ABSTRACT

Title of dissertation: HARMONIC ANALYSIS INSPIRED DATA FUSION FOR APPLICATIONS IN REMOTE SENSING

Timothy J. Doster, Doctor of Philosophy, 2014

Dissertation directed by: Professor John J. Benedetto
Department of Mathematics

Professor Wojciech Czaja
Department of Mathematics

This thesis will address the fusion of multiple data sources arising in remote sensing, such as hyperspectral and LIDAR. Fusing of multiple data sources provides better data representation and classification results than any of the independent data sources would alone. We begin our investigation with the well-studied Laplacian Eigenmap (LE) algorithm. This algorithm offers a rich template to which fusion concepts can be added. For each phase of the LE algorithm (graph, operator, and feature space) we develop and test different data fusion techniques. We also investigate how partially labeled data and approximate LE preimages can used to achieve data fusion. Lastly, we study several numerical acceleration techniques that can be used to augment the developed algorithms, namely the Nyström extension, Random Projections, and Approximate Neighborhood constructions. The Nyström extension is studied in detail and the application of Frame Theory and $\Sigma\Delta$ Quantization is proposed to enrich the Nyström extension.

HARMONIC ANALYSIS INSPIRED DATA FUSION FOR APPLICATIONS IN REMOTE SENSING

by

Timothy J. Doster

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2014

Advisory Committee:
Professor John J. Benedetto, Chair
Professor Wojciech Czaja, Co-Chair
Professor Radu V. Balan
Professor Ramani Duraiswami
Professor Kasso A. Okoudjou

Dedication

To my loving and brilliant wife Karen. You saw me through this thesis as well as graduate school, without your support I don't think I would have made it.

Acknowledgments

First, I would to thank my advisors John and Wojtek, without your guidance over the past five years I would not be at this position today. I had not even considered Harmonic Analysis or the Norbert Wiener Center until I received an email from you during my first year at Maryland proposing an independent study. Since that day you have both been actively interested in my education, research, well being, and personal life.

Wojtek, thank you for starting out every meeting by asking how I was doing. It was wonderful to see that you not only cared about my mathematical output but that my life was in a good place. Thank you for providing me funding for much of my time here at Maryland, it was not only a relief to avoid teaching but a pleasure to get to study so many interesting problems. Lastly, thanks for your encouraging words as I hit wall after wall writing my thesis; I think it went something like this, "If you don't finish, they won't hire you, and they will never hire you, so you must finish, go finish now."

John, thank you for always checking in on me in the lab in the morning (or were you just looking for the coffee?). I learned so much in your classes, from abstract harmonic analysis to the finer points of billiards, that I will leave Maryland with not only deep understanding of my thesis problem but a broad understanding of the greater subject.

I would also like to acknowledge my other committee members Kasso, Radu, and Ramani. I had the privilege to have a class taught by each of you and then was

lucky enough to coordinate it so you could all be on my committee. Thank you for reviewing my thesis, and for all you taught me in the classroom.

Aside from the many wonderful classes taught by my committee members, I would like to acknowledge the formative instruction I received in scientific computing from Professors O'Leary and Elman, pattern recognition from Professor Chellappa, and in applied harmonic analysis from Professor Easley.

I don't think I, or anyone else in the AMSC program at Maryland, can fully thank Alverda for guiding us toward graduation. Seeing her search the building for me before my defense to give me the very important paper work I needed to actually graduate reminded me what great person she is.

Meeting not only Alverda, but Konstantina, the then director of the AMSC program, sealed the deal for me coming to Maryland. Thank you Konstantina for your kind words, and for always remembering your students.

I would have never succeeded at Maryland without excellent teachers and advisors at my undergrad, RIT; thank you Professors Wiandt, Messinger, Basener, Barth-Hart, and Ross.

I was fortunate enough to spend three excellent summers at Naval Research Lab learning to transform theory to practice, thank you Al, Eric, and Brian for the opportunity.

Collaboration is the heart of great research. I would like to acknowledge all of my fellow students in the Norbert Wiener Center. Thank you for all the thoughts, ideas, and help procrastinating: Alex, Kevin D., Kevin H., Matt B., Matt G., Chae, Travis, Julia, Alfredo, Xuemei, Vinodh, Paul, Clare, Mark, Tom, Ben, Arial, Dan,

Karamatou, and James. And to my fellow class of 2009, we might have never talked about the wonders of harmonic analysis, but we still had a great time: Bryant, Michelle, Victoria, Arijit, Brianna, and Tyler.

Finally I would like to thank my family, my mother and father, uncle, brother and wife. You supported me through my studies, this is as much for you as it is for me.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Notations

m	meters
Hz	Hertz
nm	nanometers
HS	Hyperspectral
MS	Multispectral
HSI	Hyperspectral Image
MSI	Multispectral Image
IR	Infra-red
UV	Ultraviolet
$E[\cdot]$	Expected value
C_X	The covariance matrix for X
A^T	Transpose of matrix A
$\langle \cdot, \cdot \rangle$	Innerproduct
$\mathbb{Z}$	Integers
$\mathbb{R}$	Real numbers
$\mathbf{1}$	A vector of all 1's
H	Hilbert Space
C^D	Functions with D continuous derivatives
$\mathcal{M}$	Manifold
DR	Dimension Reduction
PCA	Principle Component Analysis
KPCA	Kernel Principle Component Analysis
LE	Laplacian Eigenmaps
SE	Shrödinger Eigenmaps
$G = (V, E)$	A Graph G, Vertex set V, and edge set E
D	Degrees Matrix
$\omega(\cdot, \cdot)$	Weighting function
W	Weights Matrix
L	Graph Laplacian
$\mathcal{L}$	Normalized Graph Laplacian
I_D	The $D \times D$ Identity matrix
$I_{M \times N}$	The $M \times N$ Identity Matrix
$\mathcal{D}$	A Dictionary
$\mathcal{A}$	An Alphabet
F	Frame
$\mathcal{F}$	Frame Analysis Operator
$\mathcal{S}$	Frame Operator
MPGD	Minimum Path Gradient Distance

Chapter 1

Introduction

The problem being studied in this thesis is how to combine output from different sensors into a common product. Such a technique is desirable because, even though two sensors might measure the same area, they might collect very different types of information. Making direct measurements between the data sets would be difficult, if, for example, they each recorded a different number of channels. Thus, we will look to the field of harmonic analysis for ways to express the independently acquired data sources in a common representation. We will mainly use dimension reduction algorithms, particularly Laplacian Eigenmaps (LE), to find this common representation. Dimension reduction algorithms, aside from reducing high dimensional data to a lower dimensional representation without sacrificing pairwise information, are also excellent at extracting dominant features from data sets. These dominant features can then be combined by the data fusion techniques presented herein.

The data sources that we have chosen to use, not only for testing our algorithms but also for designing them, come from the field of remote sensing. Remote sensing is the study of distant objects via the objects' spectral properties. Both passive (e.g., hyperspectral cameras) and active (e.g., LIDAR sensors) technologies can obtain substantial amounts of information about a distant object. This remote

sensing information is a perfect source of data to study because of its abundance of practical uses and varied technologies that produce the data. For example, even in the subdomain of hyperspectral sensors, there are dozens of different types of sensors, each acquiring a unique collection of discrete spectra at different spatial resolutions, view angles, distances from the earth, and times.

1.1 Summary of Results

In the study of data fusion with harmonic analysis we develop several techniques for fusing different types of data, spatial and hyperspectral, LIDAR and hyperspectral, and finally hyperspectral and hyperspectral.

We first provide an approximate preimage algorithm for Laplacian Eigenmaps, where in the past only results for Kernel PCA (KPCA) existed. We accomplish this by building upon the results for KPCA and using the Nyström out of sample extension and inequality constrained quadratic programming to find the approximate preimage.

Next, we develop a $\Sigma\Delta$ Quantization landmark algorithm for the Nyström extension as a means to accelerate the computation of the lower dimension embeddings of our data sets. This allows us to see the Nyström extension, and the choice of landmark points, in a different mathematical context. We also formulate a means to move from an LE analysis of a data set to a Schrödinger Eigenmap (SE) analysis of the same data using efficient matrix factorization updates.

For the problems of spatial-spectral fusion with hyperspectral sensors, we use

the Laplacian Eigenmaps algorithm as a template to address spatial-spectral fusion at the graph, operator, and feature space level. We accomplish this first set of data fusions by modifying graph distance calculations, modifying the construction of k-nearest neighborhoods, and developing fusion operators to combine separately-calculated graph Laplacians.

For the problem of hyperspectral-LIDAR fusion, we develop an algorithm, the Minimum Path Gradient Distance, to encode LIDAR information into a nontrivial graph representation. We then use this graph representation of the LIDAR data and fuse it with the hyperspectral data to produce better classification and data representation results. We also adapt the feature space rotation methodology of Diffusion Maps to Laplacian Eigenmaps and rotate the feature spaces for the hyperspectral and LIDAR together as a means of fusion.

Next, we study the problem of fusing two hyperspectral data sets, first with varying levels of overlapping pixels, then finally without any overlapping pixels. For the former, we use rotation maps learned on partial overlapping image segments. For the latter, we use a graph matching algorithm to learn pixel-to-pixel relations. Once these relations have been learned, a similar feature space rotation is formulated. Now having the two data sets combined in the same feature space, a classifier can be used to obtain classification results from the fused product.

Finally, we propose a technique for fusing expert knowledge into the data representation of a hyperspectral data set via the Schrödinger Eigenmaps algorithm.

1.2 Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 provides background on the field of remote sensing and examples of the types of data collected. These data will be analyzed later in the thesis. Chapter 3 discusses the mathematical background of dimension reduction of Laplacian and Schrödinger Eigenmaps. Chapter 4 shows how an approximate preimage can be calculated for the Laplacian Eigenmaps algorithm. Chapter 5 provides a summary of Random Projections and Approximate Neighborhood construction that we will use in our fusion methods. Chapter 6 provides the details of the developed fusion methodologies and results from example data sets. Chapter 7 proposes the application of Nyström method to accelerate our developed methods and how Frames and $\Sigma\Delta$ Quantization can be used with remote sensing data and the Nyström method.

Chapter 2

Introduction to Remote Sensing

It is often impossible to observe a distant object or area physically due to the financial costs, safety concerns and/or length of travel incurred in traveling to the object or area. In addition, if the objectives of a mission require concealment it would be impractical to survey an area. Remote Sensing is the broad field of study that attempts to solve these problems. It is a field with many definitions, see [29]; two of the most citied are:

> Remote sensing is the acquisition of physical data of an object without touch or contact [96].

> Remote sensing is the science of deriving information about an object from measurements made at a distance from the object, i.e., without actually coming in contact with it. The quantity most frequently measured in present day remote sensing systems is the electromagnetic energy emanating from objects of interest, and although there are other possibilities (e.g., seismic waves, conic waves, and gravitational force), our attention ... is focused upon systems which measure electromagnetic energy[124].

In this thesis we will restrict the definition of the remote sensing to sensing that involves only the earth's surface, but it should be noted that remote sensing can involves studying other planets, stars, galaxies, etc.

Remote sensing can be divided into two separate domains based upon how the sensors acquire their data, i.e., actively or passively. Active sensors interact with the target or area of interest by sending a signal and waiting for a response. Examples of active sensors include Radio Detection and Ranging (RADAR), Sound Navigation and Ranging (SONAR), Light Detection And Ranging (LIDAR) and Synthetic Aperture Radar (SAR). In Section 2.2 we will go into greater detail on LIDAR sensors. For more information about RADAR and SAR see [63] and [47]. Passive sensors do not interact with the target or area of interest directly, but instead acquire the radiated energy from objects, usually in the form of reflected solar radiation. Passive sensors detect radiation in a wide range of frequencies from extra-low to gamma rays which represent frequencies from 10^3Hz to 10^{20}Hz or wavelengths in the range 10^5m to 10^{-12}m. In the realm of passive sensors, multispectral and hyperspectral sensors are the most ubiquitous. Multispectral sensors combine tens of different distinct frequencies of light, while hyperspectral combine hundreds of different frequencies of light, usually ranging from the Ultraviolet (UV) to the Infra-red (IR) wavelengths, see Figure 2.1. We will discuss more broadly Hyperspectral sensors in Section 2.1. Other types of passive sensors include those that focus on microwave [95] and thermal IR [106].

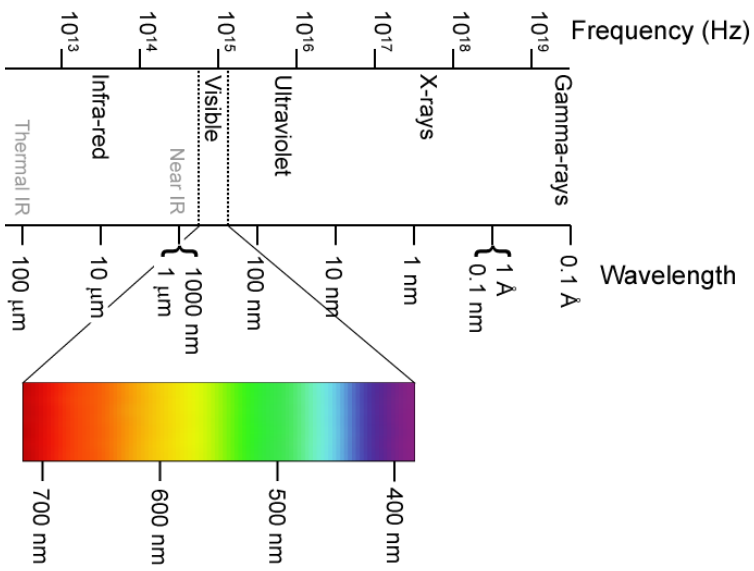


Figure 2.1: *Electromagnetic Spectrum showing the ultraviolet, visible, near-infrared, and shortwave infrared which are used in hyperspectral imaging.*

Both types of sensors have their benefits. Active sensors are favored over passive sensors due to their ability to collect information without the presence of a radiating source (i.e., the sun) and their increased range of operation. Passive sensors are favored due to their concealability (do not require a signal to be sent to the area of interest), low power usage, high angular resolution, and spectral variability [29]. A fusion of data derived from both types of sensors is thus favorable as it allows for each of the strengths to be used. Data fusion from active and passive sensors will serve as the inspiration for this thesis in the following chapters.

