother-in-law, Bokja Jeon, who always prayed for my family and my research. I would like thank my father Chung-Su Park and my mother Guja Han for their unconditional love and support. I am very proud of being your the first son. You are the best father and the best mother on earth.

Finally, I truly thank my wife, Jayeoun Lim, for her love and support, and her patience during the period of time it took me to pursue this research. My son, Daniel Jaemin Park, it was my best time on earth when you were born, June 6th, 2002. You, my beautiful wife and adorable son, are the best motivation of my study.
Chapter 1

Introduction

Manifold learning represents a novel set of methods for nonlinear dimension reduction emphasizing simple algorithmic implementation and avoiding optimization problems prone to local minima [59, 50, 69]. It has applications in several areas of computer vision such as image interpolation and tracking. At the pixel level, a set of images such as a collection of faces can be represented as high-dimensional vectors, but their intrinsic dimension is usually much smaller than the ambient feature space. Capturing this intrinsic dimension from a set of samples is of paramount importance in computer vision applications. This is evident given the increasing interest in subspace-based methods such as Principal Component Analysis (PCA) [30], Independent Component Analysis (ICA) [27], and Non-linear Matrix Factorization (NMF) [36], just to name a few. Furthermore, these generally linear methods have recently been joined by several non-linear dimensionality reduction methods such as Isomap (Isometric feature mapping) [59, 58], LLE (Local Linear Embedding) [50] and LTSA (Local Tangent Space Alignment) [69]. One chief advantage of the newer methods is that they can successfully compute dimension reduction for data points sampled from a nonlinear manifold.

In many real-world applications with high-dimensional data, the components of the data points tend to be correlated with each other, and in many cases the data points
lie close to a low-dimensional nonlinear manifold. Discovering the structure of the manifold from a set of data points possibly with noise represents a very challenging unsupervised learning problem [50, 18, 51, 59]. But once discovered, these low-dimensional structures can be used for classification, clustering, and data visualization. Traditional dimensionality reduction techniques such as principal component analysis (PCA), factor analysis and multidimensional scaling usually work well when the underlying manifold is a linear (affine) subspace [26]. However, they can not, in general, discover nonlinear structures embedded in the set of data points.

Recently, there has been a growing interest in developing efficient algorithms for constructing nonlinear low-dimensional manifolds from sample data points in high-dimensional spaces emphasizing simple algorithmic implementation and avoiding optimization problems prone to local minima [50, 59]. Two main approaches are the Isomap methods [18, 59] and the local linear embedding methods (LLE and LTSA) [50, 69].

In the first method, pairwise geodesic distances of the data points with respect to the underlying manifold are estimated, and classical multi-dimensional scaling is used to project the data points into a low-dimensional space that best preserves these geodesic distances. The latter method follows a long tradition, starting with self-organizing maps (SOM) [34], principal curves/surfaces [25] and topology-preserving networks [41, 8]. The key idea is that the information about the global structure of a nonlinear manifold can be obtained from carefully analyzing the interactions of the overlapping local structures. In particular, the local linear embedding (LLE) method constructs a local geometric structure that is invariant to translations and orthogonal transformations in a neighborhood of each data point, and seeks to project each data point into a low-dimensional space that
best preserves these local geometries \[50, 51\]. LTSA\[69\] uses the tangent space in the neighborhood of each data point to represent the local geometry, and then aligns those local tangent spaces to construct a global coordinate system for the nonlinear manifold. Similar to PCA, however, these manifold learning methods are still quite sensitive to outliers. The focus of this research is the development of an outlier handling and noise reduction method that can be used with those nonlinear methods as a preprocessing procedure to obtain a more accurate reconstruction of the underlying nonlinear manifold. More importantly, outlier handling and noise reduction will reduce the chances of short-circuit nearest neighbor connections and therefore help to better preserve the topological/geometric structures of the manifold in a neighborhood graph constructed from a finite set of noisy sample points.

Computer vision has recently received a lot of attention due to the significant increase of visual data, as well as the improvement of computer processing power and memory capacity. Computer vision tasks such as face recognition or human activity recognition are very challenging when the visual data is corrupted by noise. Hence image denoising is an important pre-processing step in computer vision applications and has been studied extensively \[9\] \[15\] \[7\] \[43\]. The usual approach to image denoising is to project the image orthogonally onto the subspace (simple smoothing) computed using a training data set. When we apply the proposed local smoothing method to image occlusion handling, it also computes the local tangent space of an given occlusion image and projects the image orthogonally onto the local tangent space. This approach has several shortcomings especially when only a part of the image is corrupted, since all the pixel values are changed after orthogonally projecting the noisy image onto the
subspace. Tsuda et al. [63] proposed an image denoising method to identify the noisy pixels by $\ell_1$ norm penalization and to update only the identified noisy pixels. It is well known that $\ell_1$ norm penalization yields a sparse solution. Therefore, the main purpose of using $\ell_1$ norm penalization in [63] is to select only noisy pixels. They construct a Linear Programming (LP) problem with $\ell_1$ constraints to identify and update the noisy pixels simultaneously.

This approach has a significant advantage over the other methods in terms of reconstruction error since it only updates a small part of the image pixels without touching the other pixels. However, this method needs a parameter to control the fraction of the updated pixels. The higher the value of this parameter, the greater the number of pixels that are updated. We propose a new image occlusion handling method using the Least Angle Regression (LARS) [19] algorithm. LARS was originally proposed as a feature selection method for the regression problem, which claims to be a less greedy version of the original forward feature selection method. LARS is based on the iteration and provides all the solutions (feature selection) by adding one feature at each iteration at a cheap cost in computation time.

To this end, we first show that the image occlusion handling problem can be converted into a regression problem. In this new regression problem, each covariate corresponds to a pixel. Secondly, we apply the LARS algorithm to the problem to obtain all the solutions (from one updated pixel to all of the updated pixels). Actually, the $\ell_1$ constraints which were used in [63] are not necessary when we use LARS algorithm because the purpose of the $\ell_1$ constraint is to compute a sparse solution, and LARS provides all
the possible sparse solutions. Finally, we select the optimal solution from among the family of solutions that the LARS algorithm yields using an image thresholding technique and a statistical model selection method. Thus, the proposed image occlusion handling method would need no parameter. We also discuss sub-block computation using Wavelet decomposition to further improve the computation time.

Afterwards, we discuss motion segmentation using matrix factorization. Matrix factorization methods have been widely used for solving the motion segmentation problems [21] [28] [29] [64] [31] [32] [33] [66] and 3D shape recovering problems [12] [62] [11]. The factorization method proposed by Tomasi and Kanade [61] was for the case of a single static object viewed by a moving camera. The basic idea of this method was to factorize the feature trajectory matrix into the motion matrix and the shape matrix, providing the separation of the feature point trajectories into independent motions. Singular Value Decomposition (SVD) is used for this factorization.

Costerira and Kanade proposed a multi-body factorization method for independent moving objects that builds upon the previous factorization method [14]. The shape interaction matrix was proposed for grouping feature points into independent objects.

Given a set of \( N \) feature points tracked through \( F \) frames, we can construct a feature trajectory matrix \( \mathbf{P} \in \mathbb{R}^{2F \times N} \) where the rows correspond to the \( x \) or \( y \) coordinates of the feature points in the image plane and the columns correspond to the individual feature points. Motion segmentation algorithms based on matrix factorization [31] first construct a shape interaction matrix \( \mathbf{Q} \) by applying the singular value decomposition (SVD) to the feature trajectory matrix \( \mathbf{P} \). Under the noise-free situation, the shape interaction matrix \( \mathbf{Q} \) can be transformed into a block diagonal matrix by a symmetric
row and column permutation thereby grouping the feature points of the same object into a diagonal block.

If the trajectory matrix $P$ is contaminated by noise, however, the block diagonal form of $Q$ no longer holds, and the methods such as the greedy technique proposed in [14] tend to perform rather poorly. Recently there has been much research proposed that specifically addresses this problem [32] [64] [21] [28] [64] [31] [66].

We have developed a novel robust factorization method using the techniques of spectral clustering. We mathematically show that the shape interaction matrix can be constructed by applying QR decomposition with pivoting instead of SVD. It is widely known that QR decomposition is more stable and efficient than SVD. We applied the spectral graph partitioning algorithm to the shape interaction matrix for clustering the feature points into independent moving objects.

1.1 Organization of the Dissertation

Chapter 2 deals with a noise handling method for manifold learning. WPCA and the weight selection are discussed in Section 2.2 and Section 2.3 respectively. The MST-based outlier handling follows in Section 2.4. We propose a new bias correction method in Section 2.5. Experimental results are discussed in Section 2.6.

Chapter 3 examines a method for image occlusion handling using the LARS algorithm. We define the problem in Section 3.1, and the LARS algorithm is summarized in Section 3.2.2. Optimal solution computation using a thresholding technique and a
statistical model selection criterion is proposed in Section 3.3. Improvement of computation time is discussed in Section 3.4. Sub-block computation using Haar Wavelet decomposition is proposed in 3.5. Experimental results are shown in Section 3.6.

Chapter 4 discusses motion segmentation using matrix factorization and spectral graph partitioning. Previous work in the area of multi-body motion segmentation is discussed in Section 4.2. Section 4.3 is devoted to a proof showing that the shape interaction matrix can be computed using QR decomposition. Motion segmentation based on spectral relaxation k-way clustering and subspace separation is described in Section 4.4. Experiment results are shown in Section 4.5.

Chapter 5 provides the summary of the dissertation and discusses future work.
Chapter 2

Local Smoothing for Manifold Learning

In this chapter, we focus on noise reduction and outlier handling by exploiting the basic idea of weighted local linear smoothing. The techniques we used are similar in spirit to local polynomial smoothing employed in non-parametric regression [38]. However, since in our context, we do not have the response variables, local smoothing needs to employ techniques other than least squares fitting. Since we are interested in modeling high-dimensional data, and therefore in local smoothing we try to avoid more expensive operations such as approximating the Hessian matrices when we carry out bias reduction. Furthermore, we apply local smoothing in an iterative fashion to further improve accuracy.

We assume that $\mathcal{F}$ is a $d$-dimensional manifold in an $m$-dimensional space with unknown generating function $f(\tau), \tau \in \mathcal{R}^d$, and we are given a data set of $N$ vectors $x_i \in \mathcal{R}^m, i = 1, 2, \ldots, N$, generated from the following model,

$$x_i = f(\tau_i) + \epsilon_i, \quad i = 1, \ldots, N, \tag{2.1}$$

where $\tau_i \in \mathcal{R}^d$ with $d < m$, and $\epsilon_i$'s represent noise. The goals of nonlinear manifold learning are to 1) construct the $\tau_i$'s from the $x_i$'s; and 2) construct an approximation of the generating function $f(\tau)$ [69]. In the local smoothing method we will discuss, the nonlinear manifold $\mathcal{F}$ is locally approximated by an affine subspace. Before applying
a manifold learning method, we propose to carry out a local smoothing procedure as follows. For each sample point \( x_i, i = 1, 2, \ldots, N \), we compute its \( k \) nearest neighbor sample points. The sample point \( x_i \) is then examined to see if it is located in the middle of two or more patches of the manifold using outlier detection based on a Minimum Spanning Tree (MST). If it is we move \( x_i \) to one of the patches, otherwise we compute an affine subspace using Weighted PCA (WPCA) from the \( k \) nearest neighbors, and project the \( x_i \) to this affine subspace. After projecting (or moving) all the sample points, we correct their bias, and then the above steps are iterated several times. Experiments were conducted using synthetic two dimensional (2D) data sets and three dimensional (3D) data sets. The proposed local smoothing method was also applied to point-based rendering problem in the field of Computer Graphics, and to an image occlusion handling problem.

2.1 Nonlinear Dimensionality Reduction using Manifold Learning

In this section, we review five important nonlinear dimensionality algorithms: Isomap, LLE, CDA, LTSA and Laplacian Eigenmaps. Let us assume that the \( d \)-dimensional manifold \( \mathcal{F} \) embedded in an \( m \)-dimensional space \((d < m)\) can be represented by a vector-valued multivariate function

\[
f : C \subset \mathbb{R}^d \to \mathbb{R}^m,
\]

where \( C \) is a compact subset of \( \mathbb{R}^d \) with open interior. We are given a set of data points \( x_1, \cdots, x_N \) with \( x_i \in \mathbb{R}^m \), which are sampled possibly with noise from the manifold, i.e.,
\[ x_i = f(y_i) + \epsilon_i, \quad i = 1, \ldots, N, \quad (2.2) \]

where \( \epsilon_i \) represents noise. By \textit{dimension reduction} we mean the estimation of the unknown lower dimensional feature vectors \( y_i \)'s from the \( x_i \)'s, realizing the objective of (nonlinear) dimensionality reduction of the data points.

### 2.1.1 Isometric Feature Map (Isomap)

Isometric Feature Map (Isomap) was a novel nonlinear dimensionality reduction method proposed by Tenenbaum et. al. [59] which finds the subspace that preserves the geodesic interpoint distances. This algorithm has three steps. The first step is to construct a neighborhood graph by determining the neighborhood points of each input point on the manifold \( M \), based on the distances \( d_X(i, j) \) between two pairs of points, \( x_i \) and \( x_j \) in the input space \( X \). Two simple methods are considered to determine the criterion of neighborhood: k-nearest neighbors and points within some fixed radius \( \epsilon \). These neighborhood relations are represented as a weighted graph \( G \) over the data points, with edges of weight \( d_X(i, j) \) between neighboring points.

In the second step, the algorithm estimates the geodesic distance \( d_M(i, j) \) between all pairs of points on the manifold \( M \) by computing their shortest path distance \( d_G(i, j) \). A simple method to compute the distance \( d_G(i, j) \) can be summarized as follows.

- **Initialization**

\[
d_G(i, j) = \begin{cases} 
    d_X(i, j) & \text{if } i, j \text{ are linked by an edge in } G \\
    \infty & \text{otherwise}
\end{cases}
\]
• Update $d_G(i, j)$ (Floyd’s algorithm)

$$d_G(i, j) = \min d_G(i, j), d_G(i, k) + d_G(k, j), k = 1, 2, \ldots, N.$$ 

Re-iterate this update procedure until no further updates occur.

The final step applies classical Multidimensional Scaling (MDS) to the matrix of graph distances $d_G(i, j)$ by constructing an embedding of the data in a $d$-dimensional Euclidean space $Y$ that best preserves the manifold’s estimated intrinsic geometry.

A continuous version of Isomap, called Continuum Isomap, was proposed in [68]. Continuum Isomap computes a set of eigenfunctions that forms the canonical coordinates of the Euclidean space up to a rigid motion for a nonlinear manifold that can be isometrically embedded onto the Euclidean space. This manifold learning in the continuous framework is reduced to an eigenvalue problem of an integral operator.

### 2.1.2 Local Linear Embedding (LLE)

Local Linear Embedding (LLE) was proposed by Roweis et. el. [50] [53]. LLE is an unsupervised learning algorithm that computes low-dimensional, neighborhood-preserving embedding of high-dimensional input data. LLE maps its input data into a single global coordinate system of lower dimensionality, and its optimizations do not involve local minima. The LLE algorithm can be summarized as follows.
Suppose that the data consists of $N$ real vectors $\text{vec}X_i \in \mathbb{R}^d$, sampled from some underlying manifold. The first step computes the weights $W_{is}$ to minimize reconstruction error specified in Equation 2.3.

$$E(W) = \sum_i^n \|x_i - \sum_{i=1}^k w_{ij}^{(i)} x_{N(j)}\|^2$$

(2.3)

where $x_{N(1)}, \ldots, x_{N(k)}$ are the neighborhood points of $x_i$, and $\| \cdot \|_2$ represents $\ell_2$ norm. The function reflects how well each $x_i$ can be linearly reconstructed in terms of its local neighbors. To compute the weights $W_{ij}$ the cost function should be minimized subject to two constraints. First, each data point $x_i$ is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if $x_i$ does not belong to the set of neighbors of $X_i$. Second, the sum of the weights associated with each point should be one, i.e., $\sum_j W_{ij} = 1$. In particular, the same weights $W_{ij}$ that reconstruct the $i$-th data point in $D$ dimensions should also reconstruct its embedded manifold coordinates in $d$ dimensions, where $d < D$.

In the second step, LLE constructs a neighborhood-preserving mapping based on the above idea.

$$\Phi(Y) = \sum_i^n \|y_i - \sum_{i=1}^k w_{ij}^{(i)} Y_{N(j)}\|^2$$

(2.4)

This cost function, similar to Eq. 2.3, is based on locally linear reconstruction errors. The function estimates local coordinates, $y_1, \cdots, y_n$ in $d$ dimensional manifold space by fixing the weights computed in Eq. 2.3.
2.1.3 Curvilinear Distance Analysis (CDA)

The CDA [37] shares the same metrics with Isomap. The only difference is that Isomap exploits the traditional MDS for projection from \( d \)-dimensional space (original data space) to \( p \)-dimensional space (projection space), while CDA works with neural methods derived from the Curvilinear Component Analysis (CCA) [16].

The error function of CDA is written as:

\[
E_{CDA} = \sum_{i=1}^{n} \sum_{j=1}^{n} (\delta_{ij}^d - d_{ij}^p)^2 F(d_{ij}^p)
\]

where \( \delta_{ij}^p \) and \( d_{ij}^p \) are respectively the curvilinear distance in the data space and the Euclidean distance in the projection space between the \( i \) – \( th \) and \( j \) – \( th \) points. The factor \( F(d_{ij}^p, \lambda_d) \) is generally chosen as a bounded and monotonically decreasing function in order to favor local topology conservation. Decreasing exponential, sigmoid or Lorentz functions are all suitable choices. A simple step function shown in Eq. 2.6 can also be used for the factor \( F(d_{ij}^p, \lambda_d) \).

\[
F(d_{ij}^p) = \begin{cases} 
1 & \text{if } d_{ij}^p \leq \lambda_d \\
0 & \text{otherwise}
\end{cases}
\]

The algorithm of CDA can be summarized in five steps:

1. Apply vector quantization on the raw data.
2. Compute \( k \)- or \( \epsilon \)-neighborhoods and link neighboring prototypes.
3. Compute distances, \( D \), using Dijkstra’s algorithm.
4. Optimize $E_{CDA}$ by stochastic gradient descent, in order to get coordinates for the prototypes in the projection space.

5. Run a piecewise linear interpolator to compute the projection of original data points (this step is not necessary if step 1 was skipped).