2.1 Hyperspectral Sensors and Imaging

A hyperspectral sensor collects information from a distant object by sensing the solar radiation that is reflected off of the object in a discrete collection of wavelengths, as seen in Figure 2.2. These sensors can be installed in planes, contained in satellites, placed on high structures, or carried by hand. The sensors record information about the scene in one of two ways: either by a push-broom or by a staring array. Push-broom sensors record one spatial line at a time for all available wavelengths. Staring arrays record all spatial lines at once but for only one light frequency. Radiation emitted from the sun, upon striking an object on the earth's surface, has its spectrum modified because at various wavelengths the radiation is absorbed to some degree by the object. This physical reaction modifies the radiation spectrum produced by the sun in a unique way for each object. The spectral signature can be used to accurately identify objects either by classifying them based on known ground truths (pixels with a descriptive label) or comparing the signals to a known spectral library of signatures. This process is complicated by the fact that the solar radiation is also modified by the atmosphere both preceding and succeeding the impact with the object.

The collected data from the hyperspectral sensor can be integrated into the form of an image which is called a hyperspectral image (HSI) or data cube. Usually some form of image processing is done on the collected data before it is ready for analysis. For example, it is necessary to smooth and georectify the data to remove effects from the uneven travel of the plane and any shift or rotations between

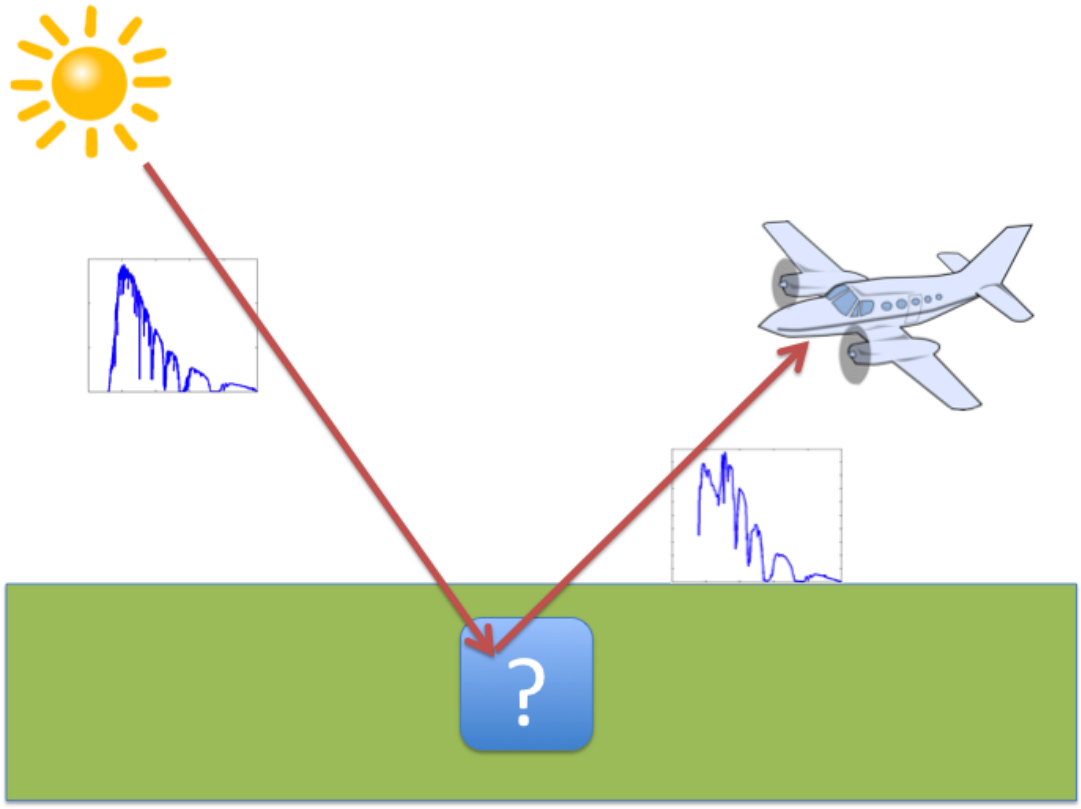


Figure 2.2: *The path of solar radiation from the sun to the hyperspectral sensor (in this case on a plane). Note how the solar spectrum is by the target on the ground.*

flightlines. It is also necessary to perform some level of in-painting, i.e. using surrounding pixels to infer information about a missing pixel, and interpolation to account for missing data in either a spectral or spatial sense, and to fit the data to a uniform pixel grid.

Depending on the application, it is useful to distinguish reflectance from radiance. As the solar radiation enters the atmosphere, it is altered by the presence of water molecules and other particulate matter in the atmosphere. The same effect happens once the solar radiation is reflected off the ground or object, as shown

in Figure 2.2. The data that are recorded by the sensor are known as the radiance spectrum. The reflectance spectrum for a particular band is the ratio of the reflected radiation at that band to the incident radiation at that band, and can be recovered from the collected radiation spectrum by using atmospheric correction equations, for example Quick Atmospheric Correction (QUAC) [23] or Fast Line-of-sight Atmospheric Analysis of Hypercubes (FLAASH) [101]. The application of atmospheric correction codes is necessary as it often useful to compare spectral signatures in an image to a library of lab-recorded spectra, i.e., spectra that have not been altered by the atmosphere.

HSIs have a wide range of practical applications, ranging from agriculture to mineralogy and exploration to applications in the security and defense fields. Farmers are able to use hyperspectral images to determine plant stress levels, amount of water being absorbed, and possible insect infestations [129]. In resource exploration, in particular in mineralogy exploration, hyperspectral images can be quickly obtained for a vast amount of territory, and then known spectral signatures corresponding to desirable minerals can be searched for, see, e.g., [134]. The defense industry makes wide use of hyperspectral images for target detection and tracking [99] because, for example, the limitations of human vision make it difficult to discern modern camouflage from vegetation [116].

HSI, in general, has hundreds of spectral bands in contrast to a normal digital image which has three spectral bands (blue, red, and green), and thus offers a more complete part of the light spectrum for viewing and analysis [99]. As a note, a multispectral sensor is a sensor that samples the light spectrum at fewer wavelengths than

a hyperspectral sensor, usually only between 7 and 20. This fine sampling performed by a hyperspectral sensor of the electromagnetic spectrum offers powerful material discernability but the large dimension number and inherent correlation amongst the bands, as seen in Figure 2.3, leads to difficultly in analysis. A collection of these spectral bands for the Indian Pines image, which will be discussed in Section 2.1.3, is shown in Figure 2.4. In Figure 2.5, we show 3 potential false color representations of the hyperspectral scene, i.e., choosing three bands to represent the red, blue and green bands in the RGB color space.

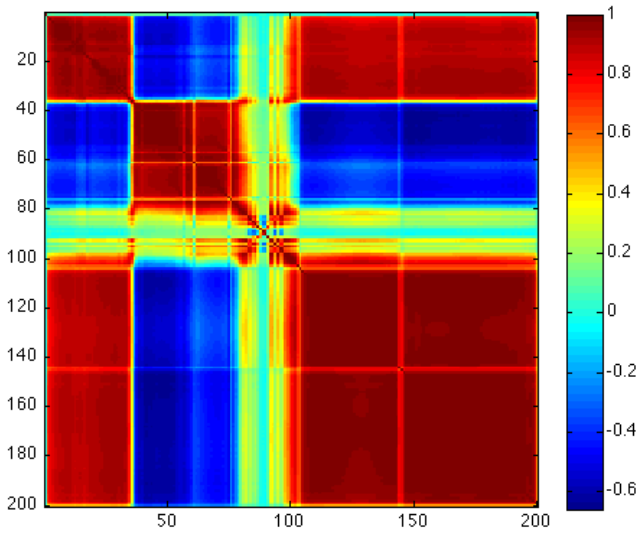


Figure 2.3: *Correlation matrix for the 200 spectral bands contained in the Indian Pines Hyperspectral scene. Red is positively correlated while blue is negatively correlated.*

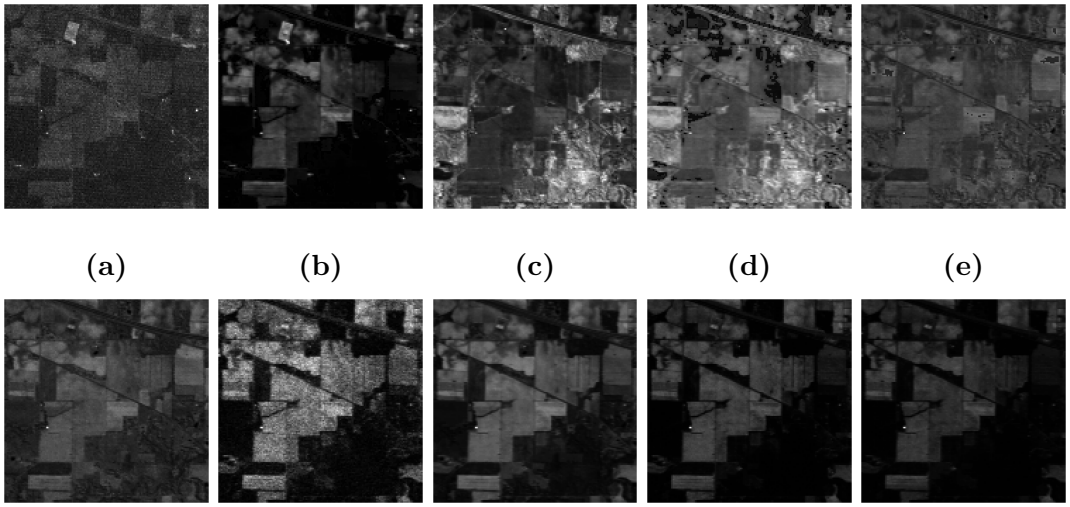


Figure 2.4: *A collection of the bands from the Indian Pines Hyperspectral Image rendered in gray scale.*

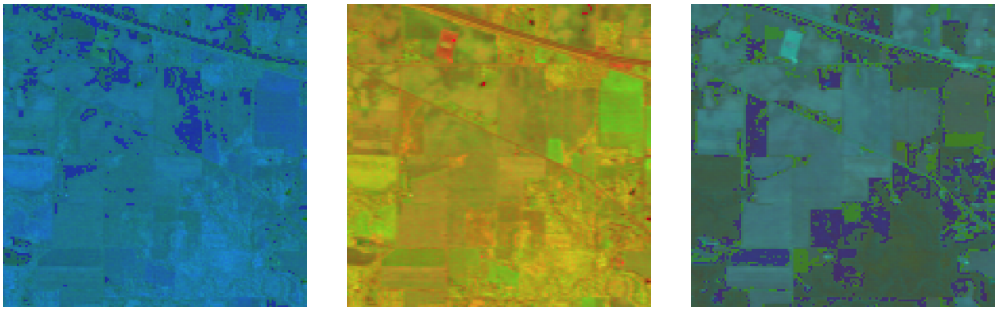


Figure 2.5: *Three potential false color renderings (assigning 1 band to red, 1 band to green, and 1 band to blue) of the Indian Pines Hyperspectral image.*

From a mathematical perspective, a hyperspectral image is a $m \times n \times D$ matrix, X, or a collection of $mn = N$, D-dimensional spectral vectors, each representing the information for one pixel [119]. As D increases, this high dimensionality of data, as well as its inherent redundancy, leads to the field of machine learning

and specifically, dimension reduction and data representation. These fields will be discussed in Chapter 3. One of the areas of research into HSI is image classification. The major goal of image classification is to classify the pixels of an image into some number of classes with the use of training data. This allows for example the creation of vegetation maps near wetlands [3]. We will define ground truth or training data for the dataset X as the set of ordered triplets, $T = \{(r_i, c_i, z_i)\}_{i=1}^{t}$, where c_i and r_i represent the row and column for the ith training pixel with class label z_i. The set T can be used to classify an image by training a classifier, which once trained, will return the most probable class label for any pixel in image. For this thesis, we will use the simple nearest neighbor (1NN) classifier. The 1NN classifier labels an unknown pixel x as z_r where:

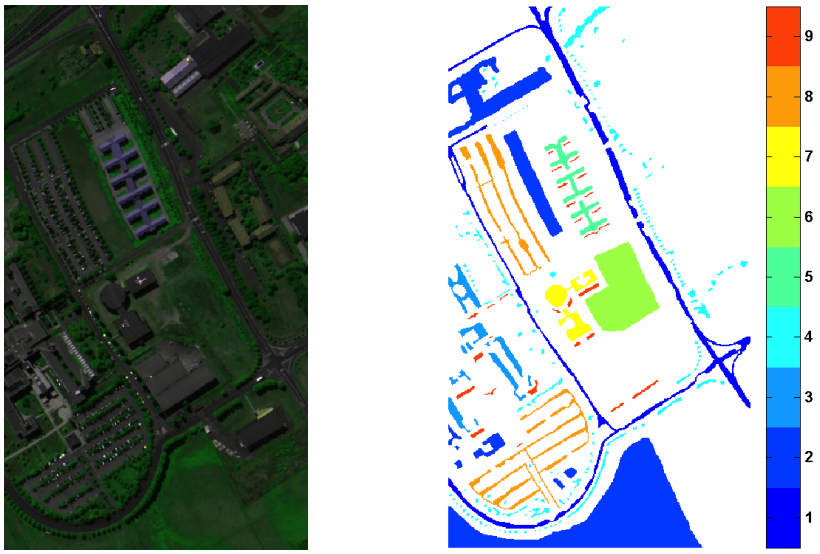
$$r = \arg\min_i \|X(r_i, c_i) - x\|_2.$$

Since we will be using hyperspectral images as an application of our data fusion methods in the later chapters, in Sections 2.1.1, 2.1.2, 2.1.3, and 2.1.4 we will introduce the popular Pavia University, Pavia Centre, Indian Pines, and Urban data sets, respectively.

2.1.1 Pavia University

The Pavia University scene is a 610×340 pixel image [54] that contains 103 spectral bands with approximately 1.3 meter resolution. It was acquired in a flyover of Pavia, Italy using a ROSIS sensor. There is a collection of $42,776$ ground truth pixels covering a total of 9 classes. Figure 2.6 contains a 3 color composite image

of Pavia University, as well as a ground truth mask and Table 2.1 contains the corresponding class names and number of samples.



(a) **(b)**

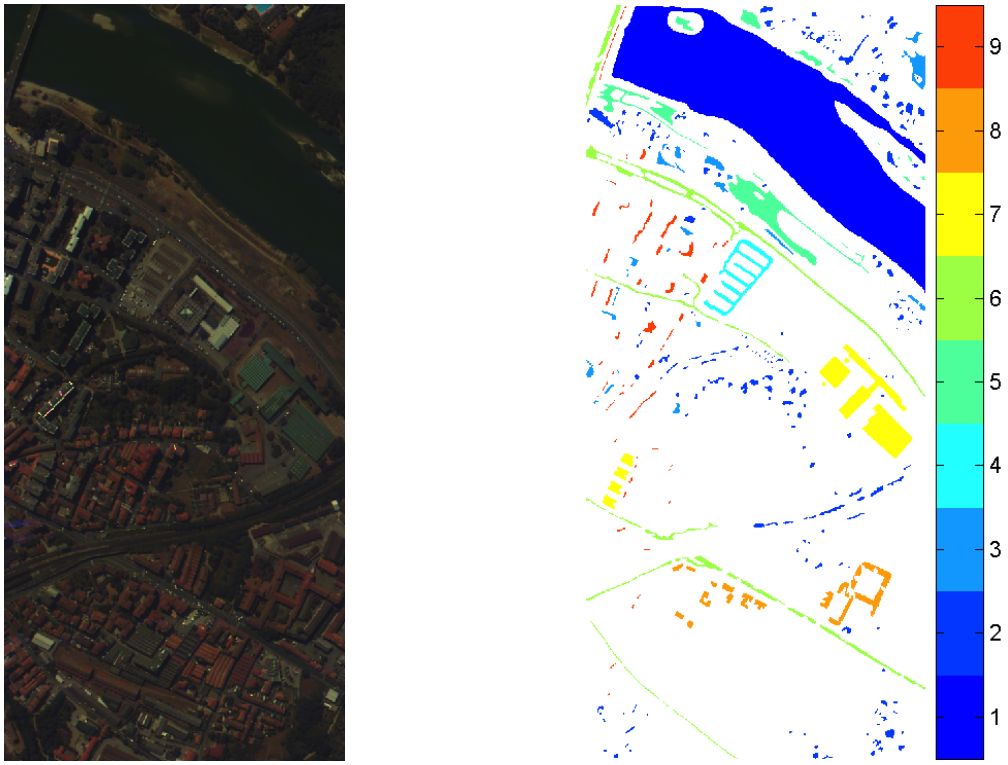
Figure 2.6: *(a) Three color composite image and (b) ground truth mask for Pavia University.*

#	Class Name	Samples	#	Class Name	Samples
1	Asphalt	6631	6	Bare Soil	5029
2	Meadows	18649	7	Bitumen	1330
3	Gravel	2099	8	Self-Blocking Bricks	3682
4	Trees	3064	9	Shadows	947
5	Painted metal sheets	1345			

Table 2.1: *Ground truth descriptions with sample count for Pavia University.*

2.1.2 Pavia Centre

The Pavia Centre scene is a 1096×492 pixel image [54] (we have chosen to crop out the right side of the image where little ground truth is available) that contains 102 spectral bands with approximately 1.3 meter resolution. It was acquired in a flyover of Pavia, Italy using a ROSIS sensor. There is a collection of $103,539$ ground truth pixels covering a total of 9 classes. Figure 2.7 contains a 3 color composite image of Pavia Centre, as well as a ground truth mask and Table 2.2 contains the corresponding class names and number of samples.



(a) **(b)**

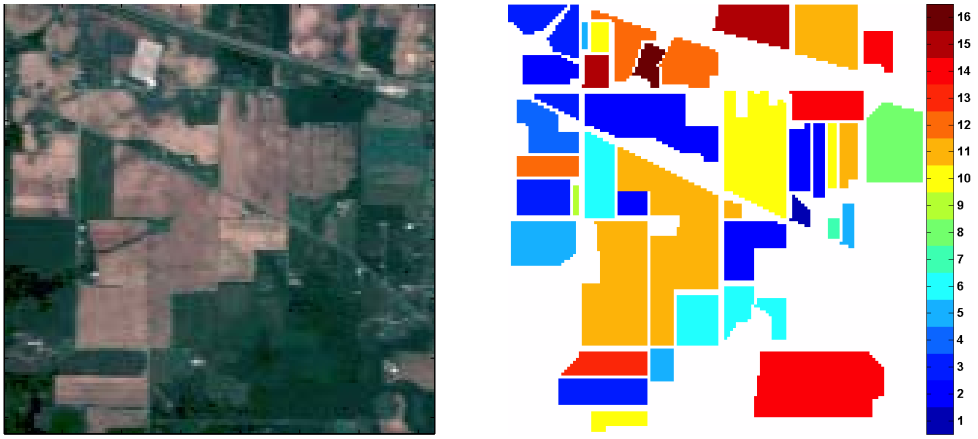
Figure 2.7: *(a) Three color composite image and (b) ground truth mask for Pavia Centre.*

#	Class Name	Samples	#	Class Name	Samples
1	Water	65278	6	Tiles	7585
2	Trees	6508	7	Shadows	7287
3	Asphalt	2905	8	Meadows	3122
4	Self-Blocking Bricks	2140	9	Bare Soil	2165
5	Bitumen	6549			

Table 2.2: *Ground truth descriptions with sample count for Pavia Centre.*

2.1.3 Indian Pines

The Indian Pines scene [88] is a 145×145 pixel image that contains 224 spectral bands with approximately 20 meter resolution; we discard 4 of the bands due to sensor noise and absorption by water. It was acquired in a flyover of northwest Indiana using an AVIRIS spectrometer. There is a collection of $10,249$ ground truth pixels associated with the data set coving a total of 16 classes. Figure 2.8 contains a 3 color composite image of Indian Pines, as well as a ground truth mask and Table 2.3 contains the corresponding class names and number of samples.



(a) **(b)**

Figure 2.8: *(a) Three color composite image and (b) ground truth mask for Indian Pines.*

#	Class Name	Samples	#	Class Name	Samples
1	Alfalfa	46	9	Oats	20
2	Corn-notill	1428	10	Soybean-notill	972
3	Corn-mintill	830	11	Soybean-mintill	2455
4	Corn	237	12	Soybean-clean	593
5	Grass-pasture	483	13	Wheat	205
6	Grass-trees	730	14	Woods	1265
7	Grass-pasture-mowed	28	15	Bldgs-Grass-Trees-Drives	386
8	Hay-windrowed	478	16	Stone-Steel-Towers	93

Table 2.3: *Ground truth descriptions with sample count for Indian Pines.*

2.1.4 Urban

The Urban scene is a 307×307 pixel image [133] that contains 210 spectral bands with approximately 3 meter resolution; we discard 48 of the bands due to sensor noise and absorption by water leaving 162 bands for analysis. There is collection of $1,058$ ground truth pixels representing 23 classes. It was acquired in a flyover of Copperas Cove, Texas using a HYDICE spectrometer. Figure 2.9 contain a 3 color composite image of Urban.

Figure 2.9: *Three color composite image of Urban.*

2.2 LIDAR

LIDAR is a sensing technology that uses light in the UV, visible, and/or IR spectrums to image the earth's surface. Based on the applications and the spatial fidelity required, a wavelength is chosen as well as a type of scattering (e.g. Lie, Mie, or Rayleigh). LIDAR systems can be used to figure out ground reflectivity, but for this thesis we will focus on its use in Digital Terrain Mapping (DTM) or Digital

Elevation Mapping (DEM). In DTM and DEM, LIDAR data is created by measuring the time it takes for the laser beam to return to its origin. The emitted laser pulse may return in a series of backscatters as it reflects off of objects of different heights in its path to the ground, see Figure 2.10. The first return gives the top of the feature, while intermediate returns can be used to infer structural information about the object, and the last return usually signifies the laser pulse impact with the ground; however it is possible that the entire pulse is reflected before hitting the ground. Then, the scene's height information can be inferred relative to a known ground level quantity. Images derived from LIDAR sensing generally offer a single channel which contains the calculated height for a small region of the scene; in addition, sometimes the first and last detected backscatter and their converted heights are included as well.

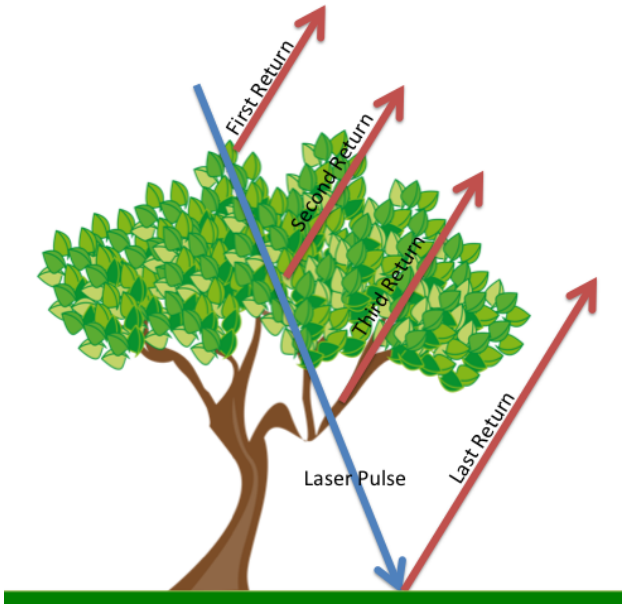
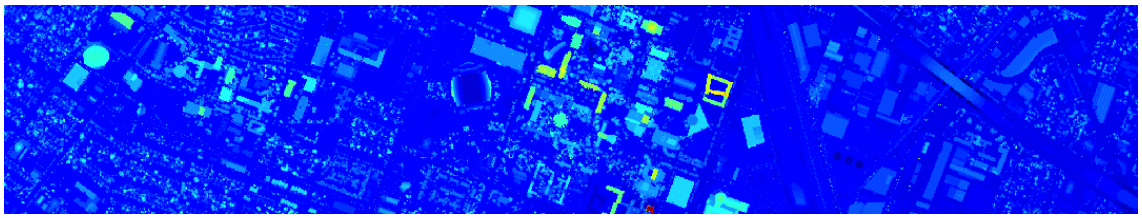


Figure 2.10: *The LIDAR laser pulse after being emitted is backscattered several times during its path to the ground. This backscattered pulses can be used to gain information about the objects in the path of the laser including height of the objects.*

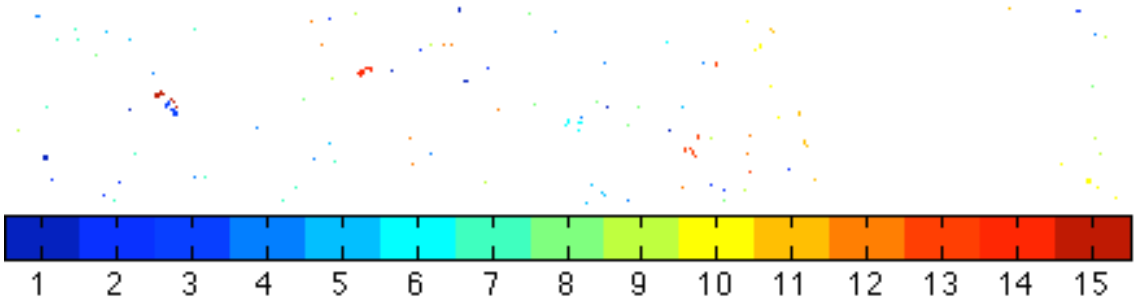
2.2.1 University of Houston

The HSI and the LIDAR provided by the 2013 IEEE GRSS Data Fusion Contest [51], as seen in Figure 2.11, are co-registered 349×1905 images. The HSI was acquired using a CASI sensor which has 144 spectral bands between 380nm and 1050nm. The image was acquired over the University of Houston on June 22, 2012 at 1400 UTC. The LIDAR image was acquired by the National Center for Airborne Mapping on June 23, 2012 at 1700 UTC. The data set contains $2,832$ labeled pixels over 15 classes as seen in Figure 2.4.

(a)



(b)



(c)

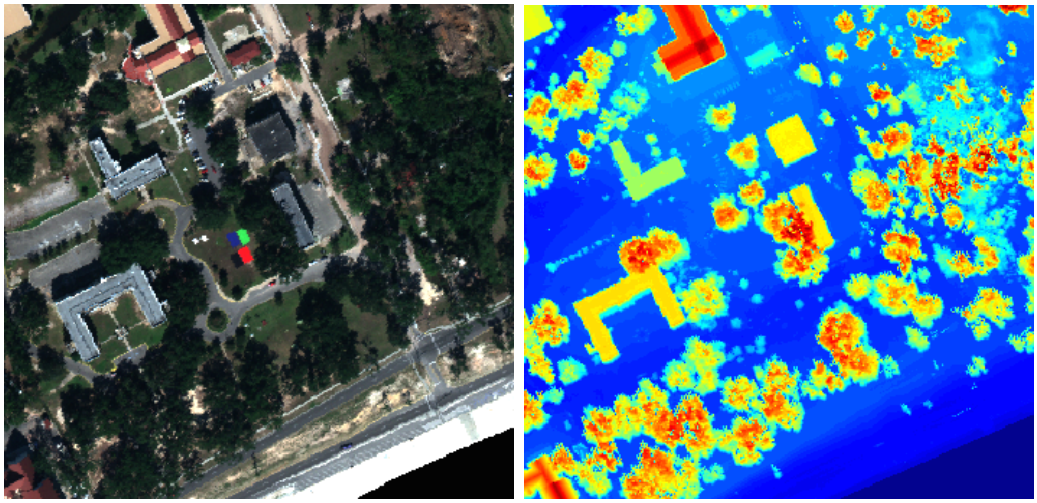
Figure 2.11: *(a) Three color composite image from HS data, (b) LIDAR height map (red = tallest, blue = shortest) and (c) ground truth mask for University of Houston*

#	Class	Samples	#	Class	Samples
1	Healthy grass	198	9	Road	193
2	Stressed grass	190	10	Highway	191
3	Synthetic grass	192	11	Railway	181
4	Trees	188	12	Parking Lot 1	192
5	Soil	186	13	Parking Lot 2	184
6	Water	182	14	Tennis Court	181
7	Residential	196	15	Running Track	187
8	Commercial	191			

Table 2.4: *Ground truth descriptions with sample count for University of Houston*

2.2.2 MUUFL Gulfport

The MUUFL Gulfport [70] dataset contains a co-registered HSI and LIDAR data of size 325×337. The HSI was acquired using a CASI-1500 sensor which has 72 spectral bands between 375nm and 1050nm. The image was acquired over the University of Southern Mississippi-Gulfport, located in Long Beach, Mississippi on November 10, 2010 at approximately 1200 CT. The LIDAR image was acquired using a Gemini sensor. The two data sets can be seen in Figure 2.12.