From a theoretical point of view, there are two main differences between CDA and Isomap. The first is the way they determine landmark points (or prototypes). The second is the way they compute low-dimensional coordinates. The Isomap exploits classical MDS, whereas the CDA optimizes the energy function in Eq. 2.5 using gradient descent or stochastic gradient descent. The Isomap shows stronger mathematical foundation, and yields the coordinates simultaneously. The CDA is based on the neural networks and can converge to local minima based on the initial parameters, $F$ and learning rate. However, if the two parameters are well adjusted, CDA can find useful projections. It can project not only the training data but also new data using interpolation method.

\subsection*{2.1.4 Local Tangent Space Alignment (LTSA)}

LTSA was proposed by Zhang and Zha [69]. The basic idea is to use the tangent space in the neighborhood of a datum point to represent the local geometry, and then align those local tangent spaces to construct the global coordinate system for the nonlinear manifold. The local tangent space provides a low-dimensional linear approximation of the local geometric structure of the nonlinear manifold. It preserves the local coordinates of the data points in the neighborhood with respect to the tangent space. Those local tangent coordinates will be aligned in the low dimensional space by different local
affine transformations to obtain a global coordinate system. Given $N$ $m$-dimensional sample points sampled possibly with noise from an underlying $d$ dimensional manifold, it produces $N$ $d$-dimensional coordinates $T \in \mathbb{R}^{dxN}$ for the manifold constructed from $k$ local nearest neighbors.

Let $X_i = [x_{i1}, \cdots, x_{ik}]$ be a matrix consisting of its $k$-nearest neighbors including $x_i$. First, compute the best $d$-dimensional affine subspace approximation for the data points in $X_i$.

$$\min_{x, \Theta, Q} \sum_{j=1}^{k} \|x_{ij} - (x + Q\theta_j)\|_2 = \min_{x, \Theta, Q} \|X_i - (xe^T + Q\Theta)\|_2$$

where $Q$ is of $d$ columns and is orthonormal, and $\Theta = [\theta_1, \ldots, \theta_k]$.

Next, compute the $d$-dimensional global coordinates $y_1, \cdots, y_N$ for the local coordinates of $\theta a_1, \ldots, \theta k$. Actually, $y_1, \cdots, y_N$ is the target coordinates for $x_1, \cdots, x_N$.

To preserve as much of the local geometry in the low-dimensional feature space, we seek to find $y_i$ and the local affine transformations $L_i$ to minimize the reconstruction errors $\epsilon^{(i)}_j$, i.e.,

$$\sum_{i} \|Y_i(I - \frac{1}{k}ee^T) - L_i\Theta_i\|_2,$$  \hspace{1cm} (2.7)

where $Y_i = [y_{i1}, \cdots, y_{ik}]$. 
2.1.5 Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering

The method proposed in [6] [5] is based on the fact that Laplacian of a graph obtained from data points may be viewed as an approximation to the Laplace-Beltrami operator defined on the manifold. It has strong connections to the spectral clustering algorithm developed in machine learning and computer vision [55] [23] [67].

Given $N$ input points $x_{i1}^N \in \mathbb{R}^m$, we can construct a weighted graph with $N$ nodes, one for each point, and the set of edge connecting neighboring points to each other. The algorithm is summarized as follows:

1. The first step is to put an edge between nodes $i$ and $j$ if $x_i$ and $x_j$ are close to each other. We can simply think about “closeness” in two ways: $k$ nearest neighbors and $\epsilon$-distance neighborhoods.

2. The second step is to determine the weights of the edges using one of two methods.

   The first method is to simply assign $W_{ij} = 1$ if and only if vertices $i$ and $j$ are connected by an edge. Or, alternatively, you can use the heat kernel equation below.

   $$ W_{ij} = \exp\left( -\frac{\|x_i - x_j\|^2}{\sigma} \right) $$

3. Then, by computing $d$ eigenvectors $y_1 \cdots y_d$ corresponding to the $d$ smallest eigenvalues, whose eigenvalues are greater than zero, we can address the eigenvector problem $Ly = \lambda Dy$ where $D = \text{diag}(d_1, \cdots, d_N)$, $d_i = \sum_{j=1}^N W_{ij}$, $L =$
$D - W$ which is the Laplacian matrix. The $y_1 \cdots y_d$ are the orthogonal basis of $d$-dimensional manifold space.

2.2 Weighted PCA

A number of robust methods exists in the literature to deal with outlier problems, especially in the statistics community: M-estimators [20], Least Median of Squares [49] and so on. These methods were applied to computer vision problems, and a number of variations were published to compute robust linear subspaces despite the presence of outliers [65, 15, 56]. The weighted PCA (WPCA) we are presenting is similar to the ideas of robust M-estimation. Since we only consider object-level outliers, our version of WPCA has a closed-form solution. With both object-level and feature-level outliers, the WPCA problem requires an optimization problem to be solved in an iterative fashion, see [15] for example. To this end, for each sample point $x_i$ in the given data set, let $X_i = [x_{i1}, \ldots, x_{ik}]$ be a matrix consisting of its $k$-nearest neighbors in terms of the Euclidean distance. We can fit a $d$-dimensional affine subspace to the columns in $X_i$ by applying PCA to $X_i$. Since one of the goals of local smoothing is to reduce the effects of outliers, we need to develop more robust version of PCA. Our basic idea is to incorporate weighting to obtain a weighted version of PCA (WPCA).

Now, in the above example, we want to fit these $k$-nearest neighbors using an affine subspace parameterized as $c + Ut$, where $U \in \mathbb{R}^{m \times d}$ forms the orthonormal basis of the affine subspace, $c \in \mathbb{R}^m$ gives the displacement of the affine subspace, and $t \in \mathbb{R}^d$ represents the local coordinate of a vector in the affine subspace. To this end, we consider
the following weighted least squares problem:

$$\min_{c, \{t_i\}, U^T U = I_d} \sum_{i=1}^{k} w_i \|x_i - (c + Ut_i)\|_2.$$  \tag{2.8}$$

where $w_1, \ldots, w_k$ are a given set of weights which will be discussed in the next section. We denote $X = [x_1, \ldots, x_k]$ and $T = [t_1, \ldots, t_k]$. The weight vector and the corresponding diagonal matrix are defined, respectively, as

$$w = [w_1, \ldots, w_k]^T, \quad D = \text{diag}(\sqrt{w_1}, \ldots, \sqrt{w_k})$$

The following theorem characterizes the optimal solution to the weighted least squares problem in (2.8)

**Theorem 2.1.** Let $\bar{x}_w = \sum_i w_i x_i / \sum_i w_i$ be the weighted mean of $x_1, \ldots, x_k$, and let $u_1, \ldots, u_k$ be the largest left singular vectors of $(X - \bar{x}_w e^T)D$, where $e$ is the $k$-dimensional column vector of all ones. Then an optimal solution of the problem (2.8) is given by

$$c = \bar{x}_w, \quad U = [u_1, \ldots, u_k], \quad t_i = U^T (x_i - \bar{x}_w).$$

**Proof.** Let

$$E = (X - (ce^T + UT))D$$

be the weighted reconstruction error matrix. Using $v$ to denote the normalized vector of $w$

$$\sqrt{w} = [\sqrt{w_1}, \ldots, \sqrt{w_k}]^T, \quad v = \sqrt{w} / \|\sqrt{w}\|_2,$$
we rewrite the error matrix $E$ as

$$E = Evv^T + E(I - vv^T).$$

Because $Dvv^T = we^TD/(e^Tw)$, we have that

$$Evv^T = (X - (ce^T + UT))Dvv^T$$
$$= ((\bar{x}_w - c)e^T - UTwe^T/(e^Tw))D$$

and

$$E(I - vv^T) = (X - (ce^T + UT))D - Evv^T$$
$$= (X - (\bar{x}_we^T + UT))D$$

with $\tilde{T} = T(I - we^T/(e^Tw))$. Clearly, $\tilde{E} = E(I - vv^T)$ is the reconstructed error matrix corresponding to the feasible solution $(\bar{x}_w, \tilde{T})$ to (2.8). Since $\|\tilde{E}\|_F \leq \|E\|_F$ and $\tilde{E}\sqrt{w} = 0$, we can conclude that an optimal solution of (2.8) should have an error matrix $E$ satisfying $E\sqrt{w} = 0$. With this condition, we have

$$Xw = XD\sqrt{w} = (ce^T + UT)D\sqrt{w} = (ce^T + UT)w.$$  

It follows that

$$c = \bar{x}_w - U\alpha, \quad \alpha = Tw/e^Tw.$$
\[ X - (ce^T + UT) = X - (\bar{x}w + U(T - \alpha e^T)). \]

With an abuse of notation, denoting \((T - \alpha e^T)\) by \(T\), the optimization problem (2.8) reduces to

\[
\min \left\{ t_i \bigg| U^T U = I_d, (X - \bar{x}w e^T)D - UTD \right\}_F.
\]

The optimal solution, as given by the Singular Value Decomposition (SVD) of matrix \((X - \bar{x}w e^T)D\), is

\[
U = U_d, \quad TD = U_d^T (X - \bar{x}w e^T)D.
\]

It follows that \(T = U_d^T (X - \bar{x}w e^T)\), completing the proof.  

Thus, the sample point \(x_i\) is projected to \(x_i^*\) as

\[
x_i^* = \bar{x}w + UU^T (x_i - \bar{x}w).
\]

This projection process is done for each of the sample points in \(\{x_1, \ldots, x_N\}\).

2.3 Selecting Weights for WPCA

In this section, we consider how to select the weights to be used in WPCA. Since the objective of introducing the weights is to reduce the influence of the outliers as much as possible when fitting the points to an affine subspace, ideally the weights should be
chosen such that $w_i$ is small if $x_i$ is considered as an outlier. Specifically, we let the weight $w_i$ be inversely proportional to the distance between $x_i$ and an ideal center $x^*$. Here $x^*$ is defined as the mean of a subset of the points in which outliers have been removed. The ideal center $x^*$, however, is unknown. We will use an iterative algorithm to approximate $x^*$ starting with the mean of the sample points.

To begin, we consider weights based on the isotropic Gaussian density function defined as

$$w_i = c_0 \exp(-\gamma \|x_i - \bar{x}\|_2),$$  \hspace{1cm} (2.9)

where $\gamma > 0$ is a constant, $\bar{x}$ an approximation of $x^*$, and $c_0$ the normalization constant such that $\sum_{i=1}^{n} w_i = 1$. Other types of weights discussed in [38] can also be used.

Our iterative weight selection procedure for a given set of sample points $x_{i1}, \ldots, x_{ik}$ is structured as follows: Choose the initial vector $\bar{x}_{w(0)}$ as the mean of the $k$ vectors $x_{i1}, \ldots, x_{ik}$, and iterate until convergence using the following:

1. Compute the current normalized weights, $w_i^{(j)} = c_j \exp(-\gamma \|x_i - \bar{x}_{w(j-1)}\|_2)$.
2. Compute a new weighted center $\bar{x}_{w(j)} = \sum_{i=1}^{k} w_i^{(j)} x_i$.

Initially, we can choose $\bar{x}$ as the mean of all the $k$ sample points $\{x_i\}$ as an approximation to the ideal center $x^*$. The existence of outliers can render the mean $\bar{x}$ quite far away from $x^*$. Then we update $\bar{x}$ by the weighted mean $\bar{x}_w = \sum w_i x_i$, using the current set of weights given by (2.9), and compute a new set of weights by replacing
\( \bar{x} \) by \( \bar{x}_w \) in (2.9). The above process can be carried out in several iterations. We now present an informal analysis to illustrate why the iterative weight selection process can be effective at down-weighting the outliers. Consider the simple case where the data set of \( k + \ell \) samples has several outliers \( x_1, \ldots, x_\ell \) that are far away from the remaining set \( x_{\ell+1}, \ldots, x_{\ell+k} \) which are relatively close to each other. We also assume \( \ell \ll k \).

Denote \( \bar{x}_0 \) the mean of all \( \ell + k \) sample points and \( \bar{x} \) the mean of the sample points \( x_{\ell+1}, \ldots, x_{\ell+k} \). It is easy to see that

\[
\bar{x}_0 = \frac{1}{\ell + k} \left( \sum_{i=1}^{\ell} x_i + k \bar{x} \right) = \bar{x} + \delta, \quad \delta = \frac{1}{\ell + k} \sum_{i=1}^{\ell} (x_i - \bar{x}).
\]

If \( \|\delta\| \gg \|x_j - \bar{x}\| \) for \( j = \ell + 1, \ldots, \ell + k \), we have \( \|x_j - \bar{x}_0\|_2 \approx \|\delta\|_2 \). Therefore, for \( j = \ell+1, \ldots, \ell+k \), we have \( w_j = c_0 \exp(-\gamma\|x_j - \bar{x}_0\|_2) \approx c_0 \exp(-\gamma\|\delta\|_2) \).

On the other hand, since \( \|x_i - \bar{x}\|_2 \gg \|\delta\|_2 \) for \( i \leq \ell \), we have \( w_i = c_0 \exp(-\gamma\|x_i - \bar{x}_0\|_2) \approx 0 \). If \( \|\delta\| \) is not large compared with the distances between \( \bar{x} \) and the clustered points \( x_{\ell+1}, \ldots, x_{\ell+k} \), we also have \( \|x_i - \bar{x}_0\| \approx \|x_i - \bar{x}\| \gg \|x_j - \bar{x}\| \approx \|x_j - \bar{x}_0\| \) for \( i \leq \ell < j \). This implies \( w_i \approx 0 \) and \( w_j \approx \text{constant} \). Therefore, recalling that the weight set is normalized such that \( \sum w_i = 1 \), we conclude that \( w_0 \approx 0 \), \( i \leq \ell \), \( w_i \approx \frac{1}{k} \), \( i > \ell \). Therefore, \( \bar{x}_w \approx \bar{x} \) and the updated weights \( w_i \) have the effect of down-weighting the outliers.

We have not carried out a formal analysis of the above iterative process. Based on several simulations, the sequence \( \{\bar{x}_w^{(j)}\} \), in general, seems to converge fairly fast.
2.4 Further Improvement for Outlier Handling using MST

In most situations, WPCA with the above weight selection scheme can handle noise and outliers quite well. However, it is not easy for WPCA to handle outliers located between two or more patches of a manifold as shown in Fig. 2.1-(b).

Fig. 2.1 depicts two examples of WPCA for various $\gamma$'s. In case (a), we present a set of outliers under which WPCA performs quite well. The weights of the outliers become close to zero when we increase the value of $\gamma$ and the first principal axis is computed correctly using WPCA (the green line or the red line). In Fig. 2.1-(b), the weight selection method yields non-negligible weights for the two outliers lying between the two patches of a manifold and almost zero for weights associated with the other points of the patches for a large value of $\gamma$. The first principal axis becomes the green line with $\gamma = 1$ see Fig. 2.1-(b).

We will now show that we can improve the performance of local smoothing further by using Minimum Spanning Tree (MST). The key observation is that if a point $x_i$ is an outlier as shown in Fig. 2.1-(b), the distance between the outlier point and the nearest patch is much greater than the distances between two near-by points in a patch.

A spanning tree for a connected undirected weighted graph $G$ is defined as a sub-graph of $G$ that is an undirected tree and contains all the vertices of $G$ [4]. A minimum spanning tree (MST) for $G$ is defined as a spanning tree with the minimum sum of edge weights. Suppose that we examine an arbitrary sample point, say $x_i$, and without loss of generality let $X = \{x_1, \ldots, x_k\}$ denote the set of $k$-nearest neighbors of
The MST-based outlier detection procedure proceeds as follows. First, we compute a weighted Minimum Spanning Tree (MST) using $X = \{x_1, ..., x_k\}$. We define an undirected, weighted, fully connected graph $G = (V, E, W)$, where $V = \{x_1, ..., x_k\}$, $E = \{e_12, e_13, ..., e_{(k-1)k}\}$ and $W = \{w(e_{12}), w(e_{13}), ..., w(e_{(k-1)k})\}$. An element in $E$, $e_{ab}$, stands for an edge associated with $x_a$ and $x_b$, and $w(e_{ab})$ is the Euclidean distance between the two vertices.

An MST algorithm such as Prim’s algorithm [4] provides a $k - 1$ edge list, $E' = \{..., e_{ia}', e_{ib}', ...\}$ where we assume $e_{ia}'$ and $e_{ib}'$ are the two edges associated with $x_i$, which spans all the points in $X$ with the minimum total edge weights\(^1\). After obtaining the MST, we compute the average weight ($mean_w$) of the MST. If the maximum edge weight associated with $x_i$ is much greater than the average edge weight, we divide the MST into two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. This is done by disconnecting one of the edges, $e_{ia}'$ or $e_{ib}'$, whose weight is greater than the other as shown in Fig. 2.2. In the next step, we examine whether or not the point is far from the other patch. Let us assume $G_1$ contains $x_i$. We compare the maximum edge weight in $G_1$ to the average edge weight ($mean_w$). If the edge weight is also much greater than the average weight, the point is determined as the outlier between the two patches.

If a point, $x_i$ is diagnosed as an outlier between two local patches, it is moved in the direction of one of the patches without projecting it onto its tangent space according to the following equation:

\(^1\)In this section, we only consider the outliers between two patches of a manifold (Fig. 2.1-(b)) because WPCA can handle the other type of outliers shown in Fig. 2.1-(a) quite well.
\begin{align*}
    x_i^* &= \alpha x_i + (1 - \alpha)x_t, \quad 0 < \alpha < 1 \\
    t &= \underset{\{W(e'_{ia}), W(e'_{ib})\}}{\text{argmax}} 
\end{align*}

Fig. 2.2 illustrates an example where a point $x_i$ (red circle) is an outlier located between the two local patches. The circles represent data points and the lines stand for edges computed using MST. The two blue circles are two points connected by the point, $x_i$. In the figure, the MST is divided into two subgraphs, $G_1$ and $G_2$ by disconnecting the edge, $e_{ia}$, because the edge weight is much greater than the average edge weight of the MST. The two dashed-ellipsoids show the two sub-graphs. The point, $x_i$, is determined as an outlier between two patches because the maximum weight, $W(e_{bc})$, in graph $G_1$ is also much greater than the average edge weight, $\text{mean}_w$. The outlier is moved to, $x_i^*$, in the direction of $x_a$ because $W(e_{ia})$ is less than the distance between $x_i$ and $x_c$. 
Fig. 2.1. WPCA results for various $\gamma$’s. Blue line denotes the first principal component direction of the original PCA, and the other color lines denote those of WPCA with different $\gamma$’s (cyan:$\gamma=0.1$, magenta:$\gamma=0.5$ and green:$\gamma=1$). Magenta line is overlapped with green line in (a), and cyan line is overlapped with blue line in (b). “X” mark denotes the weighted center after applying WPCA.