(a) **(b)**

Figure 2.12: *(a) Three color composite image from HS data and (b) LIDAR height map (red = tallest, blue = shortest) for MUUFL Gulfport scene.*

Chapter 3

Introduction to Dimension Reduction

In recent years, large advancements in technology have allowed for sensors to record at increasingly finer spectral, spatial, and temporal scales. This advancement in technology, as well as the large drop in storage costs, has lead to an abundance of data. This data is only useful if it can be analyzed in an efficient and time sensitive manner. It is no longer possible for analysts to analyze this data unassisted; dimension reduction techniques offer a partial solution to this problem by reducing the number of spectral dimensions present in a data set.

Dimension reduction, is a type of machine learning or data representation method, which seeks to transform high dimensional data to a lower dimensional representation while at the same time preserving the intrinsic qualities of the data. High dimensionality is problematic because of the *curse of dimensionality* [12] or the empty space phenomenon [120]. The *curse of dimensionality* describes situations where the complexity of a problem grows exponentially with the number of dimensions. The empty space phenomenon illustrates that when data is described by a few observations then the high dimensional space becomes sparse.

There are two main branches of dimensionality reduction algorithms: linear and nonlinear. Linear methods, which include Principal Component Analysis (PCA) [110], Independent Component Analysis [43] and Multi-Dimensional Scaling (MDS)

[132], have been used past to study problems in image processing, signal processing, andstatistics. There are also a growing number of nonlinear dimension reduction techniques, such as Laplacian Eigenmaps (LE) [9], Schrödinger Eigenmaps (SE) [48], Diffusion Maps (DM) [42], Kernel PCA (KPCA) [118], Local Linear Embedding (LLE) [114], Local Tangent Space Alignment (LTSA) [140], and Hessian LLE (HLLE) [58]. All of these nonlinear techniques can be described as Output Normalized Algorithms [73]. These nonlinear methods have been used to study the same problems as the linear methods. However, while linear methods are superior in their computational efficiency, they are limited by their assumption that the original data lies on a linear manifold. This assumption can be shown to be untrue for many data sources, for example, hyperspectral data is nonlinear, see, e.g., [92, 36, 3]. For this reason, and nonlinear dimension reduction algorithms' well-documented use in remote sensing data [80, 41, 65, 75, 97], we will rely on nonlinear methods for our fusion methodologies.

To illustrate the advantages of nonlinear methods over linear methods we will look at the toy data set known as the Helix show in Figure 3.1 (a). The Helix data set, which is a torodial helix in 3 dimensions, should be unraveled to form a circle in 2 dimensions as it is fundamentally a 2-dimensional object. We see that PCA, a linear method, and KPCA, a method which is somewhere between a linear and nonlinear method, both have trouble with the data. In Figure 3.1 (b) and (c), both of these methods are unable to unravel the helix; instead they project the third dimension onto the hyperplane which runs through the origin. This results in distances between points, especially near the cusps of the asterisk, becoming skewed.

The LE algorithm, however, is able to properly recover the 2-dimensional circle, as seen in in Figure 3.1 (d). While this is only a toy example it illustrates the power of nonlinear dimension reduction techniques to find lower dimensional representations of complicated data sets.

For the following sections in this Chapter, let the data set $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ where $x_i(p)$ is the pth coordinate of the vector x_i. We will seek an embedding $Y = \{y_1, \ldots, y_N\} \subset \mathbb{R}^d$ where $d \leq D$. y_i will be referred to as the lower dimensional representation of the data point x_i. We will call Y, the feature space of X and the d vectors composing Y the feature vectors. We can also consider a function $\Phi : \mathbb{R}^D \mapsto \mathbb{R}^d$, a dimension reduction function.

For the data fusion methods we are developing, we will make use of Principle Component Analysis, Laplacian, and Schrödinger Eigenmaps. Some detail of these algorithms and their mathematical background in Sections 3.1 - 3.4. In Section 3.2, we provide a brief discussion of spectral graph theory as this is vital for the exposition of LE and SE.

3.1 Principle Component Analysis (PCA)

PCA is a linear method that uses an orthogonal transform to convert a set of data into its so called principle components. The principle components are chosen to maximize the preserved variance in the original data or to minimize the reconstruction error. It can be shown that both definitions are analogous, see [93]. PCA is also sometimes referred to as Karhunen-Loève Transform (KLT) in signal pro-

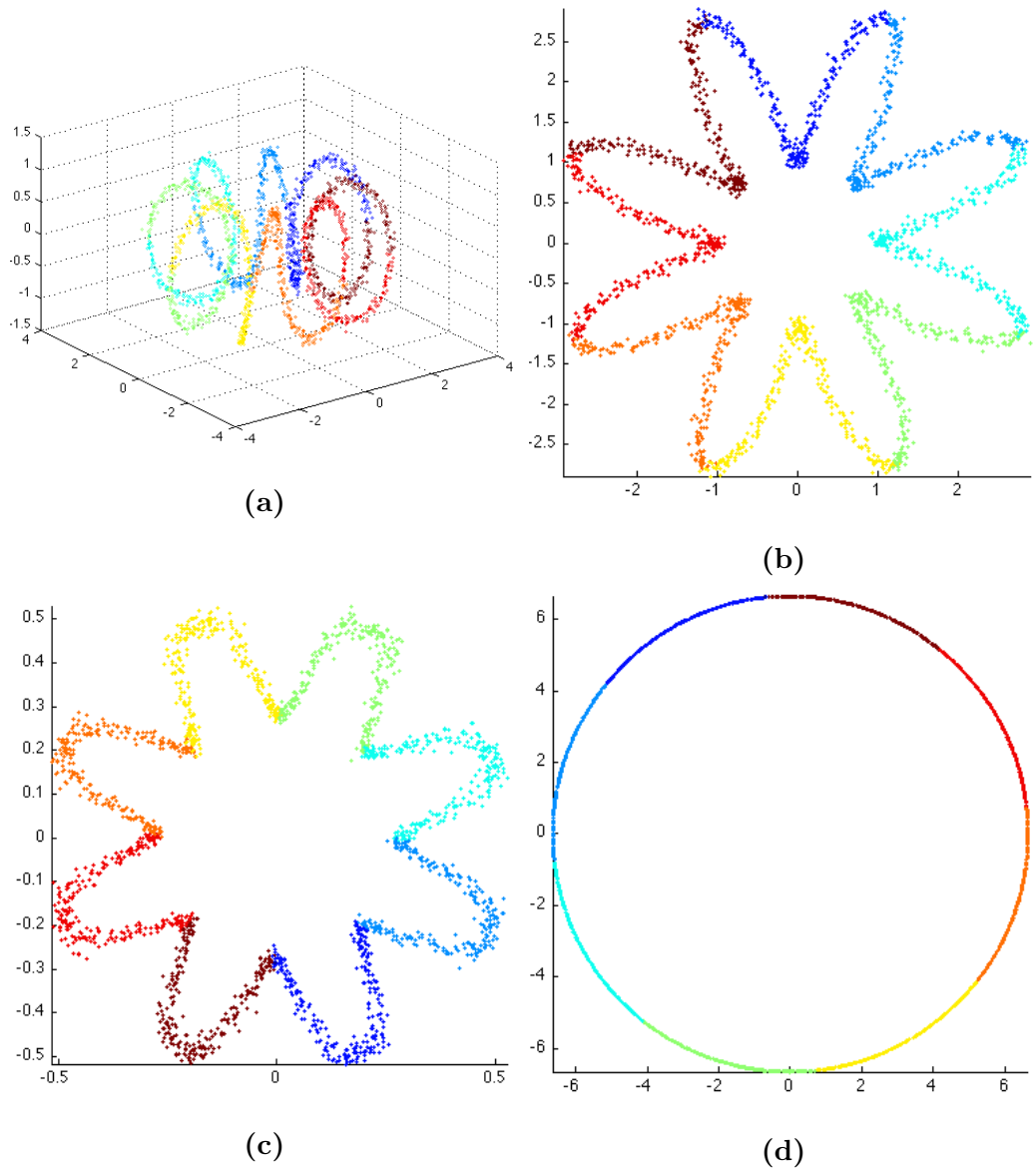


Figure 3.1: *(a) Helix Data Set, 1st and 2nd principle embedding dimension for (b) PCA, (c) KPCA and (d) LE.*

cessing and Proper Orthogonal Decomposition (POD) in engineering. It can also be shown to be of the same form as Independent Component Analysis (ICA) and Multi-dimensional Scaling (MDS). The dimension that PCA projects into, unlike the nonlinear methods which will be discussed in the following sections, is bounded above by the initial dimension of the data. It also differs because, it has a natural inversion or in other words, an exact preimage.

PCA assumes that the D observations, for $x \in X$, can be realized as the linear transformation, W, of d latent variables from a Gaussian distribution, $y \in Y \subset \mathbb{R}^d$. This can be compactly stated as:

$$x = Wy. \tag{3.1}$$

The PCA algorithm has three steps which are summarized below:

1. Mean center the data. This can be accomplished in linear algebra terms by,

   $$X = X - \frac{1}{N} X \mathbf{1}\mathbf{1}^T,$$

   where $\mathbf{1}$ is a $N \times 1$ matrix of 1's.

2. Minimize the reconstruction error or maximize the preserved variance. Since these operations are equivalent we will only detail the latter.

   Let the covariance matrices for X and Y be defined as:

   $$C_Y = E\left\{YY^T\right\} \tag{3.2}$$

   and

   $$C_X = E\left\{XX^T\right\}. \tag{3.3}$$

Now substituting (3.1) and (3.2) into (3.3), we obtain:

$$\begin{aligned} C_X &= E\left\{WYY^TW^T\right\} \\ &= WC_YW^T. \end{aligned} \tag{3.4}$$

We can now rearrange (3.4), by left and right multiplication by W and W^T to produce,

$$C_Y = W^TC_XW. \tag{3.5}$$

If we diagonalize C_X, as seen in (3.5), then

$$C_X = V\Sigma V^T$$

where V is the $D \times D$ matrix of left singular vectors and Σ is the $D \times D$ diagonal matrix of singular values. Thus,

$$C_Y = W^TV\Sigma V^TW.$$

Now we let:

$$W = VI_{D\times d},$$

where $I_{d\times D}$ is the $d \times D$ identity matrix.

Thus we have shown that:

$$C_Y = I_{d\times D}\Lambda I_{D\times d}.$$

3. Calculate the lower dimensional embedding Y by preserving only the first d principle components:

$$Y = I_{d\times D}V^TX,$$

3.2 Spectral Graph and Operator Theory

To give a proper treatment of the LE and SE algorithms and then develop fusion methodology with them, it is vital to understand the spectral graph theory that underlies their algorithms. For this section we will reference the basic results from [38] and [24], which offer an excellent introduction to spectral graph theory.

3.2.1 Unweighted Graphs

We will first need to start with the basic definition of a graph:

Definition 3.2.1 (Unweighted Directed Graph, Vertex, Edge)**.** The unweighted directed graph $G = (V, E)$ is defined by a set of vertices (or nodes) $V = \{v_1, v_2, \ldots, v_N\}$ and a set of ordered pairs of vertices, $E = \{(u, v), u, v \in V\}$. The ordered pair of vertices $e = (u, v)$ is known as an edge.

If we drop the notion of orderedness to the edge pairs we get an undirected graph:

Definition 3.2.2 (Undirected Unweighed Graph)**.** An undirected, unweighted graph $G = (V, E)$ has an edge set E such that if $(u, v) \in E$, $(v, u) \in E$, as well.

We will now refer to an undirected graph simply as a graph and when we need to refer to a directed graph specifically, we will. To eventually build our definition of the graph Laplacian, we will need to define the degree of a vertex and the Adjacency matrix associated with a graph:

Definition 3.2.3 (Adjacency Matrix)**.** The adjacency matrix for a graph $G =$

(V, E), $N = |V|$, is the $N \times N$ matrix matrix A such that

$$A(i,j) = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Definition 3.2.4 (Degree)**.** For a graph $G = (V, E)$, $N = |V|$, the degree for a vertex $v \in V$ is $d_v = |\{(v, u) : u \in V, (v, u) \in E\}|$. It is common to store all the degrees for a graph G in a $N \times N$ diagonal matrix D such that $D(i, i) = d_i$.

We can now define the graph Laplacian as:

Definition 3.2.5 (Graph Laplacian)**.** The graph Laplacian, L, of the graph $G = (V, E)$, $|V| = N$, is defined as $L = D - A$, or for $u, v \in V$,

$$L(u,v) = \begin{cases} d_u, & \text{if } u = v \\ -1, & \text{if } (u, v) \in E \\ 0, & \text{otherwise.} \end{cases}$$

It is sometimes convenient to look at the normalized graph Laplacian, $\mathcal{L}$,

Definition 3.2.6. The normalized graph Laplacian, $\mathcal{L}$, $G = (V, E)$, $|V| = N$, is defined as $\mathcal{L} = D^{-1/2}AD^{-1/2}$, or for $u, v \in V$,

$$\mathcal{L}(u,v) = \begin{cases} 1, & \text{if } u = v \text{ and } d_u \neq 0 \\ -\frac{1}{\sqrt{d_u d_v}}, & \text{if } (u, v) \in E \\ 0, & \text{otherwise.} \end{cases}$$

$\mathcal{L}$, can also be represented as:

$$\begin{aligned}\mathcal{L} &= D^{-1/2}LD^{-1/2} \\ &= I - D^{-1/2}AD^{-1/2} \\ &= SS^T,\end{aligned}$$

where S is a $N \times |E|$ matrix where each column corresponds to an edge, i.e. for $e_k = (v_i, v_j)$, then in column k, row i has an entry of $1/\sqrt{d_i}$, row j has an entry of $-1/\sqrt{d_j}$, and all other entries are 0.

By the definition, $\mathcal{L}$ (and L if A is symmetric) is a symmetric matrix of real entries and thus, a Hermitian matrix.

Let g be a function that maps the vertices of G to real numbers, i.e., $g : V \to \mathbb{R}$ and let f be the function defined as $g = D^{1/2}f$. Now using the above definitions of $\mathcal{L}$ we can calculate:

$$\begin{aligned}\frac{\langle g, \mathcal{L}g\rangle}{\langle g, g\rangle} &= \frac{\langle g, D^{-1/2}LD^{-1/2}g\rangle}{\langle g, g\rangle} \\ &= \frac{\langle f, Lf\rangle}{\langle D^{1/2}f, D^{1/2}f\rangle} \\ &= \frac{\sum_{u\sim v}(f(u) - f(v))^2}{\sum_v f(v)^2 d_v},\end{aligned} \tag{3.6}$$

where $\sum_{u\sim v}$ is the sum over all unordered vertex pairs $(u, v) \in E$. The calculation in (3.6) gives us the definition of Dirichlet Sum and Rayleigh Quotient:

Definition 3.2.7 (Dirichlet Sum)**.** The Dirichlet Sum of G is defined as:

$$\sum_{u\sim v}(f(u) - f(v))^2.$$

Definition 3.2.8 (Rayleigh Quotient)**.** The Rayleigh Quotient is defined as $\frac{\langle g,\mathcal{L}g\rangle}{\langle g,g\rangle}$.

By Equation (3.6) and the fact that $\mathcal{L}$ is Hermitian, we can see that all the eigenvalues are real and nonnegative. We can also see that $D^{1/2}\mathbf{1}$ is a an eigenfunction of $\mathcal{L}$ with eigenvalue 0:

Since the eigenfunctions of $\mathcal{L}$ must be orthogonal to each other, we can conclude that the second eigenvalue, λ_1, of $\mathcal{L}$ has corresponding eigenfunction $g = D^{1/2}f$ where

$$\lambda_1 = \inf_{f \perp D\mathbf{1}} \frac{\sum_{u \sim v}(f(u) - f(v))^2}{\sum_v f(v)^2 d_v}. \tag{3.7}$$

Just as we expressed the eigenvalue λ_1 in terms of the Rayleigh quotient in (3.7) we can express λ_k as:

$$\begin{aligned} \lambda_k &= \inf_f \sup_{g \in P_{k-1}} \frac{\sum_{u \sim v}(f(u) - f(v))^2}{\sum_v (f(v) - g(v))^2 d_v} \\ &= \inf_{f \perp DP_{k-1}} \frac{\sum_{u \sim v}(f(u) - f(v))^2}{\sum_v f(v)^2 d_v}, \end{aligned} \tag{3.8}$$

where P_{k-1} is the subspace generated by the eigenfunctions corresponding to $\lambda_0, \lambda_1, \ldots, \lambda_{k-1}$.

Using (3.8) and the trace of $\mathcal{L}$ it is possible to prove some basic facts about the spectrum of G:

Lemma 3.2.9 (Eigenvalues of Unweighted Laplacians)**.** *For a graph $G = G(V, E)$, $|V| = N$,*

1. $\sum_{i=0}^{N-1} \lambda_i \leq N$
2. *For $N \geq 2$, $\lambda_1 \leq \frac{N}{N-1}$*

3. *For $N \geq 2$ and no isolated vertices, $\lambda_{N-1} \leq 2$*

4. *The spectrum of G is the union of the spectrum of the connected components of G. A connected component is the largest collection of vertices in G such that there is sequence of edges which connect all pairwise sets of vertices.*

5. *Let f and λ be an eigenfunction-eigenvalue pair, then for any $v \in V$ we have that*

$$\frac{1}{d_x} \sum_{y:y \sim x} (f(x) - f(y)) = \lambda f(x).$$

3.2.2 Weighted Graphs

Now we move away from the trivial case of unweighted graphs and introduce a weighting function, ω, on the edges of the graph. This will be vital to LE and SE algorithms as it will allow for embeddings that preserve certain intrinsic qualities of the high dimensional data. Now that the edges will have a nontrivial weight associated with them will need to revisit some of our previous definitions:

Definition 3.2.10 (Weighed Graph)**.** The weighted graph, $G = (V, E, \omega)$ is defined by a set of vertices (or nodes) $V = \{v_1, v_2, \ldots, v_N\}$, a set of ordered pairs of vertices, E, and a function $\omega : V \times V \to \mathbb{R}$. It is common to store all the weights produced by the weighting function in a $N \times N$ matrix, W such that $W(i, j) = \omega(v_i, v_j)$.

Definition 3.2.11 (Degree of Weighted Graph)**.** For a weighted graph $G = (V, E, \omega)$ with associated weight matrix W, the degree of a vertex $v \in V$ is $d_v = \sum_{u:u \sim v} \omega(v, u)$. It is common to store all the degrees for a graph G is a $N \times N$ matrix D such that $D(i, i) = d_i$.

The unweighted graph is thus a special case of the weighted graph where $W = A$; i.e. $\omega(u, v) = 1$ if $(u, v) \in E$ and 0 otherwise. With these definitions of the weight matrix and degree of a weighted graph, we can define the weighted graph Laplacian and the weighted normalized graph Laplacian:

Definition 3.2.12 (Graph Laplacian)**.** The graph Laplacian, L, of the graph $G = (V, E, \omega)$, $|V| = N$, is defined as $L = D - A$, or for $u, v \in V$,

$$L(u,v) = \begin{cases} d_u - \omega(u,u), & \text{if } u = v \\ -\omega(u,v), & \text{if } (u,v) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Definition 3.2.13. The normalized graph Laplacian, $\mathcal{L}$, of the graph $G = (V, E, \omega)$, $|V| = N$, is defined as $\mathcal{L} = D^{-1/2}AD^{-1/2}$, or for $u, v \in V$,

$$\mathcal{L}(u,v) = \begin{cases} 1 - \frac{\omega(u,u)}{d_u}, & \text{if } u = v \text{ and } d_u \neq 0 \\ -\frac{\omega(u,v)}{\sqrt{d_u d_v}}, & \text{if } (u,v) \in E \\ 0, & \text{otherwise} \end{cases}$$

Just as for the case of the unweighted $\mathcal{L}$, we can characterize the eigenvalues of the weighted $\mathcal{L}$, for example:

$$\begin{aligned} \lambda_1 &= \inf_{g \perp D^{1/2}\mathbf{1}} \frac{\langle g, g\mathcal{L} \rangle}{\langle g, g \rangle} \\ &= \inf_{f : \sum f(x) d_x = 0} \frac{\sum_{x \in V} f(x) L f(x)}{\sum_{x \in V} f^2(x) d_x} \\ &= \inf_{f : \sum f(x) d_x = 0} \frac{\sum_{x \sim y} (f(x) - f(y))^2 \omega(x,y)}{\sum_{x \in V} f^2(x) d_x}. \end{aligned}$$

and in general, λ_k is:

$$\begin{aligned}\lambda_k &= \inf_f \sup_{g \in P_{k-1}} \frac{\sum_{u \sim v}(f(u) - f(v))^2 \omega(x,y)}{\sum_v (f(v) - g(v))^2 d_v} \\ &= \inf_{f \perp DP_{k-1}} \frac{\sum_{u \sim v}(f(u) - f(v))^2 \omega(x,y)}{\sum_v f(v)^2 d_v}.\end{aligned}$$

3.2.3 Heat Kernel

The heat kernel, H_t, derives its name from being the fundamental solution to the heat equation:

$$\frac{\partial u}{\partial t} = -\mathcal{L}u.$$

For the graph G, H_t is the $N \times N$ matrix:

$$\begin{aligned}H_t &= e^{-t\mathcal{L}} \\ &= I - t\mathcal{L} + \frac{t^2}{2!}\mathcal{L} - \frac{t^3}{3!} + \ldots, \qquad (3.9)\end{aligned}$$

where (3.9) is a Taylor expansion. If we let $t = 1$ and use a first-order approximation we get:

$$\begin{aligned}H_1 &\approx I - \mathcal{L} \\ &= I - I + D^{-1/2} L D^{-1/2} \\ &= D^{-1/2} L D^{-1/2}.\end{aligned}$$

The heat kernel allows us to study diffusion on graphs which is a means to identifying important graph connections and preserving them in a lower dimensional space.

3.2.4 Laplace-Beltrami Operator

The graph Laplacian is analogous to the Laplace-Beltrami operator on manifolds. Let $\mathcal{M}$ be a smooth, compact, finite dimensional Riemannian manifold. The eigenvalues of the Laplace-Beltrami operator on Riemannian manifolds can be expressed as:

$$\lambda_{\mathcal{M}} = \inf_{\int_{\mathcal{M}} f = 0} \frac{\int_{\mathcal{M}} |\nabla f|^2}{\int_{\mathcal{M}} |f|^2}.$$

3.3 Laplacian Eigenmaps (LE)

Belkin and Nyogi were inspired to develop the Laplacian Eigenmaps algorithm by the observation that points in a high dimensional space that are close should also be close in a lower dimensional representation [9]. In the LE algorithm, the data is assumed to lie on or near a d-dimensional manifold $\mathcal{M}$. For practical applications, It is not necessary to find the actual manifold, instead it suffices to only approximate it with a data dependent graph. Let $G = (V, E, \omega)$, defined on the data set X, be such a graph. In the LE algorithm, vertices are defined as the data points contained in X, i.e., $V = X$. The undirected edge set $E = \{e_i\}$, denotes the connections between vertices in the G. Finally the weight function $\omega : V \times V \to \mathbb{R}$ measures the similarities between vertices in the G.

The choices of E and ω are both important and affects the final lower dimensional representation. E can be defined as the set of all possible edges, thus making G a complete graph. This, however, is avoided in practice because a sparse graph yields computational advantages. Also, since the aim of the algorithm is to preserve

relationships between data points that are close in the higher dimensional space, including only edges between vertices that are close is consistent with this goal. Thus E is chosen using the closest neighbors for each data point using a distance measure, typically L^2. We will discuss this method further in Section 3.4.2 and 5.2. The weighting function, ω, must also be chosen as to best preserve the local information in the higher dimensional space. The heat kernel is a natural choice to use because it describes the evolution of temperature over a region with fixed boundary conditions and is widely used to study the spectrum of the graph Laplacian, which can be thought of as an information spreading process and thus makes a good choice for the constraint weights. Thus, the kernel, or weighting matrix W defined by the heat kernel, is:

$$W(i,j) = \omega(v_i, v_j) = \begin{cases} e^{-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}} & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{3.10}$$

where σ is a parameter which will be discussed in Section 3.4.2. Once an appropriate graph and weighting function on the graph have been defined, the cost function, C_{LE}, is defined to provide an optimal lower dimensional embedding:

$$C_{LE} = \underset{y^T D y = I}{\arg\min} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \|y_i - y_j\|_2^2 W(i,j). \tag{3.11}$$

We can see that the minimization of the cost function will force points y_i and y_j to be close in the lower dimensional space if $W(i,j)$ is large, which by the weighting function is guaranteed if their high dimensional distance is small. The constraints here avoid the trivial solution and remove affine solution families. With

linear algebra manipulation, it can be shown that $\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\|y_i - y_j\|_2^2 W(i,j)$ is equivalent to the trace(YLY^T),

$$\begin{aligned}
C_{LE} &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\|y_i - y_j\|_2^2 W(i,j) \\
&= \frac{1}{2}\sum_{p=1}^{d}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(y_i(p) - y_j(p)\right)^2 W(i,j) \\
&= \frac{1}{2}\sum_{p=1}^{d}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(y_i^2(p) - 2y_i(p)y_j(p) + x_j^2(p)\right) W(i,j) \\
&= \frac{1}{2}\sum_{p=1}^{d}\left(\sum_{i=1}^{N} y_i^2(p)\sum_{j=1}^{N} W(i,j) - 2\sum_{i=1}^{N}\sum_{j=1}^{N} y_i(p)y_j(p) + \sum_{j=1}^{N} y_j^2(p)\sum_{i=1}^{N} W(i,j)\right) \\
&= \frac{1}{2}\sum_{p=1}^{d}\left(\sum_{i=1}^{N} y_i^2(p)D(i,i) - 2\sum_{i=1}^{N}\sum_{j=1}^{N} y_i(p)y_j(p) + \sum_{j=1}^{N} y_j^2(p)D(j,j)\right) \\
&= \frac{1}{2}\sum_{p=1}^{d}\left(2e_p(Y)De_p(Y^T) - 2e_p(Y)We_p(Y^T)\right) \\
&= \sum_{p=1}^{d}\left(e_p(Y)Le_p(Y^T)\right) \\
&= \text{trace}(YLY^T),
\end{aligned}$$

where e_p is a function which gives the pth coordinate for the embedded point. Now that minimizing C_{LE} can be expressed as minimizing the trace,

$$C_{LE} = \arg\min_{y^T Dy = I} \text{trace}\, yLy^T,$$

the Courant-Fisher Min-Max Theorem [115], which is stated as follows.

Theorem 3.3.1 (Courant-Fisher Min-Max Theorem)**.** *The eigenvalues of a Hermitian matrix A are characterized by the relation*

$$\lambda_k = \min_{S, dim(S)=n-k+1}\ \max_{v\in S, v\neq 0} \frac{\langle Av, v\rangle}{\langle v, v\rangle}.$$

Theorem 3.3.1 allows us to solve for the principle (smallest non-zero) eigenvalues and associated eigenvectors.

To summarize, the LE algorithm is divided into three main steps:

1. Construct a graph representation G of the data set X by defining edges as the kNN for each data point. Symmetrize the graph as $G = \max(G, G^T)$.

2. Weight the graph by the heat kernel to create the weights matrix W. Calculate the graph Laplacian as $L = D - W$, where D is the degree matrix.

3. Minimize the cost function in (3.11).

3.3.1 Convergence Results for Laplacian Eigenmaps

After the successful application LE to a wide-array of real world problems, Belkin and Niyogi revisited their algorithm and proved in [10] and [11] that the eigenvectors of the graph Laplacian converge to the eigenfunctions of the Laplace-Beltrami operator when the data is sampled from a uniform distribution on the embedded manifold. In the following paragraphs we will replicate the results found in the mentioned papers.