The circles represent data points and lines represent MST edges. The two blue circles, $x_a$ and $x_b$ are two neighbor points of $x_i$ composing of two MST edges associated with the point. The green square is the new position of $x_i$.

Fig. 2.2. An example of Minimum Spanning Tree (MST)
2.5 Bias Reduction

Local-smoothing based on linear local fitting suffers from the well-known *trim the peak* and *fill the valley* phenomenon for the regions of the manifold where the curvature is large [38]. When we increase the number of iterations in local linear smoothing, the net effect is that the set of projected sample points tend to shrink (Fig. 2.3, (b) and (c)), and eventually converge to a single point. One remedy is to use high-order polynomials instead of the linear one [38, 52], but this can be very expensive for high-dimensional data because we need to approximate quantities such as the Hessian matrix of \( f(\tau) \).

Because of this, we propose another solution to bias correction. Let \( x_i \) be an arbitrary sample point and denote \( x_i^* \) as the updated point of \( x_i \), \( i = 1, \ldots, N \) by applying the above smoothing algorithm (WPCA and MST). Clearly, if we know an approximation \( \delta_i \) for the difference \( f(\tau_i) - x_i^* \) of the updated point \( x_i^* \) of \( x_i \), \( x_i^* \) can be improved by adding \( \delta_i \) to \( x_i^* \) by reducing the bias in \( x_i^* \),

\[
   x_i^* \leftarrow x_i^* + \delta_i
\]

The effectiveness of a bias correction method depends on the bias estimation of \( f(\tau_i) - x_i^* \). We propose to estimate the bias as \( f(x_i) - x_i^* \approx x_i^* - \hat{x}_i^* \), where \( \{\hat{x}_i^*\} \) is the updated points of \( \{x_i^*\} \) obtained by the same smoothing procedure (WPCA and MST) (Similar ideas have been applied in nonlinear time series analysis [52]). This idea suggests the following bias estimation

\[
   \delta_i = (\bar{x}_i^w + U_i U_i^T (x_i - \bar{x}_i^w)) - (\bar{x}_i^w + U_i^* U_i^{*T} (x_i^* - \bar{x}_i^w)),
\]
where $\bar{x}_i^w$ and $U_i$ are the weighted mean and orthonormal basis matrix of the affine subspace constructed from WPCA using the $k$ nearest neighbors of $x_i$ from the data set $\{x_i\}$, and $\bar{x}_i^{*w}$ and $U_i^{*}$ are the weighted mean and orthonormal basis matrix of the affine subspace constructed from WPCA using the $k$ nearest neighbors of $x_i^{*}$ from the data set $\{x_i^{*}\}$. This gives the following updating formula:

$$x_i^{*} = 2(\bar{x}_i^w + U_i U_i^T (x_i - \bar{x}_i^w)) - (\bar{x}_i^{*w} + U_i^{*} U_i^{*T} (x_i^{*} - \bar{x}_i^{*w})).$$

(2.10)

The proposed smoothing algorithm along with bias correction is summarized as follows. Given a set of sample points $\{x_i, \ldots, x_N\}$:

1. **(Local Projection)** For each sample point, $x_i$
   - Compute the $k$-nearest neighbors of $x_i$, say $x_{i1}, \ldots, x_{ik}$
   - Compute MST using $x_{i1}, \ldots, x_{ik}$
   - Determine if $x_i$ is an outlier between two patches in the manifold using MST
   - If it is, move $x_i$ in the direction of one of the two patches.
   - Otherwise, project $x_i$ onto an affine subspace which is computed by WPCA to obtain $x_i^{*}$.

2. **(Bias Correction)** For each $x_i^{*}$
   - Correct bias using Eq. 2.10

3. **(Iteration)** Go to 1 until convergence
Fig. 2.3. Shrinking Example. Green dots, red dots and blue dots are the original input data points, the smoothing results at the iteration and the target manifold, respectively. Parameters: $k=30$, $\gamma=0.1$; (a) Original input image. (b)&(c) are the results without bias correction. (d)&(e) are results with bias correction.

In our experiments with noise-corrupted data, it was observed that the results converge to the original manifold very fast, but no formal proof of convergence has been worked out.

2.6 Experimental Results

In this section, we present experimental results for our local smoothing algorithms using a 2D spiral curve data set and three image data sets. We will use $k$ to represent the number of the nearest neighbors, and $d$ the dimension of affine subspaces used for local smoothing. We also use $\gamma$ as the parameter used in Eq. 2.9 to compute weights.

2.6.1 2D Spiral Curve Data

For the 2D data points of the spiral curve, a Gaussian random noise corrupted data set was generated with 500 points from the curve and is plotted in Fig. 2.3-(a). We also generated an outlier overlaid spiral curve data set with 500 points: 400 points
sampled from the spiral curve and 100 points randomly selected for outliers (Fig. 2.5-(a)). The outlier points were generated uniformly in the range of \( x \in [-15, 15] \) and \( y \in [-15, 10] \), which includes the spiral curve.

Fig. 2.4 and 2.5 depict the smoothing results of the spiral curve data with Gaussian noise and the outlier overlaid spiral curve data, respectively. In these experiments, different \( \gamma \) values are used by fixing \( k \) to be 30. The first row in Fig. 2.4 illustrates smoothing results without applying the MST outlier handling method, and the second row illustrates those after applying the MST outlier handling. The results in the figure show that the MST outlier handling method makes the smoothing method less sensitive to \( \gamma \). Fig. 2.4 and Fig. 2.5 illustrate that the relatively small \( \gamma \) produces better results for the Gaussian noise data while relatively large \( \gamma \) yields better results for the outlier-overlaid data. These can be explained as follows. In the outlier-overlaid case, most points are close to the curve, and a relatively small number of points are located far from the curve. When we compute a tangent space for an outlier point, most of the points in its \( k \)-nearest neighbor are close to the curve. In this situation, large \( \gamma \) yields a better tangent space than small \( \gamma \) as shown in Fig. ???. This is because the weight of outliers is close to zero and the weight of the other points on a manifold is non-zero for a large \( \gamma \). In the Gaussian noise case, all the points on the spiral curve are contaminated by zero mean random Gaussian noise. When we observe a local region of the curve, its tangent space tends to be close to that of a noise-free spiral curve at the region, and therefore low \( \gamma \) works better in this instance.

Fig. 2.6 presents smoothing results for different \( k \)'s by fixing \( \gamma \) to be 0.1 which seems to be the best \( \gamma \) based on our empirical experience. In general, the parameter \( k \)
should be chosen such that $k$ nearest neighbor points should represent the local linearity in the manifold dimension $d$. For this smooth manifold the results are relatively insensitive to the choice of the parameter $k$ as depicted in Fig. 2.6. We obtained similar results for the outlier overlaid case.
Fig. 2.4. Smoothing results for Gaussian noise data with different $\gamma$’s. Green dots are the input data, and red dots are smoothing results. The first row shows the results without MST outlier handling, and the second row shows those with MST outlier handling. The parameter $k$ is set to 30 and the number of iteration to 10.
Fig. 2.5. Smoothing results for outlier-overlaid data with different $\gamma$’s. The parameter $k$ is set to 30 and the number of iteration to 10.

Fig. 2.6. Smoothing results with different $k$s. $\gamma$ is set to 0.1 and the number of iteration to 10.
2.6.2 Experiments using 3D Swiss Roll Data

In this experiment, we focus on how noise affects the performance of manifold learning, and how the proposed local smoothing algorithm performs. We chose Isomap as a manifold learning algorithm and we generated three thousand 3D points from a swiss roll shape which have been used as popular data for manifold learning experiments. From this, we generated noisy test data by overlaying zero-mean Gaussian random noise onto the coordinates of the points. The two data sets, the noise-free swiss roll data set and the noisy swiss roll data set, which are shown in Fig. 2.7-(a) and Fig. 2.7-(b) respectively, are generated using matlab as follows:

```matlab
n = (3 * pi/2) * (0.5 + 2 * rand(1, N));
height = 10 * rand(1, N);
Swiss = [n.*cos(n); height; n.*sin(n)];
Swiss_Noise = Swiss + randn(3, N) * 0.8;
```

We obtained smoothing results of the noisy swiss roll data set by applying the proposed local smoothing method to the noisy swiss roll data. We set the parameters of the number of neighborhood points to 70 and $\gamma$ for weight computation to 0.01. Fig. 2.7 shows the plots of the three data sets: the original noise free swiss roll data, noise overlaid swiss roll data, and the smoothing results of the noisy swiss roll data respectively. In Fig. 2.7, the first column illustrates the distribution of the 3D points of each data set and
the second column illustrates their 2D plots after applying Isomap to the corresponding
3D points in the first column. The Isomap results were obtained by setting the number
of neighbors to 20, but the results were very similar when other numbers of neighbors
were used. As shown in Fig. 2.7-(d), the manifold learning algorithm is very sensitive
to noise. The proposed local smoothing algorithm, however, can handle the noise quite
well to produce a smoother swiss roll that the Isomap algorithm can then easily unfold.
Fig. 2.7. Plots of Swiss roll data
2.6.3 Smoothing 3D Points for Rendering

In the field of computer graphics, rendering using point primitives has received much attention because Modern digital scanner systems capture complex objects at the very high resolution and produce huge volumes of point samples. In this rendering method, surfaces are represented as a set of points without connectivity information. Surface Elements (Surfels [46]) are popularly used as rendering elements, which are point primitives without explicit connectivity. Surfels are comprised of attributes including depth, texture color, normal vector, and others. The rendering results of data sets are captured using PointShop3D [70] which is an interactive system for point-based surface editing.

In this experiments, we apply the proposed local smoothing algorithm to the noisy 3D points for point-based rendering. We performed experiments using two data sets. The first one, which is called Gnome, has 54,772 3D points captured from a statue of Gnome using a 3D scanner. Rendering results of the original noise-free Gnome is shown in Fig. 2.8. The other, which is called Santa, has 75,781 3D points. The rendering results of the original noise-free Santa is shown in Fig. 2.9.

Fig. 2.10 and Fig. 2.12 show two noise overlaid test data sets by adding Gaussian random noise of $N(0,2)$ and $N(0,4)$, respectively. The reconstruction results of Fig. 2.10 and Fig. 2.12 are shown in Fig. 2.11 and Fig. 2.13, respectively. We set $\gamma = 0.001$ and $k = 60$ for the Local Smoothing algorithm use for these experiments. Fig. 2.14 and Fig. 2.16 show two noise overlaid test data sets by adding Gaussian random noise of $N(0,3)$ and $N(0,5)$, respectively. The reconstruction results of Fig. 2.14 and Fig. 2.16
are shown in Fig. 2.15 and Fig. 2.17, respectively. We also set $\gamma = 0.001$ and $k = 60$ for the Local Smoothing algorithm in these experiments.

In both the Gnome and Santa experiment, we see that the Local Smoothing algorithm performs quite well. Although some detail is lost from the original, much of the noise has been smoothed out.
Fig. 2.8. Original Gnome data

Fig. 2.9. Original Santa data
Fig. 2.10. Noisy *Gnome* data with Gaussian random noise, $\mathcal{N}(0, 2)$

Fig. 2.11. Smoothing results of noisy *Gnome* data in Fig. 2.10
Fig. 2.12. Noisy Gnome data with Gaussian random Noise, $\mathcal{N}(0, 4)$

Fig. 2.13. Smoothing results of noisy Gnome data in Fig. 2.12
Fig. 2.14. Noisy Santa data with Gaussian random Noise, $\mathcal{N}(0, 3)$

Fig. 2.15. Smoothing results of noisy Santa data in Fig. 2.14
Fig. 2.16. Noisy *Santa* data with Gaussian random noise, $\mathcal{N}(0,5)$

Fig. 2.17. Smoothing results of noisy *Santa* data in Fig. 2.16
2.6.4 Face Images and Two Video Sequences (Human Walking and Ballet)

Experiments were also conducted using three image data sets: face images (total 698), a human walking video sequence (109 frames) and a ballet video sequence (166 frames). The face images lie essentially on a 3D manifold shown in [59, 58]. The three dimensions can be interpreted as follows: one dimension corresponds to the lighting variation and the other two dimensions correspond to horizontal and vertical pose variations, respectively. The gray-scale images are $64 \times 64$. The human walking video sequence and ballet video sequence are digitized in the sizes of $240 \times 352$ and $240 \times 320$ respectively. We generated images of the video sequences by using simple vision techniques. The cropped images of the walking clip are down-sized to $60 \times 40$, and the ballet images are down-sized to $50 \times 45$. They are also gray-scaled from 0 to 1.

We randomly selected images from each data set and overlaid a constant intensity noise patch at a random location. The intensity value of the patch is also selected randomly. The examples of the occlusion-overlaid images are shown in Fig. 2.19-(a) for the face images, Fig. 2.21-(a) for the human walking images and Fig. 2.23-(a) for the ballet images. We converted each image to a vector by concatenating each row: a 4096 dimensional vector for a face image, a 2400 dimensional vector for a human walking image and a 2250 dimensional vector for a ballet image.

Fig. 2.19 illustrates the first 42 occlusion-overlaid face images (in the size of $20 \times 20$ to 20% of the face image) and their smoothing results after applying the smoothing method, which are the results corresponding to Fig. 2.18-(c). As shown in Fig. 2.19-(b), most occlusion images are successfully projected onto the face manifold by removing
the occlusion parts after applying the proposed smoothing method to them. However, it is possible that some outliers are not projected to their original position in the face manifold. We will consider this problem in the next chapter.
Fig. 2.18. Residual variances for each dimension after applying Isomap to face images. (a) shows residual variances for original images without noise. $k_{\text{Isomap}}$ in (a) stands for the number of neighbors for Isomap. (b)-(e) show the residual variance for occlusion-overlaid cases (green line) and their smoothing results (blue line). $s \times s$, $r\%$ means a $s \times s$ size of noise patch is overlaid to $r\%$ of the data.

Fig. 2.19. (a) shows occlusion-overlaid face images in the size of $20 \times 20$ to $20\%$ of the data set. (b) shows the smoothing results of (a).
Fig. 2.20. Residual variances for each dimension after applying Isomap. (a) shows residual variances for original images without noise. (b)-(e) show the residual variances for occlusion-overlaid cases.

Fig. 2.21. (a) shows occlusion-overlaid human walking images in the size of 10 × 10 to 40% of the data set. (b) shows the smoothing results of (a).
Fig. 2.22. Residual variances for each dimension after applying Isomap. (a) show residual variances for original images without noise. (b)-(e) show the residual variance for noise contamination cases.

(a) Original images  (b) $10 \times 10$, 30%  (c) $10 \times 10$, 40%  (d) $15 \times 15$, 30%  (e) $15 \times 15$, 40%

Fig. 2.23. (a) shows occlusion-overlaid images in the size of $15 \times 15$ to 30% of the data set. (b) shows the smoothing results of (a).
Chapter 3

Image Denoising Using LARS Algorithm

In this chapter, we propose a parameter-free image denoising method adapted from the idea of identifying and updating only the noisy pixels. To achieve this goal, we start from the optimization problem of the $\ell_\infty$ norm object function with $\ell_1$ constraints in [63]. Actually, the $\ell_\infty$ norm object function with $\ell_1$ constraints is converted to Linear Programming (LP) in [63]. We use the $\ell_2$ instead of $\ell_\infty$ for the object function. Next, we show that the optimization problem can be converted into the ‘Least Absolute Shrinkage and Selection Operator’ (Lasso) [60] problem. Lasso was proposed by Tibshirani as a method for linear model estimation and has favorable properties of both subset selection and ridge regression [60]. Lasso minimizes the residual sum of the squares subject to the sum of the absolute value of the coefficients. The $\ell_1$ norm constraint of Lasso tends to shrink some coefficients and sets others to zero. Both the Lasso algorithm and the LP based image denoising approach share the basic idea that $\ell_1$ norm constraints produce sparse solutions [47]. Both of these methods need a parameter. The parameter controls the size of the model for the Lasso algorithm, while the parameter controls the number of pixels to be updated for the LP approach.

Recently, Efron et al. proposed a new model selection method for regression problems called Least Angle Regression (LARS) [19]. LARS was claimed to be a less greedy version of the traditional forward model selection method. A valuable property
of LARS is that it provides all possible LARS estimates for a given regression problem with efficient computation time. In other words, LARS provides the family of image denoising solutions starting with one updated pixel to determine all of the updated pixels in our application through iteration. A simple modification to the LARS algorithm implements Lasso (let us call it as LARS-Lasso). This LARS-Lasso algorithm provides all possible Lasso estimates. The image occlusion handling problem can be solved using either the LARS algorithm or LARS-Lasso algorithm. Actually, we do not need the $\ell_1$ constraint of the object function when we use LARS because the only reason we needed the constraint was to provide sparse solutions and LARS automatically provides all the possible sparse solutions we need. We also obtained the results that LARS and LARS-Lasso performed very similarly based on our experiments. Therefore, we will focus on image denoising using the LARS algorithm. To achieve a parameter-free system, we propose a method for determining the final optimal solution from the family of solutions that LARS provides, using the thresholding technique and statistical model selection criterion of Akaike Information Criterion (AIC).

Computer vision tasks of automatic learning such as face recognition and human gait recognition needs clean training data to construct robust learning systems. It is very difficult to manually verify all training images if the amount of training data is large.

Recently, Robust PCA (RPCA) [15] was proposed to compute a linear subspace using an occlusion overlaid image data set to deal with noise. Another proposed solution to this cleaning problem is to apply the proposed local smoothing method to the noisy data as per Chapter 2-Section 2.6.4. Also, using the local Weighted PCA (WPCA)
discussed in Chapter 2, we can then apply the LARS occlusion handling algorithm to the cleaning problem.

We propose an image decomposition method using modified Haar wavelet analysis to improve the computation time. This method divides an image into several independent subbands of the same size (smaller than the image) which preserve not only the frequency localization but also the time localization. With this time-frequency localization property, we can apply the LARS algorithm to each subband independently to reduce the computation time.