Let $\mathcal{M}$ be a smooth embedded k-dimensional manifold in $\mathbb{R}^D$ with induced Riemannian structure and measure μ. Now define the Laplacian operator, L^t : $L^2(\mathcal{M}) \to L^2(\mathcal{M})$, by

$$\hat{L}^t(f)(p) = (4\pi t)^{-(k+2)/2} \left(\int_{\mathcal{M}} e^{\frac{-\|p-x_i\|^2}{4t}} f(p) d\mu_q - \int_{\mathcal{M}} e^{\frac{-\|p-x_i\|^2}{4t}} f(x_i) d\mu_q \right),$$

and its discrete version as,

$$\hat{L}_N^t(f)(p) = \frac{(4\pi t)^{-(k+2)/2}}{N}\left(\sum_{i=1}^{N} e^{\frac{-\|p-x_i\|^2}{4t}} f(p) - \sum_{i=1}^{N} e^{\frac{-\|p-x_i\|^2}{4t}} f(x_i)\right).$$

In [10] it was shown that the graph Laplacian of a point cloud converged to the Laplace-Beltrami operator on the underlying manifold:

Theorem 3.3.2 (Belkin and Niyogi, 2005)**.** *Let $x_1, \ldots, x_N$ be sampled from a uniform distribution on a manifold $\mathcal{M} \subset \mathbb{R}^D$. Put $t_N = N^{-1/(k+2+\alpha)}$, where $\alpha > 0$ and let $f \in C^\infty(\mathcal{M})$. Then there is a constant C, such that in probability,*

$$\lim_{n\to\infty} C\frac{(4\pi t_N)^{-(k+2)/2}}{N} L_N^{t_N} f(x) = \Delta f(x).$$

It was however more difficult to show that the eigenvectors of the graph Laplacian converged to the eigenfunctions of the Laplace-Beltrami operator. In [11] Belkin and Niyogi showed that, if we assume that the points x_i are chosen independent and identically distributed (i.i.d.) from a uniform distribution on $\mathcal{M}$, then there exists a sequence t_N, such that eigenfunctions of $\hat{L}_N^{t_N}$ converge to the eigenfunctions of Δ in probability, this is stated in the following spectral convergence theorem.

Theorem 3.3.3 (Spectral Convergence, Belkin and Niyogi 2007)**.** *Let $\lambda_{n,i}^t$ be the ith eigenvalue of $\hat{L}_n^t$ and $e_{n,i}^t$ be the corresponding eigenfunction (which for each fixed i, will be shown to exist for t sufficiently small). Let λ_i and e_i be the corresponding eigenvalue and eigenfunction of Δ respectively. Then there exists a sequence $t_n \to 0$, such that*

$$\lim_{n\to\infty} \lambda_{n,i}^{t_n} = \lambda_i$$

$$\lim_{n\to\infty} \left\| e_{n,i}^{t_n}(x) - e_i(x) \right\|_2 = 0$$

where the limits are in probability.

To prove the Spectral Convergence theorem two parts must be satisfied, first it must be shown that there is spectral convergence of $\hat{L}_N^t \to L^t$ as $N \to \infty$, and second, spectral convergence of the functional approximation $L^t \to \Delta$ as $t \to 0$; these are stated in Theorem 3.3.4 and 3.3.5.

Theorem 3.3.4 (Belkin and Niyogi, 2007)**.** *Let λ_i, λ_i^t, e_i, e_i^t be the ith smallest eigenvalues and corresponding eigenfunctions of Δ and L^t respectively. Then*

$$\lim_{t\to 0} \left|\lambda_i - \lambda_i^t\right| = 0 \text{ and}$$

$$\lim_{t\to 0} \left\|e_i - e_i^t\right\|_2 = 0.$$

Theorem 3.3.5 (Luxburg, Belkin and Bousquet, 2004)**.** *For a fixed sufficiently small t, let $\lambda_{n,i}^t$ and λ_i^t be the ith eigenvalue of $\hat{L}_n^t$ and L^t respectively. Let $e_{n,i}^t$ and e_i^t be the corresponding eigenfunctions. Then*

$$\lim_{n\to\infty} \lambda_{n,i}^t = \lambda_i^t \text{ and}$$

$$\lim_{n\to\infty} \left\|e_{n,i}^t(x) - e_i^t(x)\right\|_2 = 0,$$

assuming that $\lambda_i^t \leq \frac{1}{2t}$. The convergence is almost sure.

The two Theorems, 3.3.4 and 3.3.5, can shown symbolically as:

$$\text{Eig}\hat{L}_N^t \xrightarrow[\text{probabilistic}]{N\to\infty} \text{Eig}L^t \xrightarrow[\text{deterministic}]{t\to 0} \text{Eig}\Delta$$

Finally the authors are able to show that there exists a sequence t_N which guarantees the convergence in Theorem 3.3.3.

3.4 Schrödinger Eigenmaps

In [48], Czaja and Ehler generalize LE by introducing a potential, V, on the graph. In general, V is a symmetric, positive semi-definite $N \times N$ matrix. We include a brief overview of the method here. For further details and proofs, see [48] and [75]. The graph Laplacian is replaced with

$$E = L + \alpha V,$$

where the parameter α indicates the relative significance of the potential with respect to the Laplacian operator. We do not directly incorporate the parameter α into the potential matrix as we would like to retain the ability to study a parameter family of SE embeddings. Accordingly, the cost function which needs to be minimized, C_{LE} becomes C_{SE}:

$$C_{SE} = \underset{y^T D y = I}{\arg\min} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \|y_i - y_j\|_2^2 W(i,j) + \alpha \sum_{i=1}^{N} \sum_{i=1}^{N} V(i,j) \|y_i - y_j\|_2^2. \quad (3.12)$$

As before, with linear algebra this can be shown to be equivalent to:

$$\begin{aligned} C_{SE} &= \underset{y^T D y = I}{\arg\min} \operatorname{trace}(Y E Y^T) \\ &= \underset{y^T D y = I}{\arg\min} \operatorname{trace}(Y (L + \alpha V) Y^T) \\ &= \underset{y^T D y = I}{\arg\min} \operatorname{trace}(Y L Y^T) + \alpha \operatorname{trace}(Y V Y^T). \end{aligned}$$

As we showed in the last section, the graph Laplacian under certain conditions will converge to the Laplace-Beltrami operator. It is possible to make the same arguments with the Schrödinger operator, see [48] for the proof.

We will consider two types of potentials: barriers and clusters.

A barrier potential, V, is a nonnegative diagonal matrix. The barrier potential is designed to stop diffusion from the heat kernel as certain identified graph vertices. With the barrier potential we can simplify C_{SE}:

$$\arg\min_{y^T Dy=I} \frac{1}{2} \sum_{i,j} \|y_i - y_j\|_2^2 W_{i,j} + \alpha \sum_i V(i)\|y_i\|_2^2.$$

The diagonal potential thus introduces a penalty term at each nonzero entry. This can be seen as creating a well at the corresponding vertices. More precisely, if $V(i) \neq 0$ there exists a constant C such that,

$$\|y_\alpha(i)\|^2 \leq \frac{C}{V(i)\alpha}.$$

Thus, as $\alpha \to \infty$, $y_\alpha(i) \to 0$. Since the SE algorithm, like the LE algorithm, must preserve local geometry, the neighbors of these vertices are also pulled toward 0 as well.

Let b be an index set and let $\beta = \{v_{b(1)}, \ldots, v_{b(m)}\}$ be a collection of vertices. A cluster potential over β is defined by taking V to have zeros everywhere except for the submatrix defined by β, where

$$V[b,b] := \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}.$$

For such a V, (3.12) is equivalent to

$$C_{SE} = \arg\min_{y^T Dy=I} \frac{1}{2} \sum_{i,j} \|y_i - y_j\|^2 W_{i,j} + \alpha \sum_{k=1}^{m-1} \|y_{i_k} - y_{i_{k+1}}\|^2.$$

The penalty term has the effect of sequentially identifying points in β; as $\alpha \to \infty$,

$$\sum_{k=1}^{m-1} \|y_{i_k} - y_{i_{k+1}}\|^2 \leq \frac{C}{\alpha} \to 0.$$

Again, the embedding must preserve local geometry, and so neighbors of β are pulled inwards as well, but instead of toward 0, they will be pulled toward each other in a chain. Thus, the potential forces an increased amount of clustering around β.

We can even move away from sequentially identifying points and create off-diagonal potentials which realize complete graphs between a subset of points. This formulation would be

$$V[b,b] := \begin{pmatrix} m & -1 & -1 & \dots & -1 \\ -1 & m & -1 & \dots & -1 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ -1 & -1 & \dots & m & -1 \\ -1 & -1 & -1 & \dots & m \end{pmatrix}.$$

For such a V, (3.12) is equivalent to

$$C_{SE} = \underset{y^T D y = I}{\arg\min} \frac{1}{2} \sum_{i,j} \|y_i - y_j\|^2 W_{i,j} + \alpha \sum_{k=1}^{m-1} \|y_{i_k} - y_{i_{k+1}}\|^2.$$

In this case, the identified data points would be collapsed to their centroid in the feature space as α becomes large.

To summarize the SE algorithm is divided into three main steps:

1. Construct a graph representation, G, of the data set X by defining edges as the kNN for each data point. Symmetrize the graph as $G = \max(G, G^T)$.

2. Weight the graph by the heat kernel to create the weights matrix W. Calculate the graph Laplacian as; $L = D - W$ where D is the degrees matrix. Define a potential matrix V and constant α and calculate the Schrödinger $E = L + \alpha V$.

3. Minimize the cost function in (3.12).

3.4.1 A Toy Example To Illustrate The Use Of Potentials

To illustrate the use of diagonal and non-diagonal potentials we look at the function

$$f(x) = 1 - 2x^2,$$

placed into $\mathbb{R}^3$ by $(x, y, z) = (x, f(x), 0)$. We sample the function uniformly in x, $[-1, 1]$, 201 times (interval width of 0.01). In Figure 3.2 (a) we see the original data set. Using the LE algorithm we can produce the 2-dimensional representation of the data set seen in 3.2 (b); this is trivial due to the construction of the three dimensional data set lying on the x-y coordinate hyperplane. In Figure 3.2 (c)-(f) we apply a diagonal potential,

$$V(i, j) = \begin{cases} 1, & i, j = 101 \\ 0, & \text{otherwise,} \end{cases}$$

to the $x = 0$ coordinate. We can see that as α increases in value the $x = 0$ coordinate in the embedding gets pulled toward the origin. As the $x = 0$ coordinate is connected with its neighbors in the graph construction, the neighbors of $x = 0$ also get pulled toward the origin. In Figure 3.2 (g)-(j), we apply a cluster potential between the

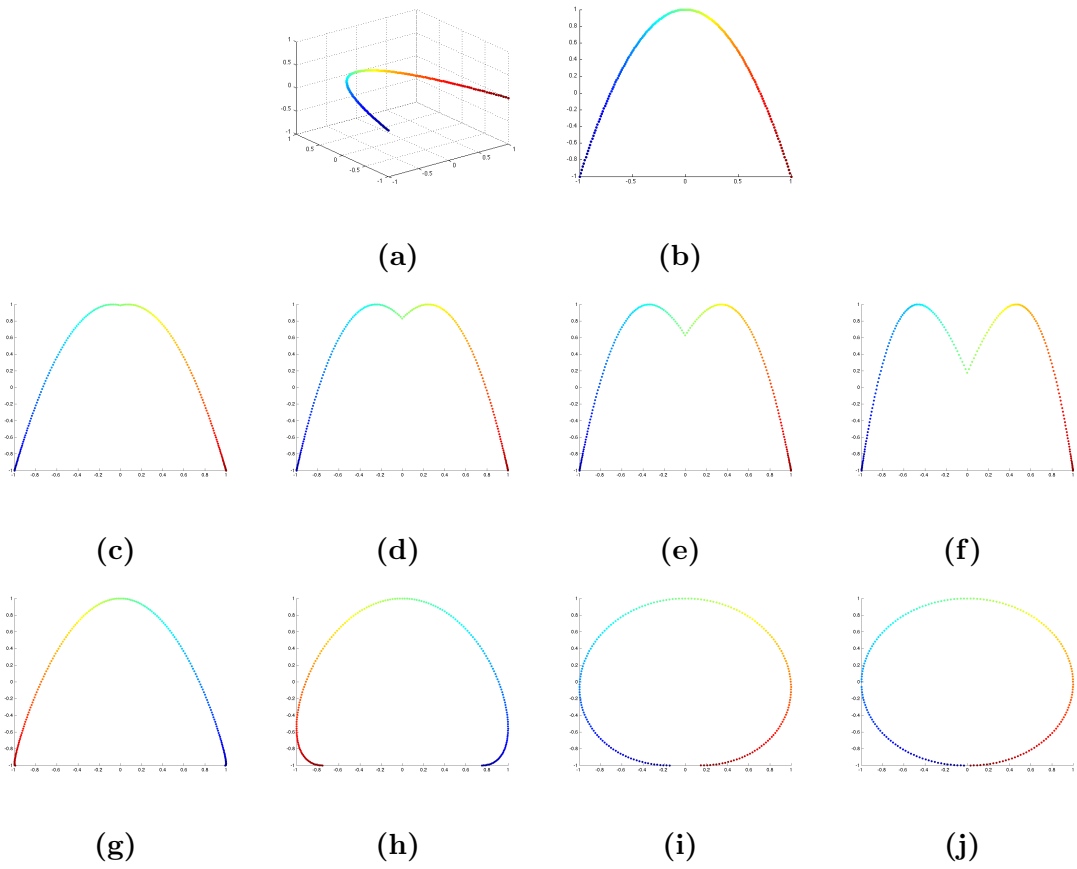


Figure 3.2: *(a) Original data, (b) LE representation (c)-(f) SE with diagonal potential placed at* $x = 0$, $\alpha = 0.01, 0.05, 0.1, 0.5$. *(g)-(j) SE with cluster potential placed between at* $x = -1$ *and* $x = 1$, $\alpha = 0.001, 0.01, 0.1, 1.0$.

$x = -1$ and $x = 1$ coordinates,

$$V(i,j) = \begin{cases} -1, & i = 1 \text{ and } j = 201 \text{ or } i = 201 \text{ and } j = 1 \\ 1, & i, j = 1 \text{ or } i, j = 201 \\ 0, & \text{otherwise.} \end{cases} \tag{3.13}$$

In this case, as α increases, the two points in the embedding are pulled together, toward their centroid.

3.4.2 A Brief Discussion on Intrinsic Dimension, Heat Kernel Parameter, and k-Nearest Neighborhood Construction

In this section we will discuss 3 parameters which are common to the LE and SE algorithms.

The dimension which we seek to project into d, must be chosen to carefully balance the computational cost of calculating extra eigenvectors and the added benefit of being able to distinguish data points better. This optimal projection dimension, also known as the intrinsic dimension, is difficult to choose. With toy examples, such as the Helix presented in Figure 3.1, the intrinsic is easy to realize, in this case 2. Of the algorithmic predictors of intrinsic dimension, there exist, q-Dimension, Capacity dimension, Information dimension, Correlation dimension, PCA and local PCA. A summary of all of these techniques can be found in [93]. In practice we have found that the PCA dimension, which calls for looking at the spectrum from PCA, gives the only practical results.

The parameter σ in the heat kernel control the spread of information in the graph. A small value will force the cost function to concentrate on only the nearest of the kNN. A larger value will allow for a more balanced approach; closer kNN will still play a larger role in the minimization, but further kNN will have a non-trivial effect. This parameter is chosen in practice either experimentally or algorithmically. By looking at the effect different values have on the histogram of the kernel, an experimentally derived value can be found. An algorithmic solution is to take the mean or median row sum of the kernel.

There are two basic ways to select nearest neighbors, either ϵ-balls or kNN. ϵ-balls choose nearest neighbors for a point x as all other data points a distance ϵ away. This is problematic to implement in an approximate fashion, as we will discuss in Chapter 5.2, and the sparsity level of the resultant kernel matrix is difficult to control. kNN, on the other hand is relatively easy to implement and gives an exact sparsity level. Choosing the number k is also vital, as we want to balance the computational burdens of having a less sparse graph with fully describing the connections between data points in the form of edges. In practice we have found that choosing a k that is slightly larger than the minimum k required to have a fully connected graph produces adequate classification results. Also related to the choice of the parameter k is the procedure used to symmetrize the kernel resultant for the graph. kNN neighborhood construction algorithms, unlike ϵ-ball algorithms do not produce undirected graphs, so a method must be devised so as to symmetrize the kernel. We have chosen to simply remove all edge directedness, by adding all edges necessary to ensure a undirected graph. Other options are to only consider an edge if two points are mutual kNN. This procedure will often result in non-connected graphs so adding a spanning tree is required. Another choice, which we will explore more fully in Chapter 7, is which distance measure to use to determine kNN: L^p, vector angle, Mahalanobis, are all possible choices. Again, we have found that L^2 works well with a wide variety of data.

Chapter 4

The Preimage Problem

In Sections 3.1 to 3.4 we mainly focused on the feature space; this was because in the feature space it becomes easier to make comparisons between data points. As a result of the dimension reduction process, the data becomes organized and concentrated in less dimensions. However, it is also interesting to consider the mapping back from the feature space to the original space. We will refer to this mapping as the preimage [117]. We see many possible applications to this idea, for example if two data sources are fused by an analyst in a feature space, it would be illustrative to see what that fusion would look like in a more natural space to the analyst, i.e., the original data space. Thus, the preimage offers a means to use feature space based fusion with the promise that the fused product can be viewed in the original space if desired.

4.1 Kernel PCA Preimage

With the popularity of non-linear manifold learning techniques such as KPCA and LE surpassing those of linear techniques, such as PCA, vast improvements have been seen in many fields of study such as classification and feature extraction. One problem however with these non-linear techniques is the lack of an easily calculable preimage. Unlike with PCA, which is an invertible method, non-linear techniques

are not invertible and generally an exact preimage does not exist. Recent work has however shown that it is possible to find an approximate preimage for a feature space constructed using KPCA.

First let us define a preimage.

Definition 4.1.1 (Preimage)**.** Given a dimension reduction mapping, $\Phi : \mathbb{R}^D \mapsto \mathbb{R}^d$, the preimage of $y \in \mathbb{R}^d$ is $x \in \mathbb{R}^D$ such that $\Phi(x) = y$.

The difficulty of finding the preimage is that it may not always exist. In [103] the authors illustrate this with a dataset made up of a line in $\mathbb{R}^D$. Since the dataset is intrinsically two-dimensional, after the dimension reduction mapping,

$$\Phi : \mathbb{R}^D \mapsto \mathbb{R}^2,$$

all the qualities of the dataset would be preserved by a line in $\mathbb{R}^2$. Now, in the feature space, insert a point which does not lie on the 2-dimensional line; the preimage of this point does not exist. Since it is possible to develop situations where the preimage does not exist, instead of trying to find an exact preimage, we should look for one which is merely close, i.e. an approximate preimage.

Definition 4.1.2 (Approximate Preimage)**.** Given a dimension reduction mapping, $\Phi : \mathbb{R}^D \mapsto \mathbb{R}^d$, the approximate preimage of of $y \in \mathbb{R}^d$ is $x \in \mathbb{R}^D$ such that

$$x = \underset{x \in \mathbb{R}^D}{\arg\min} \|\Phi(x) - y\|_2. \tag{4.1}$$

Finding an approximate preimage by minimizing the distance in (4.1) has been studied in the past but it requires a closed form definition of the mapping, which

we don't have. In [1], an application of the Nyström out of sample extension allows for the solution to the preimage problem to be phrased as:

$$x = \arg\min_{x \in \mathbb{R}^D} \|\mathcal{E}K_x - y\|_2. \tag{4.2}$$

where, $\mathcal{E} \in \mathbb{R}^{m \times n}$ is the Nyström out of sample extension and K_x is the kernel vector evaluation of the point x, i.e., $K_x = [K(x, x_1), \ldots, K(x, x_N)]$ and $(K_x)_i = K(x, x_i)$. Next we will need the definition of the Moore-Penrose pseudoinverse.

Definition 4.1.3 (Moore-Penrose pseudoinverse)**.** The Moore-Penrose pseudoinverse of $m \times n$ matrix A, is $A^\dagger$ such that the following hold

1. $AA^\dagger A = A$
2. $A^\dagger AA^\dagger = A^\dagger$
3. $(AA^\dagger)^* = AA^\dagger$
4. $(A^\dagger A)^* = A^\dagger A$.

Now, using the Moore-Penrose pseudoinverse, we are able to solve (4.2) for an optimal K_x:

$$\hat{K}_x = \mathcal{E}^\dagger y. \tag{4.3}$$

For KPCA, see [118], we know that the kernel matrix has the form:

$$K(x, y) = e^{-\frac{\|x-y\|_2^2}{2\sigma^2}}. \tag{4.4}$$

Now if we apply the log function to (4.4) we are able to state the distance between

data points as,

$$\begin{aligned} \|x-y\|_2^2 &= -2\sigma^2 \log(K(x,y)) \\ &= -2\sigma^2 \log(\hat{K}(x,y)), \end{aligned} \tag{4.5}$$

where (4.5) comes from (4.3). In our compact notion (4.5) becomes:

$$\|x - x_i\|^2 = -2\sigma^2 \log((\hat{K}_x)_i). \tag{4.6}$$

The preimage problem can now be solved using MDS in [87].

4.2 Laplacian Eigenmap Preimage

Due to the nearest neighbor condition in LE, the relation (4.6) is not readily available. This is because $(K_x)_i$ is a function of $\|x - x_i\|_2$ for all $x_i \in \mathcal{N}(x)$, the nearest neighbors of x. We are thus forced to develop new techniques. The problem can be traced to Nyström extension of the LE kernel [20], which we will discuss in Chapter 7:

$$K(x, x_i) = \frac{k(x, x_i)}{\sqrt{\sum_{j=1}^{N} \tilde{k}(x, x_j) \sum_{j=1}^{N} \tilde{k}(x_i, x_j)}},$$

where $\tilde{k}(x, x_j)$ is the kernel restricted to the nearest neighbors of x. In the denominator,

$$\sum_{j=1}^{n} \tilde{k}(x, x_j),$$

$(K_x)_i$ depends simultaneously on all $\|x - x_j\|_2$, $j \in \{1, ..., N\}$. Since we have a sparse kernel defined by the nearest neighbors, let us specify that there are c many

neighbors for x, i.e., c many non-zeros. To allow for easier calculations let:

$$\begin{aligned} e_j &= k(x, x_j) \\ &= e^{-\|x-x_j\|_2^2/2\sigma^2}. \end{aligned}$$

Now let $e = [e_1, \ldots, e_c]$ be the vector holding all the nearest neighbors weights which contribute to K_x, with

$$(K_x)_{i_j} = \frac{e_j}{\sqrt{\sum_l e_l \sum_{x_l \in \mathcal{N}(x_{i_j})} k(x_{i_j}, x_l)}}. \tag{4.7}$$

We are able to manipulate (4.7),

$$\begin{aligned} \frac{1}{(K_x)_{i_j}} &= \frac{\sqrt{\sum_l e_l}\sqrt{\sum_{x_l \in \mathcal{N}(x_{i_j})} k(x_{i_j}, x_l)}}{e_j} \\ \frac{e_j}{\sqrt{\sum_l e_l}} &= (K_x)_{i_j} \sqrt{\sum_{x_l \in \mathcal{N}(x_{i_j})} k(x_{i_j}, x_l)} \\ \frac{e_j^2}{\sum_l e_l} &= ((K_x)_{i_j})^2 \sum_{x_l \in \mathcal{N}(x_{i_j})} k(x_{i_j}, x_l) \\ e_j^2 &= a_j \sum_l e_l, \end{aligned}$$

to obtain a system of c equations

$$e_j^2 - a_j^2 \sum_l e_l = 0, \tag{4.8}$$

$1 \leq j \leq c$, where

$$a_j = (K_x)_{i_j} \sqrt{\sum_{x_l \in \mathcal{N}(x_{i_j})} k(x_{i_j}, x_l)}.$$

The non linear system of equations in (4.8) now needs to be solved. The first obvious choice is Newton's method, which can be used to find the solution if conditioned to avoid the trivial solution. However, we can do better by using

inequality constrained quadratic programming because we can take advantage of the structure in the LE kernel. Let us rewrite (4.8) as,

$$\min_{e} \|\mathcal{A}(e)\|_2,$$

where $\mathcal{A}_i(e) = e_i^2 - a_i^2 \sum_j e_j$.

If we now assume that the a_i's are ordered such $a_1 \geq a_2 \geq ... \geq a_c$ we can constrain the minimization such that we get $1 \geq e_1 \geq ... \geq e_c \geq 0$. This ordering is naturally what we would expect in the kernel amongst the nearest neighbors. Let us now define a matrix B to hold our constraints:

$$B_{i,j} = \begin{cases} -1 + \alpha(i) & \text{if } j = i, \\ 1 & \text{if } j = i + 1, \\ 0 & \text{otherwise,} \end{cases}$$

where α is a function which controls the spread of the weights in the kernel. We can learn such a function by studying the distribution of the weights in the rows of the kernel, but for simplicity, let α be the constant function, $\alpha(i) = \bar{\alpha}$. Finally let the inequality contained quadratic programming solution to (4.8) be:

$$\min_{e, B(e)<0} \|\mathcal{A}(e)\|_2. \tag{4.9}$$

Given a means now to solve for the e_j's, we can find the distances in the original space for the neighbors of x:

$$\|x - x_{i_j}\|_2^2 = -2\sigma^2 \log(e_j), \quad x_{i_j} \in \mathcal{N}(x).$$

We have now shown a means to solve for distances in the original space, we now proceed by the methods outlined in [87] to calculate x.

Table 4.1 demonstrates the advantages of using the inequality constrained quadratic programming approach over Newton's method. In this experiment, the true $e_j = k(x, x_j)$ were generated as uniform random variables with $\frac{2}{3} < e_j < 1$. The kNN was set to 25. Noise of various standard deviations, where the standard deviation is determined as a percentage of maximum entry in K_x, were added to K_x after it is calculated from e_j, $j \in \{1, ..., 25\}$. The error is taken to be $\|\widehat{e} - e\|_2 / \|e\|_2$. We can see from the results that the inequality contained quadratic programming outperforms Newton's method in its stability in the presence of noise. For future calculations we will make use of (4.9).

	$\sigma = 2\%$	$\sigma = 4\%$	$\sigma = 6\%$
Newton's Method	0.0269	0.0575	0.0837
Constrained Quadratic Programming	0.0187	0.0347	0.0487

Table 4.1: *Comparison of Newton's Method to Constrained Quadratic Programming in the solution of* (4.8)*. Columns are the added noise corresponding to p% of the max of* K_x*.*

In [40] an extension to this method is offered. To deal with the noise inherent in the process, an L_1 regularization scheme is applied to the techniques developed here.

In Table 4.2 we look at denoising the MNIST dataset [91] seen in Figure 4.1 with KPCA, LE, and LE with L_1 regularization preimage techniques. This dataset is composed of a collection of 28×28 gray scale hand written digits from 0 to 9.

For each digit the dataset contains 2000 images, we select at random 200 of these to train our system. The system was trained by embedding the set of images, treated as a matrix of size 2000×28^2, i.e. each image is converted to a 1×28^2 vector. In the dimension reduction procedure the images are observations and the pixels contained in the image are the variables. Now, given a non training sample, we apply white Gaussian noise, project it into the feature space using a Nyström out of sample extension, see Chapter 7, and then use the preimage routines to pull back the sample into the image space. Since the preimage was calculated using nearest neighbors which do not have noise added, when the sample is pulled back to the image space it should be denoised. Once back in the image space we can use the SNR to evaluate the denoising procedure.

Figure 4.1: *A collection of ten of each digit from the MNIST Digits database.*

	$\sigma^2 = .2$			$\sigma^2 = .4$			$\sigma^2 = .6$		
Digit	KPCA	LE	LE(L_1)	KPCA	LE	LE(L_1)	KPCA	LE	LE(L_1)
0	4.28	2.38	5.38	4.08	2.44	5.09	4.20	1.41	4.75
1	5.37	3.77	4.85	5.02	3.72	4.64	5.13	3.84	4.45
2	4.27	2.33	5.12	3.92	2.28	4.76	3.68	2.32	4.54
3	4.17	1.49	4.73	4.02	2.20	4.30	3.94	1.61	4.32
4	3.66	2.45	4.65	3.66	2.15	4.32	3.37	1.63	3.72
5	3.54	1.92	4.63	3.48	1.37	4.39	3.35	2.57	3.99
6	4.20	2.05	5.41	3.98	2.16	5.07	3.99	1.61	4.85
7	4.33	3.23	4.85	4.35	2.43	4.50	4.04	2.92	3.90
8	3.68	1.80	4.44	3.33	2.36	4.16	3.55	1.97	4.10
9	3.97	2.62	4.72	3.76	1.58	4.25	3.67	2.49	4.01

Table 4.2: *SNR for Kernel PCA, LE, and LE with L_1 Regularization (LE(L_1))*

The results in Table 4.2 make it clear that the LE preimage, especially with L_1 regularization, is a comparable method to KPCA. The LE with L_1 regularization preimage is only beaten by the KPCA preimage for the 1's digit; this is to be expected since the digit one is linear and KPCA linearizes the underlying manifold. LE has the advantage of being computationally less costly then KPCA; as LE utilizes a sparse kernel and KPCA does not. As the number of elements in the dataset or pixels in the image increase, even the non L_1 regularized version of the LE preimage routine becomes appealing. It it also advantageous to work with LE, as opposed

to KPCA, because LE in reality is a feature extraction tool. The fact that LE without L_1 regularization was comparable to KPCA, and LE with L_1 regularization outperformed KPCA in most cases with reduced dimensions is very promising.