3.1 Problem Definition

We consider a sequence of images which forms a low dimensional manifold. A series of images usually lie on a low dimensional manifold compared to their dimension. Let us denote \( x, \hat{x}, \bar{x} \) and \( \tilde{x} \) as the occlusion-overlaid image, its original occlusion-free image, its estimated image by projecting it onto the local tangent in the direction of \( \alpha \) and its estimated image by projecting it orthogonally on the local tangent of manifold, respectively. Fig. 3.1 depicts the images \( (x, \hat{x}, \bar{x} \text{ and } \tilde{x}) \) in the two dimensional space. The vertical axis in Fig. 3.1 represents the noise-corrupted pixels of \( \hat{x} \) and the horizontal axis represents the other noise-free pixels of \( \hat{x} \). The curve indicates the manifold of the images, and the dashed line represents the local tangent of \( x \). As shown in the figure, \( \alpha \) is parallel to the vertical axis, which means it has non-zero values only for the noisy pixels and zero values for the other pixels. The ideal solution for denoising an noisy image is to compute \( \alpha \) correctly. Therefore, the widely used image denoising methods based on Principal Component Analysis (PCA), which project a occlusion overlaid image onto the
tangent space computed from PCA, are not plausible for handling occlusion because the tangent space is usually not orthogonal to $\alpha$.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{tangent_space.png}
\caption{This curve represents a manifold of images including, $\tilde{x}$. The vertical axis and the horizontal axis represent the occlusion pixels and the non-occlusion pixels of the image $x$ respectively.}
\end{figure}

Even though the claim raised above is reasonable intuitively, some people might argue that it would not work for very high dimensional data such as images. So, the following is an experimental example using images.

For this experiment, we select a gray image in the size of $64 \times 64$ shown in Fig. 3.2-(a) and name it boat. We generated 14 more images as training data for local tangent computation by increasing/decreasing all the intensity values in the boat image uniformly. The first row in Fig. 3.3 shows 7 images by decreasing the intensity value in multiples of 10, and the second row shows the other 7 images generated by increasing the intensity value by multiples of 10. The images lie on a one dimensional linear manifold even though the original dimension is 4096. We overlaid a constant noise patch whose intensity value is 255 in the size of $20 \times 20$ to the original boat image. The noise overlaid image, $x$ is
shown in Fig. 3.2-(b). A basis vector for the tangent space of the training images is computed using PCA. Two reconstruction images, \( \tilde{x} \) and \( \hat{x} \), are obtained by applying the algorithm which will be discussed in this chapter and by projecting \( x \) onto the tangent space, respectively. The two reconstructed images are shown in Fig 3.2-(c) and Fig 3.2-(d).

Fig. 3.2. Boat test image and its reconstruction images

Fig. 3.4 depicts the distribution of the images including the noisy test image and its projection images in 2D space. The vertical axis and horizontal axis represent the average intensity of the occlusion pixels and that of the non-occlusion pixels in the test image. Blue circles represent 14 training images and the black square indicates the original position of the occlusion image before the occlusion patch was overlaid. The pink star, the green star and the red star represent the occlusion image, and its two reconstruction images obtained using the algorithm we will propose and the orthogonal projection, respectively. As shown in Fig. 3.4, the orthogonal projection approach
Fig. 3.3. Fourteen Training boat images

projects the occlusion image onto the wrong position, while the method we are proposing reconstructs the occlusion image almost perfectly.

Based on this observation, we propose an image occlusion handling method using the LARS algorithm, which was originally proposed as a forward model selection method. This proposed method also exploits an image denosing method using $\ell_1$ constraints as proposed by Tsuda et. al. [63], which will be discussed in the following section.

We use the $\ell_2$ norm of the object function instead of the $\ell_\infty$ norm, which is used in [63], to compute the vector, $\alpha$, as shown in Fig. 3.1 and Fig. 3.4. Section 3.2.1 discusses about how the image denoising problem can be converted into a Lasso problem.
The blue circles represent the training images, the black square and pink dot represent the noise-free test image (Fig. 3.2-(a)) and its occlusion overlaid image (Fig. 3.2-(b)) respectively. The red dot and green dot indicate the reconstruction result of LARS (Fig. 3.2-(c)) and that of orthogonal projection (Fig. 3.2-(d)) respectively.

Fig. 3.4. Plot of the boat images into 2D space.
3.1.1 Image Denoising using $\ell_1$ constraints

Tsuda et al. proposed a method to compute the $\alpha$ by solving a Linear Programming (LP) equation with $\ell_1$ norm constraints [63]. Once $\alpha$ is computed, we can simply obtain the original occlusion-free image by $\hat{x} = \alpha + x$. This is, however, a chicken-and-egg problem because $\hat{x}$ is needed to compute $\alpha$ and $\alpha$ is needed to compute the noise-free image $\hat{x}$.

Tsuda et. al suggested a solution to this problem. First, he makes the following assumptions:

1. The denoised image is close to the manifold ($dist_1(\hat{x}, \bar{x}) \leq \epsilon_1$, for small positive value $\epsilon_1$).

2. The denoised image should be computed close to the noise-corrupted image. ($dist_2(x, \hat{x}) \leq \epsilon_2$, for small positive value $\epsilon_2$).

Before formulating the problem, let us denote $\mathcal{U}$ as the basis vectors (simply the eigen-vectors from PCA) that define the local tangent of the manifold. As shown in Fig. 3.1, the original noise-free image $\hat{x}$ and the estimated denosing image using the local tangent space $\bar{x}$ are represented as follows:

$$\hat{x} = x + \alpha \quad (3.1)$$

$$\bar{x} = \mathcal{U}t \quad (3.2)$$
where $t$ is a vector containing coefficients for linear combination of the basis vectors in $U$. Therefore, $\bar{x}$ locates on the subspace spanned by $U$.

The two assumptions discussed above can be re-formulated by utilizing $\ell_\infty$ norm for $\text{dist}_1$ and $\ell_1$ norm for $\text{dist}_2$ and using Eq. 3.1 and Eq. 3.2 as follows:

\[
\|\hat{x} - \bar{x}\|_\infty = \|x + \alpha - Ut\|_\infty \leq \epsilon_1 \tag{3.3}
\]
\[
\|x - \hat{x}\|_1 = \|\alpha\|_1 \leq \epsilon_2 \tag{3.4}
\]

Based on Eq. 3.3, the optimization problem is designed as shown in Eq. 3.5. It needs to compute not only $\alpha$ but also the coefficients $t$ for $\bar{x}$ simultaneously.

\[
\min_{\alpha, t} \|x + \alpha - Ut\|_\infty \tag{3.5}
\]
\[
\text{s.t.}
\]
\[
\|\alpha\|_1 \leq C,
\]

where $\| \cdot \|_\infty$, and $\| \cdot \|_1$ are $\ell_\infty$ norm and $\ell_1$ norm respectively, and $C$ is a constant to determine the sparseness of $\alpha$. Actually, $C$ is the parameter which controls the number of pixels to be updated. In the object function, the $\ell_\infty$ norm object function and the $\ell_1$ norm constraints were used because they can be converted into Linear Programming. The objective function in Eq. 3.5 is re-formulated as Linear Programming
using a regularization parameter, $\nu$, and $\alpha$ restated as $\alpha = \alpha^+ - \alpha^-$, $\alpha^+, \alpha^- \geq 0$, as shown in Eq. 3.6.

\[
\min_{\alpha^+, \alpha^-} \frac{1}{N} \sum_{n=1}^{N} (\alpha_n^+ + \alpha_n^-) + \nu \epsilon
\]  
(3.6)

s.t.

$$\alpha^+, \alpha^- \geq 0, \quad |x_n + \alpha_n^+ - \alpha_n^- - Ut| \leq \epsilon, \quad n \in [1,N]$$

The regularization constant, $\nu$, should be determined to control the fraction of the nonzero of $\alpha$ (pixels to be updated). Refer to [63] for the upper and lower bounds of the pixels to be updated. Eq. 3.6 also can be expressed in standard LP form with $\ell_1$ constraints. Let us also restate $t$ as $t = t^+ - t^-$, $t^+, t^- \geq 0$ in the same manner that we restated $\alpha$ to $\alpha^+$ and $\alpha^-$. $\theta, f, b$ and $A$ are defined as below.

$$\theta^T = [\alpha^+ T, \alpha^- T, t^+ T, t^- T, \epsilon]$$  
(3.7)

$$f^T = [e_d^T, e_d^T, 0_d^T, 0_d^T, \nu]$$  
(3.8)

$$b = \begin{bmatrix} x \\ -x \end{bmatrix}, \quad A = \begin{bmatrix} -I_d & I_d & U & U & -\epsilon \\ I_d & -I_d & -U & -U & -\epsilon \end{bmatrix}$$  
(3.9)
where $e_d$ is a vector with $d$ (the dimension of the images) elements of 1’s and $0_d$ is a zero vector with $d$ elements. In the same manner, $I_d$ is an identity matrix in the size of $d \times d$.

Now, we can obtain the simple standard LP formula shown in Eq. 3.10 from Eq. 3.6.

$$\min_{\theta} f^T \theta \quad (3.10)$$

s.t.

$$\theta \leq 0, \quad A \cdot \theta \leq b$$

### 3.2 Image Denoising using LARS

As previously mentioned, Least Angle Regression (LARS) was proposed by Efron et. al. in [19] as a new model selection method for regression problems. It is claimed to be less greedy than traditional forward selection algorithms. The LARS algorithm provides all possible LARS estimates for a given problem with an efficient computation time.

Section 3.2.1 discusses the LARS objective function and its relationship with the image denoising objective function discussed in Section 3.1.1. When we use the $\ell_2$ norm of the object function instead of the $\ell_\infty$ norm as shown in Eq. 3.11, we can solve the image denoising problem using LARS algorithm.

The LARS algorithm is summarized in Section 3.2.2.

$$\min_{\alpha} \|x + \alpha - U\ell\|_2 \quad (3.11)$$
s.t. \[ \|\alpha\|_1 \leq C \]

### 3.2.1 Relationship between Image Denoising Objective Function and Lasso

In regards to the LARS algorithm, let us define regression data \( \{(X_i, y_i)\}_{i=1}^n \), where \( X_i = [x_{i1}, x_{i2}, \cdots, x_{im}] \) are the predictor variables and \( y_i \in \mathcal{R} \) are the responses.

Let us also define a matrix \( X \) and a vector \( y \) as:

\[
X_{n \times m} = \begin{bmatrix}
  x_{11} & \cdots & x_{1m} \\
  \vdots & \ddots & \vdots \\
  x_{n1} & \cdots & x_{nm}
\end{bmatrix}, \quad y_{n \times 1} = \begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix}
\]  \hspace{1cm} (3.12)

Each row of \( X \) corresponds to the predictors of a datum. Let us suppose that the predictor variables have been standardized to have zero mean and unit length, and the responses also have zero mean. Then, LARS estimate, \( \hat{\alpha} \) is defined as:

\[
\hat{\alpha} = \arg \min_{\alpha} \|y - X\alpha\|_2
\]  \hspace{1cm} (3.13)

Now, we show how the object function in Eq. 3.13 can be converted into a regression problem for LARS. Suppose that the solution of the objective function in Eq. 3.11 is close to zero. Then, we can approximate \( t \) as in Fig 3.14.

\[
x + \alpha - Ut \simeq 0 \iff t \simeq U^\dagger(x + \alpha), \]  \hspace{1cm} (3.14)

where \( U^\dagger \) represents \( (U^TU)^{-1}U^T \).
Obviously, $\mathbf{U}^T\mathbf{U}$ is non-singular in the most cases for computer vision applications because the number of rows of $\mathbf{U}$ (the number of pixels of an image) is usually much larger than the number of columns of $\mathbf{U}$ (the number of basis vectors). We can then reformulate the objective function in Eq. 3.11 by substituting $\mathbf{U}^\dagger(x + \alpha)$ for $t$ as follows:

$$\min_{\alpha} \|x + \alpha - \mathbf{U}t\|_2 
\simeq \min_{\alpha} \|x + \alpha - \mathbf{U}(\mathbf{U}^\dagger(x + \alpha))\|_2 
\iff \min_{\alpha} \|(I - \mathbf{U}\mathbf{U}^\dagger)(x + \alpha)\|_2 
\iff \min_{\alpha} \|(I - \mathbf{U}\mathbf{U}^\dagger)x - (\mathbf{U}\mathbf{U}^\dagger - \mathbf{I})\alpha\|_2, \quad (3.15)$$

where $I$ stands for the identity matrix.

The new objective function in Eq. 3.15 is now in the equivalent format of Eq. 3.13 for LARS, in which $y = (I - \mathbf{U}\mathbf{U}^\dagger)x$ and $X = \mathbf{U}\mathbf{U}^\dagger - \mathbf{I}$. If we use orthonormal vectors for $\mathbf{U}$ such as the basis vectors from PCA, $\mathbf{U}^\dagger$ can be simplified as $\mathbf{U}^T$ because $\mathbf{U}^T\mathbf{U}$ becomes the identity matrix. Therefore, LARS can be used for solving the image denoising problem.

### 3.2.2 Overview of LARS algorithm

In this section, we summarize the LARS algorithm proposed in [19]. We will use their notation. Let $x_1, \ldots, x_m$ represent the covariates (the columns of the matrix $X$ in
Eq. 3.12). It is assumed that the covariates have been standardized to have zero mean and unit length.

\[ \sum_{i=1}^{n} y_i = 0, \quad \sum_{i=1}^{n} x_{ij} = 0, \quad \sum_{i=1}^{n} x_{ij}^2 = 1, \quad j = 1, 2, \ldots, m \]  \hspace{1cm} (3.16)

A candidate vector of regression coefficients \( \hat{\alpha} = [\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_m] \) provides a prediction vector \( \hat{\mu} \). The regression problem is to compute the vector \( \alpha \) by minimizing the total squared error of the objective function shown below:

\[ \min_{\alpha} \sum_{j=1}^{n} (y_i - x_i \alpha)^2 = \min_{\alpha} \| y - X \alpha \|_2 \]  \hspace{1cm} (3.17)

The LARS algorithm is an improved version of the classic forward model selection approach for regression problems. It builds up estimates, \( \hat{\mu} = X \hat{\alpha} \), in successive steps where each iteration adds one covariate to the model. LARS claims to be a stylized version of the Forward Stagewise Linear Regression (stagewise) \([19]\). LARS exploits correlations shown in 3.18 as a metric for feature selection.

\[ \hat{c} = c(\hat{\mu}) = X^T(y - \hat{\mu}) \]  \hspace{1cm} (3.18)

Let us define \( \mathcal{A} \) as the active set of indices corresponding to the covariates selected by the current iteration. Let us also define \( \mathcal{A}_+ \) as the set of indices at the next iteration of \( \mathcal{A} \). \( X_\mathcal{A} \) is a matrix containing the covariates of \( \mathcal{A} \). \( \mu_\mathcal{A} \) denotes as the equiangular vector computed using the vectors in the active set. \( \hat{\mu}_\mathcal{A} \) represents the estimation of \( y \) using \( X_\mathcal{A} \). It is initialized as a zero vector in the beginning of the algorithm.
The LARS algorithm can be divided into three steps for each iteration. The first step is to select one covariate from the inactive set (denoted as $A^c$), which has the highest absolute correlation with residual (the maximum of $X_{A^c}^T(y - \hat{\mu}_{A^c})$) and add it into the active set. Actually, this absolute correlation is the same as that of the active set, which will be discussed further. In the second step, the equiangular vector $\mu_A$ is computed using the vectors in the active set. In the last step, a new estimation of $y$, $\hat{\mu}_{A^+}$ is computed using $\hat{\mu}_A$ and the equiangular vector, $\mu_A$. The regression solution, $\hat{\alpha}$ is computed using the predictor vector $\hat{\mu}_{A^+}$.

Now, let us discuss why the correlation is used as a metric by LARS. The LARS algorithm finds one covariate and adds it into the selected feature set in each iteration by considering the objection function in Eq. 3.17. Let us divide the matrix $X$ into two parts: $X = [X_A X_{A^c}]$, in which, $X_A$ contains the selected covariates at the current time, and $X_{A^c}$ contains the other covariates from which the LARS algorithm searches for the next candidate covariate. Then, $X\alpha$ in Eq. 3.17 can be expressed as:

$$X\alpha = [X_A X_{A^c}] \begin{bmatrix} \alpha_A \\ \alpha_{A^c} \end{bmatrix}$$

$$= X_A \alpha_A + X_{A^c} \alpha_{A^c} \quad (3.19)$$

By applying the result in Eq. 3.19 to Eq. 3.17, we obtain a new object function:
\[
\min_{\alpha} \| y - X\alpha \|_2 \\
= \min_{\alpha} \| y - X_A\alpha_A - X_C\alpha_C \|_2 
\] (3.20)

In Eq. 3.20, \(X_A\alpha_A\) is the prediction vector \(\hat{\mu}_A\) computed using \(x \in X_A\). Let the current residual \(r = y - X_A\alpha_A\). Then, the object function for the next feature selection is as shown in Eq. 3.21. For simplicity, \(X_c^A\) and \(\alpha_c^A\) are denoted as \(X\) and \(\alpha\) in Eq. 3.21.

\[
\arg \min_{x_i \in X} (r - x_i\alpha)^2 \\
= \arg \min_{x_i \in X} (r - x_i\alpha)^T (r - x_i\alpha) \\
= \arg \min_{x_i \in X} (r^T r - 2\alpha^T x_i^T r + \alpha^T x_i^T x_i\alpha) \\
= \arg \min_{x_i \in X} (r^T r - \alpha^T (2x_i^T r - \alpha)), 
\] (3.21)

where \(x_i^T x_i = 1\) (refer Eq. 3.16).

We can then reformulate the above object function to Eq. 3.22 because \(r^T r\) and \(\alpha\) can be treated as constants.

\[
\arg \max_{x_i \in X} x_i^T r = \arg \max_{x_i \in X} x_i^T (y - \hat{\mu}_A) 
\] (3.22)

Therefore, the next covariate is selected which has the maximum correlation with the residual which is the reason that the LARS algorithm uses the correlation metric.
Now, let us briefly discuss the procedure for the LARS algorithm. It can be summarized as follows using the same notation used in [19].

**Step 0.** Initialize \( \hat{\alpha} \) and \( \hat{\mu} \) as zero vectors, and \( \mathcal{A} \) as an empty set.

**Step 1.** Select one covariate from the inactive set and add it to active set

- Compute the current correlations \( \hat{c} \), the maximum correlation \( \hat{C} \), and the sign(+/−) of \( \hat{c} \) (s). Select the covariate from the inactive set whose correlation is \( \hat{C} \) and insert it into the active set \( \mathcal{A} \)

\[
\hat{c} = X^T(y - \hat{u}_\mathcal{A}) \\
\hat{C} = \max_j \{\|\hat{c}_j\|\} \\
\mathcal{A} = \{j : \|\hat{c}_j\| = \hat{C}\} \\
s_j = \text{sign}\{\hat{c}_j\}, \ j \in \mathcal{A},
\]

where \( s_j \in s, \hat{c}_j \in \hat{c} \) and \( \hat{u}_\mathcal{A} \) is the current LARS estimate of \( y \).

- Construct \( X_\mathcal{A} \) by adding a covariate, which is the matrix containing the selected covariates at the current time.

\[
X_\mathcal{A} = (\cdots s_j x_j \cdots)_{j \in \mathcal{A}}
\]

where \( x_j = [x_{1j}, \cdots, x_{nj}]^T \).
Step 2. Compute $u_A$ (the equiangular unit vector) and $A_A$ as follows

$$A_A = (1^T_A G_A^{-1} 1_A)^{1/2}$$

$$u_A = X_A w, \quad w = A_A G_A^{-1} 1_A,$$

where $G_A = X_A^T X_A$, and $1_A$ is a vector of 1’s of length equaling the number of indices in the set $A$.

Step 3. Compute $\hat{\mu}_{A_+}$ and $\hat{\alpha}^+$ for the next iteration.

$$\hat{\gamma} = \min_{j \in A^c} \{ \hat{C} - \hat{\epsilon}_j, \hat{C} + \hat{\epsilon}_j \} \quad \frac{\hat{C} - \hat{\epsilon}_j}{A_A - a_j}, \frac{\hat{C} + \hat{\epsilon}_j}{A_A + a_j}$$

where $A^c$ is complement of $A$ and $\min^+$ indicates that the minimum is taken over only positive components within each choice of $j$ [19].

$$\hat{\mu}_{A_+} \leftarrow \hat{\mu}_A + \hat{\gamma} u_A$$

$$\hat{\alpha}_j^+ \leftarrow \hat{\alpha}_j + \hat{\gamma} s_j w_A, \quad j \in A,$$

where $\hat{\alpha}$ and $\hat{\alpha}^+$ are the solutions of Eq. 3.13 at the current iteration and the next iteration respectively.
We can also implement the Lasso (LARS-Lasso) algorithm by slightly modifying this step as follows:

\[ \tilde{\gamma} = \min_{\gamma_j > 0} \{ \gamma_j \}, \quad \gamma_j = -\hat{\alpha}_j s_j w_A, \quad j \in A \]

\[ \tilde{j} = \arg \min_{j, \gamma_j > 0} \{ \gamma_j \}, \]

- If \( \tilde{\gamma} < \hat{\gamma} \), update \( u_A \) using \( \tilde{\gamma} \), delete the index \( \tilde{j} \) as follows and go to Step 3.

\[ \hat{\mu}_A^{-} \leftarrow \hat{\mu}_A^{-} + \tilde{\gamma} u_A \]

\[ \hat{\alpha}_j^+ \leftarrow \hat{\alpha}_j + \tilde{\gamma} s_j w_A, \quad j \in A \]

\[ A_{+} \leftarrow A - \{ \tilde{j} \} \]

- Otherwise, update \( u_A \) using \( \hat{\gamma} \) as follows and go to Step 1.

\[ \hat{\mu}_A^{-} \leftarrow \hat{\mu}_A^{-} + \hat{\gamma} u_A \]

\[ \hat{\alpha}_j^+ \leftarrow \hat{\alpha}_j + \hat{\gamma} s_j w_A, \quad j \in A, \]

where \( \hat{\alpha} \) and \( \hat{\alpha}^+ \) are the solution of Eq. 3.13 at the current iteration and the next iteration respectively.

For explanation of the algorithm in detail, refer to the original LARS paper [19].
3.3 The Optimal Solution Computation Using Thresholding and AIC

In this section, we discuss how to estimate the final optimal solution of $\hat{\alpha}$ among the family of solutions that the LARS algorithm provides, using a thresholding technique and a statistical model selection criterion. Efron et. al. also proposed the model selection method by deriving the $C_p$ estimate. However, a lot of questions were raised regarding whether the $C_p$ estimate is the correct stopping rule for LARS. It is obvious that the $C_p$ estimate cannot work well for all the LARS problems. In our case, we deal with visual data, so we can exploit useful information coming from the visual cue.

3.3.1 Statistical Model Selection Criteria

There are several statistical model selection criteria which have been established by statisticians such as Mallow’s $C_p$ [39] [40], Akaike Information Criterion (AIC) [1] [2] and its corrected version [57] called $AIC_c$, Baysian Information Criterion (BIC) [3], Minimum Description Length (MDL) [48] and so on.

The $C_p$ criterion proposed by Mallows [39] is based on the mean squared prediction error. AIC uses a log-likelihood loss function and is based on Kullback-Leibler (K-L) distance [35]. A predictive expectation version of the log-likelihood is used to estimate the relative expected K-L distance between the approximating model and the true distance between the two models [13]. Many authors, however, have shown that the small-sample properties of AIC lead to over-fitting. So $AIC_c$ is proposed to correct the small-sample over-fitting tendencies of AIC by estimating expected K-L distance directly rather than estimating an approximation of it.
BIC, which is similar to AIC, is based on the Bayesian approach for model selection. The model with the minimum value of BIC is chosen as the model with the largest posterior probability (the probability of the data given a model). This criterion penalizes for over-fitting (complex model) more than AIC.

The Minimum Description Length (MDL) is statistically identical to the BIC approach, but is motivated from an optimal coding viewpoint in information theory.

For more information about model selection in detail, refer to [13], [42] or [24].

3.3.2 Selecting the Optimal Solution Using Thresholding and AIC

As discussed in Section 3.2.2, the LARS algorithm provides a value for $\alpha$, which is initially a zero vector, at each iteration. Fig. 3.5-(a) depicts the images of the absolute value of $\alpha$ computed at the iteration of 25, 50, 100, 300 and 500, and (b) shows their corresponding binary images after applying a thresholding algorithm, in which white and black represent a nonzero and zero value of $\alpha$, respectively. Let us denote $|\alpha|$ as the absolute value of $\alpha$. As illustrated in the images, the $|\alpha|$ images at the iteration of 100, 300 and 500 are very similar to each other, whereas their thresholding images are very different. The magnitude of non-zero values in $\alpha$ is not very significant after all the values of the occlusion are converted to a non-zero by the LARS algorithm.

Based on this observation, we propose a method for selecting one final solution $\alpha$ among the family of $\alpha$’s that the LARS algorithm provides. We do this by combining a thresholding technique and with the AIC model selection method. At each iteration of the LARS algorithm, the number of foreground pixels of an $\alpha$ image is computed by applying a thresholding algorithm to that $|\alpha|$ image. The algorithm that we utilize
is the Otsu algorithm [44]. Fig. 3.5-(c) depicts the binary images after applying the thresholding algorithm, and Fig. 3.6-(a) shows the number of foreground pixels (vertical axis) along the iteration (horizontal axis). As shown in the graph, the number of foreground pixels increases until all the pixels corresponding to occlusion in $\alpha$ are converted non-zero, and it becomes stable after that.

Fig. 3.6-(b) also illustrates the reconstruction error along the iteration. As shown in the figure, the error forms “L” shape. It decreases significantly until the entire occlusion region is covered by the LARS algorithm. It reaches the minimum at a certain point, and then increases a little bit after the minimum.

Now, we discuss how the final $\alpha$ is selected using the thresholding results utilizing the AIC model selection method. Let us denote $\theta$ as the parameter set to define a model, and $\hat{\theta}$ as the maximum likelihood estimate of $\theta$. Then, AIC is defined as follow:

$$AIC = -2 \log \text{lik} + 2K,$$
$$\log \text{lik} = \sum_{j=1}^{N} \log P(y_j | \hat{\theta}),$$  \hspace{1cm} (3.23)

where $K$ is the number of parameters, $P(y_j | \hat{\theta})$ is a conditional probability (probability of $y_j$ given the parameter $\hat{\theta}$) and $\log \text{lik}$ represents the maximized log-likelihood.

Let us define $AIC_i$ and $\alpha_i$ as the AIC and $\alpha$ at iteration $i$. Then we choose the solution for $\hat{\alpha}$ which gives the smallest AIC over the set of $\alpha_i$’s as shown in Eq. 3.24.
\[ \hat{i} = \arg\min_i \{AIC_i\} \]

\[ \hat{\alpha} = \alpha_i \]

We compute \( \loglik_i \), which is the maximized log-likelihood at the iteration \( i \), as follows:

\[
\loglik_i = \sum_{j=1}^{N} \log P(y_j | \hat{\theta}_i) = \frac{N \times |FG_i|}{\sum_{k=1}^{\#\text{iter}} |FG_k|},
\]

where \( |FG|_i \) indicates the sum of the absolute values of each foreground pixel in \( \alpha_i \), and \( \#\text{iter} \) is the number of the iteration.
Fig. 3.5. (a) $|\alpha|$ images computed by applying the LARS algorithm. (b) Binary images of $|\alpha|$’s. The white pixels represent a non-zero value. (c) Binary images after applying the thresholding algorithm to (a). (d) Reconstruction images by adding $\alpha$ to the occlusion image.

Fig. 3.6. (a) The blue line and red dashed line represent the number of foreground pixels and the sum of the absolute values of the foreground pixels, respectively, after applying the thresholding algorithm. (b) $\ell_2$ norm reconstruction error.
3.4 Improvement of the Computation Time

The most time consuming process in the LARS algorithm is the computation of $G_A^{-1}$, inverse of $G_A$. As discussed in Section 3.2.2, one covariate is involved for the LARS algorithm for each iteration. This means that $G_A^{-1}$ of the next iteration, which is denoted as $G_A^{-1+}$, can be efficiently computed using $G_A^{-1}$. This issue was also briefly mentioned in [19]. In this section, we discuss the efficient computation of $G_A^{-1+}$ in detail based on the Sherman-Morrison-Woodbury formula [22].

Suppose that a covariate vector $x$ is added to $X_A$ for the next iteration. Then, $X_{A+}$ is defined as Eq. 3.24. The $x$ in Eq. 3.24 is the multiplication of the vector of the covariate by its corresponding sign (refer to the LARS algorithm in Sec 3.2.1).

\[
X_{A+} = [X_A \ x] \tag{3.24}
\]

$G_{A+}$ is defined as:

\[
X_{A+} = [X_A \ x] \\
G_{A+} = [X_A \ x]^T [X_A \ x] = \begin{bmatrix} X_A^T X_A & X_A^T x \\ x^T X_A & x^T x \end{bmatrix} \tag{3.25}
\]

Let us denote $X_A^T x$ and $x^T x$ as $b$ and $c$ respectively. Then,
\[ G_{A+} = \begin{bmatrix} G_A & b \\ b^T & c \end{bmatrix} \]

(3.26)

Let us factorize \( G_A \) and \( G_{A+} \) into a lower triangle and an upper triangle matrix using the Cholesky factorization algorithm:

\[
G_A = R^T R \\
G_{A+} = \begin{bmatrix} R^T & 0 \\ s^T & d^T \end{bmatrix} \begin{bmatrix} R & s \\ 0 & d \end{bmatrix},
\]

(3.27)

where \( s = R^{-T} b \) and \( d^T d = c = x^T x \).

Then, the \( G_{A+}^{-1} \) is represented as:

\[
G_{A+}^{-1} = \begin{bmatrix} R & s \\ 0 & d \end{bmatrix}^{-1} \begin{bmatrix} R^T & 0 \\ s^T & d^T \end{bmatrix}^{-1}
\]

\[
= \begin{bmatrix} R^{-1} & -R^{-1} s d^{-1} \\ 0 & d^{-1} \end{bmatrix} \begin{bmatrix} R^{-T} & 0 \\ -d^{-T} s^T R^{-T} & d^{-T} \end{bmatrix}
\]

\[
= \begin{bmatrix} R^{-1} R^{-T} + R^{-1} s (d^T d)^{-1} s^T R^{-T} & R^{-1} s (d^T d)^{-1} \\ -R^{-1} s (d^T d)^{-1} s^T R^{-T} & (d^T d)^{-1} \end{bmatrix}
\]

\[
R^{-1} s = R^{-1} (R^{-T} b) = G_{A}(t)^{-1} b
\]

(3.28)
Therefore, $G_{A+}^{-1}$ can be efficiently computed using $G_{A}^{-1}$:

$$\begin{align*}
G_{A+}^{-1} &= \begin{bmatrix}
G_{A}^{-1} + G_{A}^{-1}b(d^T d)^{-1}b^T G_{A}^{-T} & -G_{A}^{-1}b(d^T d)^{-1} \\
-(d^T d)^{-1}b^T G_{A}^{-T} & (d^T d)^{-1}
\end{bmatrix} \\
&= \begin{bmatrix}
G_{A}^{-1} + HeH^T & -He \\
-eH^T & e
\end{bmatrix}, \\
\end{align*}$$

(3.29)

where $H = G_{A}^{-1}b$, $e = (d^T d)^{-1} = (x^T x)^{-1}$.

As mentioned in Section 3.2.2, when we use the LARS algorithm, one covariate needs to be added, but when we use the LARS-Lasso algorithm, one covariate is either added or deleted at each iteration.

Next, let us discuss the computation of $G_{A+}^{-1}$ when a covariate is deleted from $X_A$ for the LARS-Lasso algorithm. Suppose that a covariate $x$ is deleted from $X_A$, and $E$ is the permutation matrix to switch $x$ with the last column of $X_A$. It is thus expressed as:

$$X_AE = [X_{A+} \ x]$$

We can compute $G_{A+}^{-1}$ using Eq. 3.30 as follows.

$$\begin{align*}
([X_{A+} \ x]^T[X_{A+} \ x])^{-1} &= ((X_A E)^T(X_A E))^{-1} \\
\Leftrightarrow ([X_{A+} \ x]^T[X_{A+} \ x])^{-1} &= E^T(X_A^T X_A)^{-1}E
\end{align*}$$

(3.30)

Using Eq. 3.25 through Eq. 3.29, we can derive Eq. 3.31 from Eq. 3.30.
\[
\begin{bmatrix}
G_{A+}^{-1} + H_1 eH_1^T & -H_1 e \\
-eH_1^T & e
\end{bmatrix} =
\begin{bmatrix}
Q & u \\
u^T & v
\end{bmatrix}
\] (3.31)

The right-hand side of Eq. 3.31 can be computed by exchanging the row and the column corresponding to the \(x\) with the last row and the column in \(G_A\). Finally, we compute \(G_{A+}^{-1}\) as shown in Eq. 3.32 using the relationship of \(Q = G_{A+}^{-1} + H_1 eH_1^T\), \(u = -H_1 e\) and \(v = e = (x^T x)^{-1}\).

\[
G_{A+}^{-1} = Q - H eH^T = Q - u(x^T x)u^T
\] (3.32)

### 3.5 Sub-Block Computation Using Wavelets decomposition

In this section, we further discuss how to reduce the needed computation time. The computation time needs to be reduced because a lot of matrix and vector multiplication and addition is involved in the LARS algorithm (see Sec. 3.2.2 and Sec. 3.4). Therefore we will decrease the time complexity by reducing the dimension of the input data using the Haar wavelet decomposition method.

#### 3.5.1 Basic concept of Haar Wavelet Analysis

First, we divide an image into several smaller independent sub-blocks. To achieve this goal, we exploit Haar wavelet analysis which decomposes a signal into independent subbands. Let us explain the Haar wavelet analysis briefly. We follow the notation used in [10].
Two functions, the scaling function $\phi$ and the wavelet function $\psi$, play a primary role in wavelet analysis. Eq. 3.33 shows the definition of the two functions for a one-dimensional (1D) case.

\[
\phi(x) = \begin{cases} 
1 & \text{if } 0 \leq x < 1 \\
0 & \text{otherwise}
\end{cases}
\]

\[
\psi(x) = \phi(2x) - \phi(2x - 1) \tag{3.33}
\]

Let us define $V_j$ and $W_j$ as the space of the function in Eq. 3.34 and in Eq. 3.35, respectively.

\[
\sum_{k \in \mathbb{Z}} a_k \phi(2^j x - k), \; a_k \in \mathbb{R} \tag{3.34}
\]

\[
\sum_{k \in \mathbb{Z}} a_k \psi(2^j x - k), \; a_k \in \mathbb{R}, \tag{3.35}
\]

where $a_k$ is non-zero only for a finite number. Then, we obtain that $W_{j-1}$ is the orthogonal complement of $V_{j-1}$ in $V_j$ [10] (i.e. $V_{j+1} = V_j \cup W_j$ and $V_j \cap W_j = \emptyset$). Let us denote it as $V_{j+1} = V_j \oplus W_j$. Recursively, we obtain the following:

\[
V_j = W_{j-1} \oplus W_{j-2} \oplus \cdots \oplus W_0 \oplus V_0 \tag{3.36}
\]
Given a function, \( f_j \in V_j \), in Eq. 3.37, we can decompose the function using Haar wavelet analysis as shown in Eq. 3.38 (refer to Theorem 4.12 in [10]).

\[
f_j(x) = \sum_{k \in \mathbb{Z}} a^j_k \phi(2^j x - k) \tag{3.37}
\]

\[
f_j = w_{j-1} + f_{j-1} \tag{3.38}
\]

\[
= w_{j-1} + w_{j-2} + \cdots + w_0 + f_0
\]

\[
= \sum_{k=-\infty}^{j-1} w_{j-k}
\]

where, \( f_j \in V_j \), \( f_{j-1} \in V_{j-1} \), and \( w_{j-1} \in W_{j-1} \).

We can simply extend the 1D Haar wavelet analysis to 2D Haar wavelet analysis as shown in Eq. 3.39. Refer to Appendix B for more detail regarding the derivation of Eq. 3.39.

\[
f_j(x, y) = w^x_{j-1} w^y_{j-1} + w^x_{j-1} f^y_{j-1} + f^x_{j-1} w^y_{j-1} + f_{j-1}(x, y), \tag{3.39}
\]

where the superposition characters \( x \) and \( y \) represent the direction of the 1D Haar wavelet analysis. As shown in Eq. 3.39, 2D Haar wavelet analysis decomposes a signal into 4 subbands for each level while 1D Haar wavelet decomposes it into 2 subbands as shown in Eq. 3.38. The four subbands are usually labeled as “HH”, “HL”, “LH” and “LL” [54], because \( w_{j-1} \) and \( f_{j-1} \) can be considered as a high-pass filter (differential
Fig. 3.7. Haar wavelet decomposition of an image.

operator) and a low-pass filter (average operator), respectively (refer to Eq. 3.33 and Eq. B.2). Fig. 3.7 depicts the results of the original 2D Haar wavelet decomposition of 1st and 2nd level.

3.5.2 Decomposition of Images using Modified Haar Wavelet Analysis

Our goal is, however, to divide an image into several independent blocks of equal size (or at least similar size) to decrease the computation time. The original Haar decomposition produces different sized blocks when the level is deeper than 1 as shown in Fig. 3.7-(b). So, we slightly modify the original Haar wavelet decomposition to satisfy our goal. In each level of decomposition, we apply the Haar wavelet transform to each subband instead just one. Fig. 3.8 illustrates this new decomposition in the second level Haar wavelet analysis. We can obtain $2^{2^j}$ subbands (or blocks) of the same size for $j^{th}$ level 2D Haar wavelet decomposition. Obviously, we can reconstruct the original image
Fig. 3.8. Modified Haar wavelet decomposition

completely by applying the inverse Haar wavelet transforms recursively. The subbands at each level of the decomposition are also independent to each other. This is shown using Eq. 3.36.

In our application, we apply the proposed modified Haar wavelet decomposition to the input images to divide the images into independent subbands whose size is smaller than that of the original image. After that, the proposed LARS algorithm is applied to each subband independently. Finally, the inverse Haar wavelet transform is applied to the resultant subbands of the LARS algorithm. The resulting computation time is only a fraction of the time required when you apply the LARS algorithm to the original image directly. However, it might sacrifice the space localization according to the depth of decomposition level.
3.6 Experiments

In this section, we present experimental results of the proposed parameter-free LARS occlusion handling method in three aspects. First, we show how well the proposed method of the optimal solution estimation performs. Second, we discuss how the level of sub-block computation affects performance. Finally, we apply the proposed LARS image occlusion handling method to the noisy data cleaning problem, where the data contains occlusion overlaid images. The aim is to remove the occlusion from the images automatically. We compare our algorithm to Robust PCA [15] and Local Smoothing discussed in Chapter 2.

3.6.1 Performance of Optimal Solution Estimation

Here, we focus on how well the proposed method, discussed in Section 3.3.2, estimates the optimal solution. For these experiments, we used 698 synthetic face images (in the size of $64 \times 64$), which had been captured from different view points and with varying lighting conditions [58, 59]. The intensity of the images are normalized from 0 to 1. We randomly selected a test image and overlaid a constant intensity noise patch at an arbitrary location. The reason we used constant intensity noise is that it is usually more malicious than other types of noise. The intensity value of the patch is also selected randomly from 0 to 1. The remaining 697 images are used as training data to compute the basis vectors, $\mathbf{U}$, in Eq. 3.15. Actually $\mathbf{U}$, which is computed using local Weighted PCA (WPCA) [45], contains the basis vectors for the local tangent of the test image.
Fig. 3.9. Occlusion images in the first row and their reconstruction Results in the second row

Fig. 3.9 shows the 5 test images in the first row and their reconstruction results in the second row. The first row of Fig. 3.10 plots the sum of the absolute value of the foreground pixels ($|FG|$ in Eq. 3.23) of the corresponding test images through the iteration. The second row plots their reconstruction error. As shown in Fig. 3.9 and Fig. 3.10, the proposed optimal solution estimation method performs quite well by choosing a solution close to the real optimal solution.

3.6.2 Performance for Sub-Block Computation

In an effort to reduce computation time, we utilize sub-block computation. However, there are limitations given its effect on reconstruction. Here, we discuss this further using experimental data. Fig. 3.11 shows the image denoising results obtained by repeating the test 10 times with different test images. Fig. 3.11-(a) and Fig. 3.11-(b) show the original noise-free face images and their constant intensity noise overlaid ones, respectively. Fig. 3.11-(c) shows the reconstruction results obtained by applying original PCA. We chose the number of basis vectors which minimizes the $\ell_2$ norm error between the original images and reconstructed ones. Fig. 3.11-(d) shows the reconstruction results
Fig. 3.10. The first row plots $|F_{G_i}|$’s and the second row plots their reconstruction error through the iteration of the 5 test images in Fig. 3.9. The red dot indicates the position of the selected optimal solution.

from the proposed parameter-free LARS without Haar wavelet decomposition (denoted as 0 level Haar wavelet decomposition). In Fig. 3.11, (e) $\sim$ (g) have reconstruction results of the LARS algorithm with Haar wavelet decomposition of the 1$^{st}$ level, 2$^{nd}$ level, and 3$^{rd}$ level, respectively. As shown in the figure (d)$\sim$(f), the proposed LARS method with Haar wavelet decomposition of 0 level through 2$^{nd}$ level produces very similar results in terms of human evaluation. When we decompose the image further to 3$^{rd}$ level, it produces poor reconstruction results for some of the test images.

Table 3.2 shows the quantitative comparison of the reconstruction results. As shown in Fig. 3.11 and Table 3.2, the LARS method with Haar wavelet decomposition yields very similar performance for certain levels of decomposition (2$^{nd}$ level decomposition in the figure) compare to the LARS algorithm without the decomposition. If we decompose the image too many levels, reconstruction results are not satisfactory not only in terms of human evaluation but also the reconstruction error based on $\ell_2$ norm. This is because more decomposition causes less spatial localization.
Fig. 3.11. Image denoising results of face images. (a) Original images. (b) Occlusion overlaid images of (a). (c) PCA reconstruction (d) LARS reconstruction without Haar wavelet decomposition. (e) ~ (g) LARS reconstruction by applying the proposed Haar wavelet decomposition of level 1 (4 subbands, size of $32 \times 32$), level 2 (16 subbands, size of $16 \times 16$) and level 3 (64 subbands, size of $8 \times 8$), respectively.

<table>
<thead>
<tr>
<th></th>
<th>Occ. Imgs</th>
<th>5.22</th>
<th>7.73</th>
<th>8.00</th>
<th>3.45</th>
<th>10.62</th>
<th>10.28</th>
<th>15.20</th>
<th>8.65</th>
<th>13.32</th>
<th>9.29</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b)</td>
<td>PCA</td>
<td>4.60</td>
<td>6.63</td>
<td>6.92</td>
<td>3.23</td>
<td>9.07</td>
<td>7.31</td>
<td>11.14</td>
<td>7.12</td>
<td>10.90</td>
<td>7.73</td>
</tr>
<tr>
<td>(c)</td>
<td>LARS (level 0)</td>
<td>2.17</td>
<td>1.65</td>
<td>2.51</td>
<td>3.06</td>
<td>2.08</td>
<td>1.82</td>
<td>1.99</td>
<td>1.09</td>
<td>4.55</td>
<td>2.46</td>
</tr>
<tr>
<td>(d)</td>
<td>LARS (level 1)</td>
<td>2.19</td>
<td>1.83</td>
<td>2.54</td>
<td>2.79</td>
<td>2.33</td>
<td>2.07</td>
<td>1.89</td>
<td>1.66</td>
<td>4.67</td>
<td>2.56</td>
</tr>
<tr>
<td>(e)</td>
<td>LARS (level 2)</td>
<td>1.96</td>
<td>2.05</td>
<td>3.16</td>
<td>2.67</td>
<td>2.38</td>
<td>2.81</td>
<td>2.51</td>
<td>2.33</td>
<td>4.76</td>
<td>3.46</td>
</tr>
<tr>
<td>(f)</td>
<td>LARS (level 3)</td>
<td>2.22</td>
<td>3.24</td>
<td>3.59</td>
<td>2.50</td>
<td>4.68</td>
<td>5.36</td>
<td>4.29</td>
<td>2.97</td>
<td>6.03</td>
<td>4.95</td>
</tr>
</tbody>
</table>

Table 3.1. Reconstruction Error of the test images from the second row to the seventh row of Fig. 3.11. $\| \cdot \|_2$ represents $\ell_2$ norm.
3.6.3 Occlusion Handling for Noisy Data

In this section, we show the experimental results of noisy data cleaning by comparing our parameter-free LARS algorithm to Robust PCA and Local Smoothing. Experiments were conducted using three image data sets: the synthetic face images used in Section 3.6.1, a human walking video sequence (109 frames, in the size of 60 × 40) and a ballet video sequence (166 frames, in the size of 50 × 45). The intensity of these images are also normalized from 0 to 1.

For these experiments, we made a noisy test data set from each image data set by overlaying an occlusion to a part of each image. We randomly selected approximately 30% of the images from each data set (210 images out of face data set, 33 images out of human walking data set, and 50 images out of ballet data set) and overlaid a constant intensity noise patch at a random location. The size of the occlusion patch is 20 × 20 pixels for the face data set, 15 × 15 pixels for the ballet data set, and 10 × 10 pixels for the human walking data set. The intensity value of the occlusion patch is randomly determined from 0 to 1.

To apply the LARS algorithm to the noisy data cleaning problem, we exploited the idea of Local smoothing using WPCA [45]. For each image, we compute the local tangent (\( U \) in Eq. 3.15) using WPCA and apply the proposed parameter-free LARS algorithm. Fig 3.12 shows the reconstruction results of the three image data sets. In Fig. 3.12, (a) depicts the 10 occlusion overlaid images selected from the noisy test data set, and (b) through (d) show the reconstruction results of the LARS occlusion handling method proposed in this paper, the Local Smoothing algorithm proposed in [45], and
Robust PCA proposed in [15], respectively. For the results of (c) and (d), we ran the algorithms several times by varying the parameters which the algorithms need, and the best results were selected by considering the reconstruction error based on the $\ell_2$ norm and visual evaluation.

Table 3.2 shows the quantitative results of the reconstruction error based on $\ell_2$ norm. It contains the average and the standard deviation of the reconstruction errors for each test data set obtained by applying the three occlusion handling algorithms. The column of “Only Occ.” contains the reconstruction error computed only using the 30% of the occlusion overlaid images from each test data set, and the column of “All” contains the reconstruction error computed using all of the images in each test data set. It is inevitable that non-occlusion images are degraded while the occlusion handling algorithms deal with the occlusion images. We separate the error into “Only Occ.” and “All” in order to measure how the algorithms perform not only for the occlusion images but also for all the images. In most cases, the proposed LARS algorithm and Local Smoothing algorithm preserve the non-occlusion images very well.

As shown in Fig. 3.12 and Table 3.2, the proposed LARS algorithm performs better than the other two methods in terms of visual evaluation as well as $\ell_2$ based quantitative reconstruction error. Our algorithm outperformed the others especially for complicated texture images such as face images.

We also conducted experiments using the video data set used in [15]. The video data set contains 506 images of size of $160 \times 120$ pixels, which captured an outdoor scene for two days. Some of the images include humans in the scene, which are considered as outliers, while the majority of the images contains only the outdoor background. In
In these experiments, we compared the experimental results between the RPCA algorithm and the proposed LARS occlusion handling method with local smoothing. For the LARS algorithm, we decompose each image into 64 subbands of size $15 \times 20$ pixels by applying 3rd level Haar wavelet decomposition. As shown in Fig. 3.13, both of the methods perform similarly for the video images based on human evaluation. It is not easy to provide quantitative comparison between two algorithms because no prototype data set is provided for the computation of the quantitative error.

In general, RPCA loses much of the original detail of the images, which is why RPCA performed poorly in the first three experiments. RPCA was able to perform adequately with the outdoor video set, primarily because in an outdoor scene, the maintenance of detail is not very important. The proposed LARS algorithm performed well in all four experiments, while maintaining a high level of the detail of the original images.

### Table 3.2
Average reconstruction errors based on $\ell_2$ norm and their standard deviation (inside the parenthesis).

<table>
<thead>
<tr>
<th></th>
<th>LARS</th>
<th></th>
<th>Local Smoothing</th>
<th></th>
<th>RPCA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Only Occ.</td>
<td>All</td>
<td>Only Occ.</td>
<td>All</td>
<td>Only Occ.</td>
<td>All</td>
</tr>
<tr>
<td>Face</td>
<td>3.17 (1.71)</td>
<td>2.86 (1.33)</td>
<td>4.70 (1.79)</td>
<td>3.72 (1.62)</td>
<td>6.77 (2.50)</td>
<td>5.14 (1.85)</td>
</tr>
<tr>
<td>Ballet</td>
<td>3.11 (1.31)</td>
<td>2.54 (1.01)</td>
<td>3.76 (1.20)</td>
<td>3.29 (1.13)</td>
<td>4.70 (1.16)</td>
<td>4.49 (1.04)</td>
</tr>
<tr>
<td>Walking</td>
<td>3.31 (0.76)</td>
<td>2.95 (0.72)</td>
<td>4.39 (0.86)</td>
<td>3.86 (0.82)</td>
<td>3.75 (1.09)</td>
<td>3.68 (0.92)</td>
</tr>
</tbody>
</table>
Fig. 3.12. Selected reconstruction results of the face data, the human walking video sequence, and the ballet video sequence. (a) 10 selected occlusion images (b) Reconstruction results of Proposed LARS algorithm, (c) Reconstruction results of the Local Smoothing method proposed in [45], (d): Reconstruction results of Robust PCA [15]
Fig. 3.13. (a) 20 selected occlusion overlaid images from video sequence. (b) LARS reconstruction. (c) Absolute images between (a) and (b). (d) RPCA reconstruction. (e) Absolute images between (a) and (d).
Chapter 4

Spectral Clustering for Robust Motion Segmentation

In this chapter, we deal with the issues related to the matrix factorization methods for motion segmentation. We first show that the shape interaction matrix can be extracted from the trajectory matrix using QR decomposition with pivoting, an idea that was briefly mentioned in [14]. As a by-product we give a simple and clean proof of the subspace separation theorem described in [31]. We then observe that the shape interaction matrix is very similar to the weight matrix used for graph partitioning and clustering [55] [23] [17] [67], and the motion segmentation problem can be cast as an optimal graph partitioning problem. To this end, we apply the spectral k-way clustering method [23] [17] to the shape interaction matrix to transform it into near-block diagonal form. In our particular application, we use QR decomposition based technique for cluster assignment. This technique at the same time also provides confidence levels of the cluster membership for each feature point trajectory. This confidence levels provide a more robust cluster assignment strategy. We assign a feature point directly to a cluster when it has a high confidence level for the cluster compared to those for other clusters. Using the assigned feature points in each cluster, we compute a linear subspace in the trajectory space. Because the other feature points have lower confidence levels, their cluster memberships are determined by their distances to each of the linear subspaces.
Our experiments on both synthetic data sets and real video images have shown that this method is very reliable for motion segmentation even in the presence of severe noise.

4.1 Matrix Factorization Method for Shape Recovery and Motion Segmentation

Tomasi and Kanade [61] first introduced a factorization method which has been widely used in shape recovery and motion segmentation. In this method, given a video sequence, a group of $N$ feature points are tracked along the $F$ frames. The 2D coordinates are stacked by yielding a matrix, $P$, whose row corresponds the coordinates of each frame and whose column corresponds to those of each feature point.

Suppose that we are given a set of $N$ feature points tracked through $F$ frames, we can construct a feature trajectory matrix $P \in \mathbb{R}^{2F \times N}$ where the rows correspond to the $x$ or $y$ coordinates of the feature points in the image plane and the columns correspond to the individual feature points. The factorization method decomposes the matrix, $P$, into two matrices which, respectively, represent camera motion, $M$, and object shape, $S$. Eq. 4.1 shows this factorization. The method deals with a single static object viewed by a moving camera.
\[
\begin{align*}
P &= MS \\
\begin{bmatrix}
u_{11} & \cdots & u_{1N} \\
\vdots & \vdots & \vdots \\
u_{F1} & \cdots & u_{FN} \\
v_{11} & \cdots & v_{1N} \\
\vdots & \vdots & \vdots \\
v_{F1} & \cdots & v_{FN}
\end{bmatrix} &= 
\begin{bmatrix}
\begin{bmatrix}i_{1T} \\
\vdots \\
i_{F_T}
\end{bmatrix} & \begin{bmatrix}t_{x1} \\
\vdots \\
t_{xF}\end{bmatrix} \\
\begin{bmatrix}j_{1T} \\
\vdots \\
j_{F_T}\end{bmatrix} & \begin{bmatrix}t_{y1} \\
\vdots \\
t_{yF}\end{bmatrix}
\end{bmatrix} \\
&= \begin{bmatrix}s_1, \cdots, s_N\end{bmatrix} \end{align*}
\] (4.2)

where \((u_{fi}, v_{fi})\) are the feature image position, vectors \(i_f^T = [i_x, i_y, i_z], j_f^T = [j_x, j_y, j_z]\) are the first two rows of the rotation matrix at instant \(f\), and \((t_{xf}, t_{yf})\) are the \(X\) and \(Y\) coordinates of the position of the object’s coordinate frame, in the camera frame, at the same instant. The rank of matrix \(P\) is at most 4 because the ranks of \(M\) and \(S\) are at most 4. If we decompose the matrix \(P\) using Singular Value Decomposition (SVD) by taking only the 4 biggest singular values and their corresponding eigen vectors, we can compute the matrix \(M\) and \(S\).

\[
P = U\Sigma V^T = (U \Sigma^{1/2})(\Sigma^{1/2}V^T) = (U \Sigma^{1/2}A)(A^{-1}\Sigma^{1/2}V^T) = MS \quad (4.3)
\]

where, \(A\) is a nonsingular 4x4 matrix.
Costerira and Kanade [14] extended and expanded the above method to account for multiple objects moving independently in a sequence of images. To achieve this, they introduced the shape interaction matrix, $Q = VV^T$, where $V$ is obtained from $U\Sigma V^T$ (the SVD of $P$). It is invariant to both the object motions and the selection of coordinate systems. It is assumed that the matrix $P$ has $N$ feature points tracked from $K$ objects and the feature points of each object are grouped together, i.e. $P = \{P_1, \cdots, P_K\}$. As discussed in [14], the shape interaction matrix has a block diagonal form as follows:

$$Q = VV^T = diag\{S_1^T \Lambda_1 S_1, \cdots, S_K^T \Lambda_K S_K\}$$  \hspace{1cm} (4.4)

where $S_i$ is a shape matrix of object $i$ and $\Lambda_i$ is 4x4 matrix of the moments of inertia of object $i$. Unless the matrix $P$ is a canonical form (which it rarely is), rows and columns of $Q$ are iteratively permuted in such a way that the feature points of the same object are arranged adjacently into blocks, transforming $Q$ into the canonical shape interaction matrix $Q$. However, $Q$ is very sensitive to noise. If $P$ has noise, then the $Q$ will not have a clear block diagonal form.

To overcome this weakness, several methods have been proposed. Below, we highlight some of the more prominent methods.

Gear[21] exploited the reduced row echelon form of the shape interaction matrix to group the points to the linearly independent subspace. If there is no errors, any two columns of the echelon form which have nonzero elements in the same row correspond to points belonging to the same rigid body. The echelon form matrix is represented by a weighted bipartite graph and uses a statistical approach to estimate the grouping of
points to subspaces in the presence of noise by computing which partition of the graph has the maximum likelihood.

Ichimura [28] suggested a motion segmentation method based on discriminant criterion [44] features. The main idea of this method is to select useful features for grouping noisy data. Using a noise-contaminated shape interaction matrix, it computes discriminant criterion for each row of the matrix. The feature points are then divided into two groups by the maximum discriminant criterion, and the corresponding row gives the best discriminant feature. The same procedure is applied recursively to the remaining features to extract other groups.

Wu et. al. [64] proposed an orthogonal subspace decomposition method to deal with the noisy problem of the shape interaction matrix. This method decomposes the object shape space into signal subspaces and noise subspaces. They then use the shape signal subspace distance matrix, $D$, for shape space grouping. This procedure is summarized as follows. First, points are clustered into group fragments using the shape interaction matrix $Q$ by applying simple thresholding or discriminant analysis method described in [28]. Second, the signal subspace projection matrix is calculated for each fragment by applying SVD, which is used to compute the distance matrix $D$. Third, the group fragments are merged into an independent object based on $D$. Finally, the motion and structure for each object are computed based on segmentation.

Kanatani [31] [32] reformulated the motion segmentation problems based on the idea of subspace separation. This approach divides the given $N$ feature points to form $m$ disjoint subspaces $I_i$, $i = 1, \ldots, m$. Provided that the subspaces are linearly independent, the elements $Q_{ij}$ in the shape interaction matrix $Q$ are zero if the point $i$
and the point \( j \) belong to different subspaces. Kanatani also pointed out that even a small amount of noise in one feature point can affect all the elements of \( Q \) in a complicated manner. Based on this, Kanatani proposed noise compensation methods using the original matrix \( W \) rather than the shape interaction matrix \( Q \).

Zelnik-Manor and Irani [66] showed that different 3D motions can also be captured as a single object using previous methods when there is a partial dependency between the objects. They suggested using an affinity matrix \( \bar{Q} \) where

\[
\bar{Q}_{ij} = \sum_k e^{\exp(v_k(i) - v_k(j))^2},
\]

where \( v_k \)'s are the largest eigenvectors of \( Q \). They also dealt with the multi-sequence factorization problems for temporal synchronization using multiple video sequences of the same dynamic scene.

### 4.2 Multibody Motion Segmentation Algorithms

Extending the factorization method discussed in Section 4.1, Costerira and Kanade [14] proposed a multibody factorization method which separates and recovers the shape and motion of multiple independently moving objects in a sequence of images. To achieve this, they introduce a shape interaction matrix which is invariant to both the object motions and the selection of coordinate systems, and suggest a greedy algorithm to permute the shape interaction matrix into block diagonal form. Gear [21] exploited the reduced row echelon form of the shape interaction matrix to group the feature points into the linearly independent subspaces. For Gear’s method, in the noise-free case, any two columns of the echelon form which have nonzero elements in the same row correspond to feature points belonging to the same rigid body. The echelon form matrix can be represented by a weighted bipartite graph. Gear also used a statistical approach to estimate the
grouping of feature points into subspaces in the presence of noise by computing which partition of the graph has the maximum likelihood.

Ichimura [28] suggested a motion segmentation method based on discriminant criterion [44] features. The main idea of the method is to select useful features for grouping noisy data. Using the noise-contaminated shape interaction matrix, it computes discriminant criterion for each row of the matrix. The feature points are then divided into two groups by the maximum discriminant criterion, and the corresponding row gives the best discriminant feature. The same procedure is applied recursively to the remaining features to extract other groups. Wu et. al. [64] proposed an orthogonal subspace decomposition method to deal with the noisy problem of the shape interaction matrix. The method decomposes the object shape space into signal subspaces and noise subspaces. They used the shape signal subspace distance matrix, $D$, for shape space grouping rather than the noise-contaminated shape interaction matrix.

Kanatani [31] [32] reformulated the motion segmentation problems based on the idea of subspace separation. The approach is to divide the given $N$ feature points to form $m$ disjoint subspaces $I_i$, $i = 1, \cdots, m$. A rather elaborated proof was given showing that provided that the subspaces are linearly independent, the elements $Q_{ij}$ in the shape interaction matrix $Q$ is zero if the point $i$ and the point $j$ belong to different subspaces. Kanatani also pointed out that even a small noise in one feature point can affect all the elements of $Q$ in a complicated manner. Based on this fact, Kanatani proposed noise compensation methods using the original data rather than the shape interaction matrix $Q$. 
Zelnik-Manor and Irani [66] showed that different 3D motions can also be captured as a single object using previous methods when there is a partial dependency between the objects. To solve the problem, they suggested to use an affinity matrix $\bar{Q}$ where $\bar{Q}_{ij} = \sum_k \exp(v_k(i) - v_k(j))^2$, where $v_k$’s are the largest eigenvectors of $Q$. They also dealt with the multi-sequence factorization problems for temporal synchronization using multiple video sequences of the same dynamic scene.

### 4.3 Constructing the Shape Interaction Matrix Using QR Decomposition

In this section, we exhibit the block diagonal form of the shape interaction matrix using QR decomposition with pivoting [22]. Assuming that we have $N$ rigidly moving feature points, $p_1, \ldots, p_N$, which are on an image plane which corresponds 3D points over the $F$ frames. Motion segmentation can be interpreted as dividing the feature points $p_i$ into $S$ groups [31] each spanning a linear subspace corresponding to feature points belonging to the same object. We denote the grouping as follows,

$$\{1, \ldots, N\} = \bigcup_{i=1}^{S} \mathcal{I}_i, \quad \mathcal{I}_i \cap \mathcal{I}_j = \emptyset.$$ 

Now define $l_i = |\mathcal{I}_i|$ which is the number of the points in the set $\mathcal{I}_i$, and $k_i = \dim \text{span}\{p_j\}_{j \in \mathcal{I}_i} \leq l_i$ and $P_i = \{p_j\}_{j \in \mathcal{I}_i}$.

Let the SVD of $P_i$ be $P_i = U_i \Sigma_i V_i^t$, where $\Sigma_i \in R^{k_i \times k_i}, i = 1, \ldots, S$. Then $P = [P_1, P_2, \ldots, P_S]$ can be written as,
\[ P = [P_1, P_2, \ldots, P_s] = [U_1\Sigma_1, U_2\Sigma_2, \ldots, U_s\Sigma_s] \]

\[
\begin{bmatrix}
V_1^T & 0 & \cdots & 0 \\
0 & V_2^T & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & V_s^T \\
\end{bmatrix}, \quad (4.5)
\]

where \( \text{rank}(V_i) = k_i \) for \( i = 1, \ldots, s \). Assuming that the \( S \) subspaces \( \text{span}\{p_j\}_{j \in I_i}, i = 1, \ldots, S \) are linearly independent, then the matrix \( [U_1\Sigma_1, U_2\Sigma_2, \ldots, U_s\Sigma_s] \) has a full column rank of \( k = k_1 + \cdots + k_s \). Therefore, an arbitrary orthonormal basis for the row space of \( P \) can be written as \( \Phi \text{diag}(V_1, \cdots, V_s)^T \) for an arbitrary orthogonal matrix \( \Phi \in R^{k \times k} \). Now the shape interaction matrix can be written as

\[
Q = \text{diag}(V_1, \cdots, V_s)\Phi^T\Phi\text{diag}(V_1, \cdots, V_s)^T = \text{diag}(V_1V_1^T, \cdots, V_sV_s^T).
\]

This clearly shows that \( Q_{ij} = 0 \) if \( i \) and \( j \) belong to different subspaces, i.e., if the corresponding feature points belong to different objects.

Rather than using SVD, a cheaper way to compute an orthonormal basis for the row-space of \( P \) is to apply QR decomposition with column pivoting to \( P^T \),

\[
P^TE = \hat{Q}R, \quad (4.6)
\]

where \( E \) is a permutation matrix, and \( \hat{Q} \) has \( k \) columns. It is easy to see that \( \hat{Q}\hat{Q}^T = Q \). In the presence of noise, \( P \) will not exactly have the rank \( k \), but generally
QR decomposition with column pivoting generates a $R$ matrix that reliably reveals the numerical rank of $P$. We can then truncate $R$ by deleting rows with small entries.

4.4 Motion Segmentation

4.4.1 Spectral Multi-way clustering

In the last section, we showed that the shape interaction matrix, $Q \in \mathbb{R}^{N \times N}$ has a block diagonal form when the feature points are grouped into independent subspaces corresponding to $S$ different objects. In general, this grouping is unknown, and we need to find row and column permutations of the matrix $Q$ to exhibit this block diagonal form, and thus assigning the feature points to different objects. A greedy algorithm has been proposed in [14] but it performs poorly in the presence of noise. We now present a more robust method based on spectral graph clustering [55] [23] [17] [67]. It provides a confidence level from which we can further refine the cluster memberships of the feature points.

The absolute value of the $(i, j)$ element of the shape interaction matrix $Q$ measures the similarity of feature points $i$ and $j$. Feature points belonging to the same object will have a high degree of similarity whereas feature points belonging to different object will have a low degree of similarity. In fact, in the noise-free case, feature points in different objects will have zero similarity. Our goal is then to partition the feature points into $S$ groups based on similarity. Let $W = (w_{ij})$ with $w_{ij} = |Q_{ij}|$. For a given partition of the feature points into $S$ groups, we can permute the rows and columns of $W$ so that the rows and columns corresponding to the feature points belonging to the same objects
are adjacent to each other, i.e., we can re-order the columns and rows of the \( W \) matrix such that

\[
W = \begin{bmatrix}
W_{11} & W_{12} & \cdots & W_{1S} \\
W_{21} & W_{22} & \cdots & W_{2S} \\
\vdots & \vdots & \ddots & \vdots \\
W_{S1} & W_{S2} & \cdots & W_{SS}
\end{bmatrix}.
\] (4.7)

We want to find a partition such that \( W_{ii} \) will be large while \( W_{ij}, i \neq j \) will be small, and to measure the size of a sub-matrix matrix \( W_{ij} \) we use the sum of all its elements and denoted as \( \text{sum}(W_{ij}) \). Let \( x_i \) be a cluster indication vector accordingly partitioned with that of \( W \) with all elements equal to zero except those corresponding to rows of \( W_{ii} \),

\[
x_i = [0 \cdots 0, 1 \cdots 1, 0 \cdots 0]^T.
\]

Denote \( D = \text{diag}(D_1, D_2, \cdots, D_S) \) such that \( D_i = \sum_{j=1}^{S} W_{ij} \). It is easy to see that

\[
\text{sum}(W_{ii}) = x_i^T W x_i, \quad \sum_{j \neq i} \text{sum}(W_{ij}) = x_i^T (D - W) x_i.
\]

Since we want to find a partition which will maximize \( \text{sum}(W_{ii}) \) while minimizing \( \text{sum}(W_{ij}), i \neq j \), we seek to minimize the following objective function by finding a set of indicator vectors \( x_i \). The objective function is called min-max cut in [23] [17] which is a generalization of the normalized cut objective function [55] to the multi-way partition case.
\[
\text{MCut} = \frac{x_1^T (D - W)x_1}{x_1^T Wx_1} + \frac{x_2^T (D - W)x_2}{x_2^T Wx_2} + \cdots + \frac{x_S^T (D - W)x_S}{x_S^T Wx_S}
\]

\[
= \frac{x_1^T Dx_1}{x_1^T Wx_1} + \frac{x_2^T Dx_2}{x_2^T Wx_2} + \cdots + \frac{x_S^T Dx_S}{x_S^T Wx_S} - S.
\]

If we define \(y_i = D^{1/2} x_i / \|D^{1/2} x_i\|_2\) and \(Y_S = [y_1, \ldots, y_S]\), we have

\[
\text{MCut} = \frac{1}{y_1^T \hat{W} y_1} + \frac{1}{y_2^T \hat{W} y_2} + \cdots + \frac{1}{y_S^T \hat{W} y_S} - S \quad (4.8)
\]

where \(\hat{W} = D^{-1/2} WD^{-1/2}\) and \(y_i = \frac{D^{1/2} x_i}{\|D^{1/2} x_i\|_2}\). It is easy to see that each vector \(y_i\) is orthogonal to all the other vectors and is normalized to have Euclidean norm one. If we insist that each vector \(y_i\) be constrained to inherit the discrete structure of its corresponding indicator vector \(x_i\), we then solve the combinatorial optimization problem which has been proven to be NP-hard even when \(S = 2\) [55]. The idea of spectral clustering instead is to relax these constraints and allow each vector \(y_i\) to be an arbitrary set of orthonormal vectors. In this case, the minimum of Eq. 4.8 can be achieved by orthonormal basis \(y_1, \ldots, y_S\) of the subspace spanned by the eigenvectors corresponding to the largest \(S\) eigenvalues of \(\hat{W}\). Next we discuss how to assign the feature points to each cluster based on the eigenvectors.

At this point, we should mention that the cluster assignment problem in spectral clustering is not well-understood. Here we follow the approach proposed in [67]. Assuming \(\hat{Y} = [\hat{y}_1, \ldots, \hat{y}_S]^T\) denotes the optimal solution of Eq. 4.8, the vectors \(\hat{y}_i\) can
be used for cluster assignment because \( \hat{y}_i \approx D^{1/2} \hat{x}_i / ||D^{1/2} \hat{x}_i||_2 \), where \( \hat{x}_i \) is the cluster indicator vector of \( i-th \) cluster. Ideally, if \( W \) is partitioned perfectly into \( S \) clusters, then the columns in \( \hat{X} = [\hat{x}_i, \cdots, \hat{x}_S]^T \) of the \( i-th \) cluster are the same, one for the \( i-th \) row and zeros for the others. Two columns of different clusters are orthogonal to each other. Because of this, \( \hat{Y} \) approximately inherits this property: two columns from two different clusters are orthogonal to each other, and those from one cluster are the same. We now pick a column of \( \hat{Y} \) which has the largest norm \( \hat{y}_i \) and orthogonalize the rest of the columns of \( \hat{Y} \) against this column. We then assign the columns to cluster \( i \) whose residual is small. We perform this process \( S \) times. As discussed in [67], it is exactly the same procedure of QR decomposition with column pivoting applied to \( \hat{Y} \). In particular, we compute the QR decomposition of \( Y^T \) with column pivoting

\[
Y^T E = \hat{Q} R = \hat{Q}[R_{11}, R_{12}]
\]

where \( \hat{Q} \) is a \( S \times S \) orthogonal matrix, \( R_{11} \) is a \( S \times S \) upper triangular matrix, and \( E \) is a permutation matrix. Then we compute the matrix \( \hat{R} \) as

\[
\hat{R} = R_{11}^{-1}[R_{11}, R_{12}]P^T = [I_S, R_{11}^{-1}R_{12}],
\]

The matrix \( \hat{R} \in R^{S \times N} \) gives the confidence level of each point to be assigned to each cluster. Notice that the columns correspond to the feature points and the rows correspond to the clusters. The cluster membership of each feature point is determined by the row index of the largest element (based on the absolute value of the corresponding
column of $\hat{R}$). This provides us with a baseline for motion segmentation which accounts for the presence of noise.

This presents a problem however. Although we can assign a point to a cluster with high confidence when there is a very dominantly high confidence value in the corresponding column, we can not assign a point to a cluster with high confidence when two or more values in a column are very close to each other. Table 4.4.1 shows an example of the matrix $\hat{R} \in \mathbb{R}^{3 \times 10}$ that has 10 points extracted from 3 objects. The last row of the table shows the cluster membership of each point assigned by the row index of the highest absolute value. For instance, the point $p_1$ is assigned to cluster 2 because the second row value (0.329), is greater than the other row values (0.316 and 0.203). However, we cannot have much confidence of its membership because there is no dominant value in the corresponding column.

Although this is a problem, we do have a baseline for motion segmentation which accounts for the presence of noise. Further improvement can be achieved as we discuss next.

<table>
<thead>
<tr>
<th>Cluster ID</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
<th>p6</th>
<th>p7</th>
<th>p8</th>
<th>p9</th>
<th>p10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 1$</td>
<td>0.32</td>
<td>0.35</td>
<td>0.88</td>
<td>0.33</td>
<td>0.46</td>
<td>0.56</td>
<td>0.09</td>
<td>0.28</td>
<td>0.07</td>
<td>0.12</td>
</tr>
<tr>
<td>$k = 2$</td>
<td>0.33</td>
<td>0.34</td>
<td>0.03</td>
<td>0.37</td>
<td>0.01</td>
<td>0.06</td>
<td>0.19</td>
<td>0.71</td>
<td>0.82</td>
<td>0.83</td>
</tr>
<tr>
<td>$k = 3$</td>
<td>0.20</td>
<td>0.02</td>
<td>0.03</td>
<td>0.17</td>
<td>0.57</td>
<td>0.56</td>
<td>0.78</td>
<td>0.13</td>
<td>0.09</td>
<td>0.11</td>
</tr>
</tbody>
</table>

| Assigned Cluster | 2  | 1  | 1  | 2  | 3  | 1  | 3  | 2  | 2  | 2  |

Table 4.1. An example of the matrix $\hat{R}$. There are 10 points extracted from 3 objects. The last row shows the assigned cluster
4.4.2 Refinement of cluster assignment for motion segmentation

Baseline spectral clustering shows its robustness for a noisy environment in spite of its hard clustering (assigning each point to a cluster even though it does not have high confidence for it). This method alone, however, can sometimes fail in presence of severe noise. In this section, we discuss a two-phase approach whereby in phase one we assign the cluster memberships for those feature points with high confidence levels, and in phase two we construct linear subspaces for each cluster based on the high confidence feature points, and assign the rest of the feature points by projecting them onto these subspaces.

Our approach proceeds as follows. After computing $\hat{R}$ discussed in the previous section, the points of high confidence for each clusters are selected. Let’s define $P_i = [p_{i1}, \ldots, p_{iN_i}]$ as the trajectory points in the cluster $i$. One of the easiest methods is applying a threshold to the values of each column. If the highest value in the column is greater than the threshold, the point is assigned to the corresponding cluster. Otherwise, the point is assigned to cluster 0 which is in a pending state to decide its cluster membership. Referring to the pending points as $P_0 = [p_{01}, \ldots, p_{0N_0}]$, the next step is to compute subspace(2D) for $p_{i1}, \ldots, p_{iN_i}, i = 1, \ldots, S$ using Principal Component Analysis (PCA). Thus, with $U_i$ denoting a subspace basis for the cluster $i$, we finally determine the cluster membership of each pending point by computing the minimum distance from the point to the subspaces.

$$\hat{\theta}_j = \arg \min_i ||p_{0j} - (c_i + U_i U_i^T (p_{0j} - c_i))||_2^2,$$
Fig. 4.1. The two synthetic video sequences used in [32] and [33] respectively. (a). 20 green dots are background pixels and 9 foreground points (b). 24 background points and 14 foreground points. The foreground pixels are connected with lines.

where $j = 1, \ldots, S$ and $c_i = \sum_{j=1}^{N_i} p_{ij}$.

The point $p_{0j}$ is assigned to the cluster $\hat{\theta}_j$.

4.5 Experimental Results

4.5.1 Synthetic Data

Fig. 4.1 shows two synthetic image sequences used for performance evaluation. Actually these images are used in [32] and [33]. Fig. 4.1-(a), denoted as Synthetic 1, has 20 background points (green dots) and 9 foreground points (red dots). Fig. 4.1-(b), denoted as Synthetic 2, has 24 background points and 14 foreground points. The foreground points are connected by lines for visualization purpose.

We performed experiments using not only the original tracking data but also the data added by independent Gaussian noise of mean 0 and standard deviation $\sigma$ to the
coordinates of all the points. For the noise data, we generated 5 sets for each $\sigma = 1, 2, 3, 4$, and computed the misclassification rate by simply averaging the 5 experiment results.

We compared the two methods proposed in this paper (One is $k$-way Min-Max cut clustering in Section 4.4.1 denoted as Method 1, and the other is a combination of the $k$-way Min-Max cut clustering and clustering refinement using subspace projection in Section 4.4.2 denoted as Method 2) to the Multi-stage optimization proposed in [33] denoted as Multi-Stage. Table 4.2 shows that the misclassification rates of the three methods over the different noise levels ($\sigma = 0, 1, 2, 3, 4$). Method 2 and Multi-Stage performed better than Method 1. In fact, these two methods were almost perfect for the sequences.

<table>
<thead>
<tr>
<th>Video Seq.</th>
<th>noise</th>
<th>$\sigma = 0$</th>
<th>$\sigma = 1$</th>
<th>$\sigma = 2$</th>
<th>$\sigma = 3$</th>
<th>$\sigma = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic 1</td>
<td>Method 1</td>
<td>0.0</td>
<td>1.4</td>
<td>1.4</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Synthetic 1</td>
<td>Method 2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Synthetic 1</td>
<td>Multi-Stage</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Synthetic 2</td>
<td>Method 1</td>
<td>8.2</td>
<td>10.6(1.6)</td>
<td>11.7(2.1)</td>
<td>11.7(3.6)</td>
<td>13.2(1.7)</td>
</tr>
<tr>
<td>Synthetic 2</td>
<td>Method 2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.59(1.3)</td>
</tr>
<tr>
<td>Synthetic 2</td>
<td>Multi-Stage</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.59(1.3)</td>
</tr>
</tbody>
</table>

Table 4.2. Misclassification rate (%) for the two synthetic sequences. The values in parentheses are the standard deviation values. Method 1 is the $k$-way Min-Max cut clustering discussed in Section 4.4.1 and Method 2 is the $k$-way Min-Max cut clustering plus the clustering refinement using subspace projection discussed in Section 4.4.2. Multi-Stage is the Multi-stage optimization proposed in [33].
4.5.2 Real Video Sequences

We experimented with the real video sequences used in [33]. In all these sequences, both the foreground object and the background are simultaneously moving because the camera is moving. The video sequences are denoted as video1, video2 and video3 respectively. In addition, by overlaying the foreground feature points in video1 to video2, we synthesized a fourth test video sequence which has 2 moving objects and a background denoted as video4. Fig. 4.2 shows 5 selected frames for each of the four sequences.

We also performed experiments using not only the original tracking data but also the data added by independent Gaussian noise of mean 0 and standard deviation $\sigma$ to the coordinates of all the points. For the noise data, we generated 5 sets for each $\sigma = 3, 5, 7, 10$, and computed the misclassification rate by simply averaging the 5 experiment results.

Table 4.3 shows the misclassification rates of the three methods over the different noise levels ($\sigma = 0, 3, 5, 7, 10$). Method 1 performs very well for a noise-free environment as predicted. However, with noisy environment, this method performs poorly as it misclassifies some points. Multi-Stage performs very well for video1 through video3 even in the presence of noise. However, it performs poorly for video4 possibly because of the presence of two moving objects. Based on our experiments, this method also suffers from the local minima problem. That is, using the same data, it yields different results based on the initialization. That is the reason the standard deviation for the method is so high shown in Table 4.3.
Method 2 performs well for all four video sequences, even with multiple objects and severe noise.

<table>
<thead>
<tr>
<th>Video Seq.</th>
<th>noise</th>
<th>$\sigma = 0$</th>
<th>$\sigma = 3$</th>
<th>$\sigma = 5$</th>
<th>$\sigma = 7$</th>
<th>$\sigma = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>video1</td>
<td>Method 1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>Multi-Stage</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>video2</td>
<td>Method 1</td>
<td>0</td>
<td>1.6(1.2)</td>
<td>1.6(1.2)</td>
<td>1.6(1.2)</td>
<td>2.9(1.7)</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>Multi-Stage</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>7.3(16.3)</td>
</tr>
<tr>
<td>video3</td>
<td>Method 1</td>
<td>0.0</td>
<td>2.5(0.01)</td>
<td>2.5</td>
<td>1.3</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>Multi-Stage</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>video4</td>
<td>Method 1</td>
<td>0.0</td>
<td>0.7(1.6)</td>
<td>3.4(4.7)</td>
<td>8.3(5.0)</td>
<td>9.6(6.5)</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7(1.6)</td>
</tr>
<tr>
<td></td>
<td>Multi-Stage</td>
<td>0.0</td>
<td>4.1(9.3)</td>
<td>8.2(9.6)</td>
<td>16.2 (13.2)</td>
<td>19.23 (9.87)</td>
</tr>
</tbody>
</table>

Table 4.3. Misclassification rate (%) for the real video sequences. The values in parentheses are the standard deviation values.
Fig. 4.2. Real video sequences with the feature points. 1\textsuperscript{st} row: \textit{video1}, 2\textsuperscript{nd} row: \textit{video2}, 3\textsuperscript{rd} row: \textit{video3}, 4\textsuperscript{th} row: \textit{video4} (foreground feature points in \textit{video1} are overlaid onto \textit{video2}). Red dots correspond to the background while green dots correspond to the foreground. The yellow cross marks in \textit{video4} represent the foreground feature points of \textit{video1}.

Fig. 4.3. Graph for misclassification rate. Graph of \textit{video1} is not depicted here because all of the methods perform perfectly. \textbf{Method 1}: Dashed-dot blue line, \textbf{Method 2}: Red line and \textbf{Multi-Stage}: Dashed green line.
Chapter 5

Summary and Conclusions

Chapter 2 focused on outlier handling and noise reduction for a set of noisy sample points using local linear smoothing. We showed that the local smoothing method is very effective not only in reducing Gaussian noise but also for effective handling of outliers. The performance of this method, however, depends on two parameters: the number of neighbors $k$ used in local smoothing and the $\gamma$ used for weight computation. As far as $\gamma$ is concerned, experiments show that relatively low values for $\gamma$ yield good results for the case when most points in the data are corrupted by Gaussian noise. If many points are involved in a manifold and some outliers are added to the data, a relatively high $\gamma$ tends to produce better results. For choosing the parameter $k$, the neighbor points selected should approximately form a linear affine subspace. However, the results are relatively insensitive to the choice of $k$ as is shown in Fig. 2.6. The experiments using the image data sets also show that the proposed local smoothing method has a potential to remove occlusions overlaid on images. We further show that the proposed methods can be utilized with various manifold learning methods as a preprocessing procedure to obtain a more accurate reconstruction of the underlying nonlinear manifolds.

In Chapter 3, we proposed a parameter-free image denoising method using the LARS algorithm, local WPCA and a statistical model selection criterion along with a
thresholding technique. The method adapts the $\ell_1$ penalized optimization method proposed in [63] which identifies and updates only the noise corrupted pixels simultaneously. The method proposed in [63], however, requires a parameter to control the size of the pixels to be updated. We implemented the image denoising method without the parameter by converting the problem into a LARS problem and solving it using the LARS algorithm. The LARS algorithm yields a family of solutions covering one updated pixel to all of the updated pixels. Among the solutions, we determined the final optimal solution utilizing a thresholding algorithm and AIC. The experimental results show that the proposed method performs quite well in the most cases by selecting a solution in the vicinity of the global minimum of reconstruction error without a parameter to control the number of pixels to be updated.

We also proposed a method for decomposing an image into several independent subbands (blocks) of the same size, by slightly modifying the original Haar wavelet analysis. It is possible to decrease the time complexity by applying each subband to the algorithm independently. The method, however, might lose the spatial localization (low resolution) as the decomposition level gets deeper and deeper. The proposed image decomposition method provides relatively similar results for certain levels of decomposition without losing too much spatial resolution. If the decomposition level is too deep, we lose too much spatial details and obtain poor reconstruction results. The proposed decomposition method is suitable for parallel computing because the subbands are independent of each other as proven in Section 3.5.

We utilized the occlusion handling algorithm as a preprocessing tool for cleaning training data, which contain occlusion overlaid images, for an automatic learning system.
The experimental results show that the proposed method is very promising for this purpose.

In chapter 4, we proposed a robust motion segmentation method using matrix factorization, subspace separation and spectral graph partitioning. First, we mathematically proved that the shape interaction matrix can be computed using QR decomposition which is more effective than SVD. We solved the motion segmentation problem using the spectral graph clustering technique because the shape interaction matrix has a very similar form to the weight matrix of the graph. We apply the Spectral Relaxation K-way Min-Max cut clustering method [23] [17] to the shape interaction matrix. It provides a relaxed cluster indication matrix. QR decomposition was then applied to the matrix, which generated a new cluster indication matrix, to determine the cluster membership of each point. The values of the new cluster indication matrix reflect a confidence level for each point to be assigned to clusters. This method yields a good performance in a noise free environment, but it is sensitive to noise. We proposed a robust motion segmentation method by combining the spectral graph clustering and subspace separation to compensate for the noise problem. Initially, we assigned only points of high confidence to clusters based on the cluster indication matrix. We computed the subspace for each cluster using the assigned points. We finally determined the membership of the other points, which are not assigned to a cluster, by computing the minimum residual when they are projected to the subspace.

We used the two synthetic image sequences and three real video sequences from [33] and a generated a fourth real video sequence, all of which were overlaid with various severity of noise. The proposed method performed almost to perfection with the synthetic
image sequences, and performed very well with the real video sequences even with severe noise and with multiple objects. In comparison, although the Multi-Stage optimization method [33] performed reasonably well with both synthetic and real video sequences, it performed very poorly in the presence of multiple objects and suffers from the local minima problem.

5.0.3 Contributions

We proposed a robust outlier handling and noise reduction method that can be used in conjunction with nonlinear manifold learning methods as a preprocessing procedure to obtain a more accurate reconstruction of the underlying nonlinear manifold. A closed-form solution of Weighted PCA (WPCA), which is discussed in Section 2.2, is proposed for the method by only considering object-level outliers. With both object-level and feature-level outliers, WPCA solves the optimization problem in an iterative fashion, see [15] for example. We successfully applied the local smoothing method to the image occlusion handling problem and point-based rendering problem, and showed the robustness of this method.

We also proposed a robust parameter-free image occlusion handling algorithm. The main contributions of this research include: 1) We showed that the image denoising problem can be solved using the LARS algorithm; 2) We proposed a parameter-free approach to image denoising utilizing a thresholding technique and the statistical model selection criterion of AIC; 3) We demonstrated how sub-block decomposition affect computation time; and 4) We showed that the proposed LARS algorithm can be successfully
used as a preprocessing tool for cleaning noisy data which contains occlusion overlaid images.

In Chapter 4, we proposed a robust motion segmentation method using matrix factorization, subspace separation and spectral graph partitioning. We showed that the shape interaction matrix can be computed using QR decomposition instead of Singular Value Decomposition (SVD), where QR decomposition is more stable and efficient than SVD. A spectral graph clustering technique is applied to the shape interaction matrix for the final grouping of the feature points into independent objects.

5.0.4 Future Works

Several issues deserve further investigation. First, there are three issues for local smoothing for manifold learning in Chapter 2: 1) automatic selection of parameters such as the number of neighbors and $\gamma$ for weight computation; 2) exploiting RPCA [15] for image data which assigns weights to the pixels and/or images; and 3) exploiting different metrics such as tangential distance to compute $k$ nearest neighbors for image data sets rather than the Euclidean distance.

The LARS algorithm still requires a high computation time because the size of matrix $X$ in Eq. 3.12 is very large (the number of pixels $\times$ the number of pixels), computing the solutions one pixel at a time. If we compute the solutions several pixels at a time, we can reduce the computation time significantly. We also used Euclidean distance to compute the nearest neighbors for each local tangent. It is possible to exploit different metrics such as Kullback-Leibler distance, tangential distance, etc. Sophisticated thresholding algorithms can also be used for optimal solution estimation.
For motion segmentation using matrix factorization as discussed in Chapter 4, finding a proper rank for SVD in matrix factorization is very critical for performance. Automatic estimation of the rank of SVD should be studied further. Current multi-body motion segmentation algorithms using matrix factorization deal with rigid objects. It is plausible if we can also deal with motion segmentation of articulate objects. Because the shape of each object is constantly changing, a fundamental question is raised: What is the definition of an independently moving object? Once a definition is established, we can then begin to address the problems associated with motion segmentation of articulate objects, and it would be considered breakthrough work for a multitude of motion related areas, such as real-life human motion analysis and the like.
Appendix A

Computation of the Equiangular vector $u_A$, $w$ and $A_A$

$u_A$ is defined as the unit vector with equal angles, less than $90^\circ$, with the columns of $X_A$. Therefore, we can obtain the following facts. $u_A$ can be expressed by a linear combination of the columns of $X_A$, and all the inner products between $u_A$ and the columns of $X_A$ should be the same.

\begin{align*}
    u_A^T u_A &= 1 \\ 
    u_A &= X_A w \\ 
    X_A^T u_A &= A_A 1_A 
\end{align*}

where $A_A$ is cosine value between $u_A$ and the columns of $X_A$.

From this, we obtain Eq. A.4 by applying Eq. A.2 to Eq. A.1.

\begin{equation}
    w^T X_A^T X_A w = 1
\end{equation}

We obtain $w$ by substitute $X_A w$ in Eq. A.2 for $u$ in Eq. A.3 as shown in Eq. A.5.
\[ X_A^T X_A w = A_A 1_A \]

\[ w = (X_A^T X_A)^{-1} A_A 1_A = G_A^{-1} A_A 1_A \]  

(A.5)

From here, we apply Eq. A.5 to Eq. A.4, deriving Eq. A.6.

\[ A_A^2 1_A^T (X_A^T X_A)^{-1} 1_A = 1 \]

We only need positive \( A_A \) because we allow the angle between \( u_A \) and each column of \( X_A \) to be less than 90°. Therefore, we can obtain \( A_A \).

\[ A_A = (1_A^T (X_A^T X_A)^{-1} 1_A)^{-\frac{1}{2}} = (1_A^T G_A^{-1} 1_A)^{-\frac{1}{2}} \]  

(A.6)
2D Haar Wavelet Analysis

\[
\phi(x, y) = \begin{cases} 
1 & \text{if } 0 \leq x < 1 \text{ and } 0 \leq y < 1 \\
0 & \text{otherwise}
\end{cases} \tag{B.1}
\]

\[
\phi(x, y) = \phi(x)\phi(y) \tag{B.2}
\]

Suppose a function \( f_j(x, y) \),

\[
f_j(x, y) = \sum_{k_1 \in \mathbb{Z}} \sum_{k_2 \in \mathbb{Z}} a_{k_1, k_2}^j \phi(2^j x - k_1, 2^j y - k_2). \]

Then \( f_j(x, y) \) can be decomposed into \( x \) and \( y \) components as shown in Eq. B.3.

\[
f_j(x, y) = \sum_{k_1 \in \mathbb{Z}} \sum_{k_2 \in \mathbb{Z}} a_{k_1, k_2}^j \phi(2^j x - k_1) \phi(2^j y - k_2) \tag{B.3}
\]

\[
= \sum_{k_1 \in \mathbb{Z}} a_{k_1}^j \phi(2^j x - k_1) \sum_{k_2 \in \mathbb{Z}} a_{k_2}^j \phi(2^j y - k_2)
\]

Next, we use Eq. 3.38, we obtain,
\[ f_j(x, y) = (w_{j-1}^x + f_{j-1}^x)(w_{j-1}^x + f_{j-1}^x) \] (B.4)

\[ = w_{j-1}^x w_{j-1}^y + w_{j-1}^x f_{j-1}^y + f_{j-1}^x w_{j-1}^y + f_{j-1}^x f_{j-1}(x, y), \] (B.5)

where the superposition character \( x \) and \( y \) mean the direction of 1D Haar wavelet analysis.
References


Vita

JinHyeong Park was born on June 19, 1969 in Kangkyong, Republic of Korea. He received the BS degree in the department of Computer Science at the Chungnam National University, Republic of Korea, in March, 1994. In the middle of his undergraduate study, he fulfilled mandatory military service from April 1989 to October 1990.

He joined Pos-Data Corporation in Kwanyang on January 1994, and worked as an software engineer until January 1996. Then, he joined the Chungnam National University to pursue a Master’s degree in the department of Computer Science, and earned his MS degree in March 1998. He was a lecturer in the department of Computer Science at Chungnam National University from March 1998 to December 1998. He enrolled in the Ph.D. program in the department of Computer Science at The Pennsylvania State University, University Park, in January 1999.

His research interests include Computer Vision, Digital Image Processing and Machine Learning. He is a student member of the IEEE Computer Society.