We are currently looking at the preimage technique for the application of inpainting and fusion of remote sensing data, see [40]. The preimage would be used in conjunction with the feature space rotation fusion method which we will discuss in Chapter 6. We note that the preimage of the Schödinger Eigenmaps analysis is a direct extension of the preimage method built here.

Chapter 5

Numerical Acceleration Methods for Dimension Reduction

Using the LE algorithm on large data sets and the problem of data fusion are intimately tied. For our theories on data fusion to be practical, they must be able to scale with the size of the data sources encountered. As with the problem of high dimensionality in data, which we solved using dimension reduction, the problem of high sample count can also be dealt with by harmonic analysis.

Nonlinear dimension reduction techniques, like the ones discussed in Chapter 3, can be summed up in three steps:

1. Select a representative collection of nearest neighbors for each data point.

2. Assign an appropriate weight to each pair of nearest neighbors thus representing the data as a weighted graph.

3. Minimize a cost function on the graph to find the lower dimensional representation .

Steps 1 and 3 in the above enumeration are computationally expensive, so we will seek means to accelerate their computation. Step 2 in practice is usually calculated as a byproduct of Step 1 thus by addressing Step 1 we will indirectly address Step 2.

For the first step, assuming that the dimensionality of the data is D, an exact nearest neighbor search requires the computing of N^2 distances, where N is the

number of data points; this results in a complexity of $O(DN^2)$. Big-Oh, $O(\cdot)$, is a means to measure the computational complexity of an algorithm. It indicates, in the limit case, the number of operations that an algorithm needs to execute before completing and gives a general idea of the effect on execution time of increasing the input data size. We can alleviate this computational complexity by performing an approximate nearest neighbor search. This will be discussed in Section 5.2 and will allow us to lower the exponent for the nearest neighbor search complexity from 2 to a number in the range (1,2). We can also reduce the length of the vectors we are measuring the distance between, from D to $M < D$, by using Random Projections; we will discuss this in Section 5.1. Random Projections will reduce the constant D to M with only a small amount of additional overhead.

For the third step, the eigendecomposition of the resulting kernel matrix has complexity $O(N^3)$. The Nyström method, which we will be discussed in Chapter 7, offers a way to approximate the eigendecomposition in much less time. This is accomplished by computing the eigendecomposition on only a small sub matrix of the kernel matrix made up of Nyström or landmark points. This result can then be extend, using the Nyström extension, to the rest of the kernel matrix. The Nyström method will reduce the base of the exponent from N to ℓ, where is ℓ is the number of landmarks chosen; thus reducing the computational cost significantly.

5.1 Random Projections

Random Projections, borrowing its theory from the field of compressive sensing/sampling (CS), seeks to project the data from a D dimensional space to an M dimensional space. The projection is linear and is not data dependent, only the parameter M must be chosen. The theory of CS states that every signal $x_i \in X \subset \mathbb{R}^D$ which is K-sparse can be recovered with high probability using M linear measurements z_i, $z_i = x_i\Phi$ where Φ is $M \times N$ independent and identically distributed Gaussian entries.

Using the Theorem 5.1.1 from [75], which gives a probabilistic bound for the error associated with using Random Projections with Laplacian Eigenmaps, and the work in [4] we can use Random Projections as a preprocessing step to reduce the dimensionality of the data set and then apply the Laplacian Eigenmaps algorithm.

Theorem 5.1.1 (Halevey, 2011)**.** *Given a data set* $\{x_1, \ldots, x_N\} \in \mathbb{R}^D$*, sampled from a compact d-dimensional Riemannian manifold, assume* $0 < \|x_i - x_j\| \leq A < \infty, \forall i \neq j$*. Let* $0 \leq \lambda_1 < \cdots < \lambda_d$ *be the first d non-zero eigenvalues computed by Laplacian Eigenmaps algorithm, assumed simple, with r the minimum eigengap and let* f_i *be a normalized eigenvector corresponding to* λ_i*. Use a random orthoprojector* Φ *to map the points to* $\mathbb{R}^M$*. Let* $\hat{f}_j$ *be the* j^{th} *eigenvector computed by Laplacian Eigenmaps for the projected data set. Fix* $0 < \alpha < 1$ *and* $0 < \rho < 1$*. If*

$$M \geq \frac{4 - 2\ln(1/\rho)}{\epsilon^2/200 + \epsilon^3/3000} K \ln\left(\frac{CKD}{\epsilon}\right), \quad \textit{where} \quad \epsilon = \frac{r\alpha}{4AN(N-1)},$$

then, with probability at least $1 - \rho$*,* $\|f_i - \hat{f}_i\| < \alpha$*.* $C = \frac{1900RV}{\tau^{1/3}}$ *where R is the geodesic covering regularity, V is the volume, and* τ^{-1} *is the condition number of*

the manifold.

Projecting the data from D to M dimensional space using random projections requires only the complexity of a matrix-matrix product, but offers a savings of $O(N^2(D - M))$ when computing the nearest neighbors required in the graph construction.

In Figure 5.1, we decrease the ratio between M and D for the Laplacian Eigenmaps analysis of the Indian Pines and measure percent time change and percent accuracy change in classification. The time change decreases gradually while the percent accuracy change remains approximately constant for all but the smallest ratio.

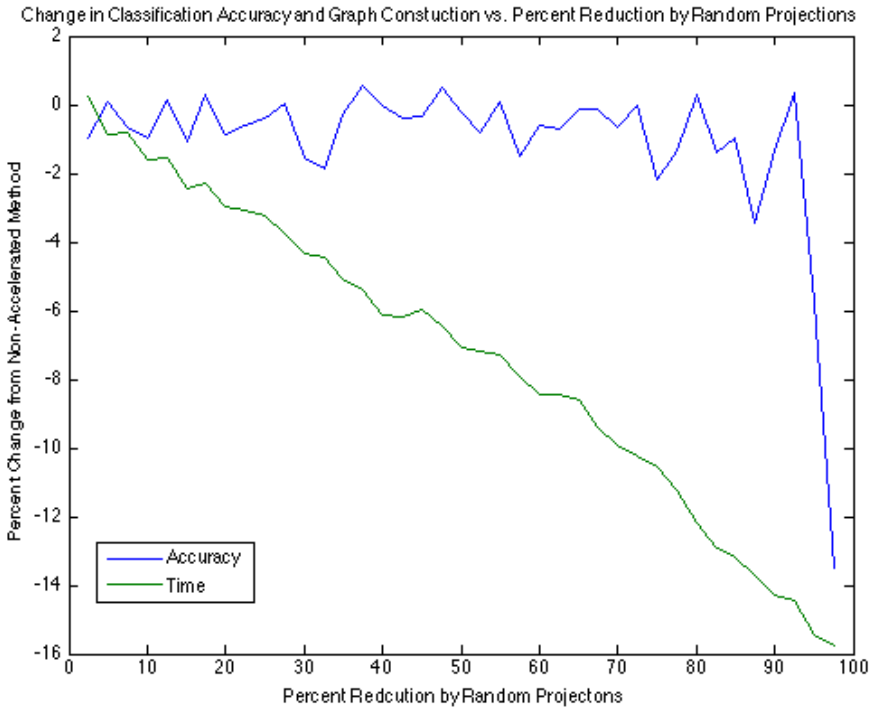


Figure 5.1: *Time savings and percent change in classification accuracy for Indian Pines as a function of percent reduction by Random Projections.*

5.2 Approximate Nearest Neighbors

As previously discussed, one of the most computationally expensive steps during any nonlinear dimension reduction procedure is choosing the kNN for the graph construction. To find the kNN for a set of N data points in D dimensional space in an exact fashion requires $O(DN^2)$ time because we are finding N^2 inner-products of length D.

There are number of approximate nearest neighbor algorithms that have been developed to deal with the large computational cost. For example, Locality-sensitive Hashing [52], Best Bin First [6], Clustered Point Sets Search [2], and Divide and Conquer with Recursive Lanczos Bisection [35]. For this thesis we will look at the Divide and Conquer with Recursive Lanczos Bisection. The basic idea of the method is to iteratively split the data set into two overlapping subsets until the subsets become small enough such that a brute force calculation is feasible. These subsets are then rejoined and only the kNN are retained for each data point. We can think of the algorithm as having three main phases that are recursively called.

The Divide phase is responsible for splitting the data into approximately equal subsets. This is accomplished by using the leading eigenvector of the Singular Value Decomposition (SVD) of the centered data. First mean center X to form $\tilde{X}$:

$$\tilde{X} = X - m_X \mathbf{1},$$

where m_X is the $D \times 1$ spectral mean of X. Now we find the SVD of $\tilde{X}$,

$$\tilde{X} = U \Sigma V^T.$$

Let $(u_\ell, \sigma_\ell, v_\ell)$ denote the largest triplet. If we define a hyperplane as $\langle u, x\rangle = 0$, $\tilde{X}$ is divided into two sets based on the sign of $u^T\tilde{x_i}$. We note that the hyperplane, $\langle u, x\rangle = 0$, maximizes the sum of the squared distance from each point $\tilde{x_i}$ to the hyperplane,

$$\sum_{i=1}^{N} = \left\|u^T\tilde{X}\right\|_2^2 = \left\|\sigma v^T\right\|_2^2 = \sigma^2.$$

Noting the relationship, $u^T\tilde{x_i} = \sigma v$, and that $\sigma > 0$, it is convenient then to divide X based on the sign of v. To find the largest triplet, the Lanczos algorithm [46] can be used.

The Lanczos algorithm constructs an orthonormal basis, Q, from a random unit vector q and small integer s to the Krylov subspace,

$$Q = [q, (\tilde{X}^T\tilde{X})q, \ldots, (\tilde{X}^T\tilde{X})^{s-1}q].$$

Then, if we let $T = Q^T(\tilde{X}^T\tilde{X})Q$, the largest triplet of T will approximate the largest triplet of $\tilde{X}$ in only $O(sDN)$.

However, for data which is close to the hyperplane, it is possible that its nearest neighbors lie on either side of the hyperplane. The inclusion of the overlap parameter, α, allows for points near the hyperplane to be included in both subsets. This is accomplished splitting the data not based on the sign of the coefficients but by $\leq t$ and $\geq -t$ where t is the αth percentile of the absolute value of the coefficients.

The Brute phase is performed once the now split data sets have size less than a chosen parameter β. This threshold, β, is chosen based on the computer architecture to balance taking unnecessary further Divide phases with the large cost of computing

all pairwise distances. The Brute phase retains only the $(k+1)$NN for each data point, the requested kNN and itself.

The Conquer phase is performed once the Brute phase has been completed for both divided data sets. During the Conquer phase any data points that were included in the overlap section have their kNN chosen from distances retained during each Brute phase.

The ANN algorithm is summarized in pseudo code in Algorithm 5.2.1, The time complexity of the ANN algorithm is summarized in the following theorem.

Theorem 5.2.1 (Cormen[45], 2001)**.** *The time complexity for the divide and conquer method is $O(DN^{t_0})$, where*

$$t_0 = \log_{\frac{2}{1+\alpha}} 2 = \frac{1}{1 - \log_2(1+\alpha)}.$$

From Theorem 5.2.1 is it easy to calculate time complexity for several different α values; in Figure 5.2 we see a plot of α vs. t_0, where the time complexity is $O(DN^{t_0})$.

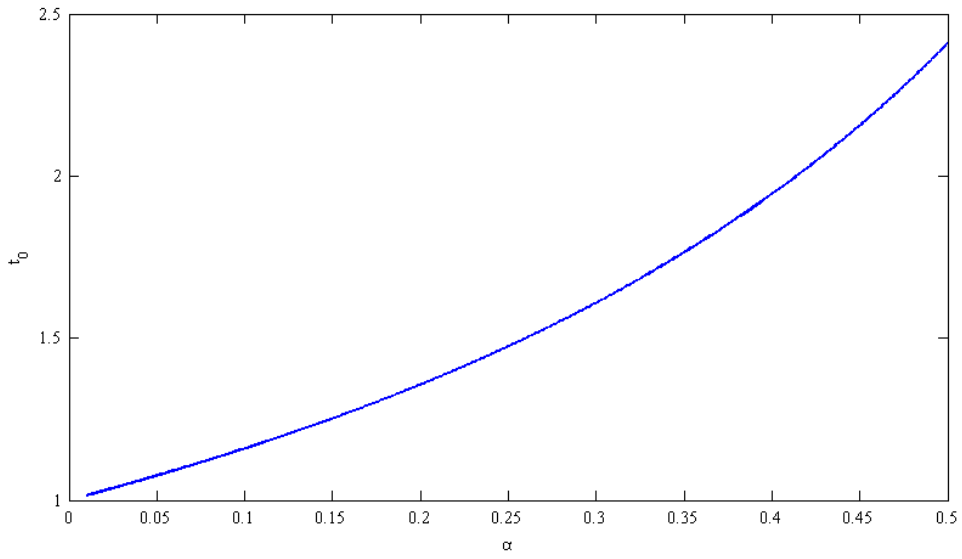


Figure 5.2: *A comparison of time complexity $O(DN^{t_0})$ vs overlap parameter α in the ANN algorithm*

In practice, with an α value equal to 0.1 and β determined by the amount of memory present in the system, we find that the neighborhood are only marginally dissimilar and the runtime goes from hours to minutes for scenes with larger than $100,000$ pixels. We also notice that for the purposes of classification, the results are almost identical. For example in the Indian Pines scene with Laplacian Eigenmaps we see an approximate 400 percent decrease in the graph construction phase with less than 1 percentage point change in classification accuracy. For the methods of Chapter 6 we will use the ANN algorithm for all analysis with $\alpha = 0.1$.

It should be noted that this algorithm would fit well into a parallel implementation because after each Divide step, the resulting two split data sets could be sent off to a separate processor.

Algorithm 5.2.1: ANN

Data: A $N \times D$ data set X, number of nearest neighbors k, overlap percentage α, and brute calculation number β.

Result: A sparse graph G.

if *Size(X)* $\leq \beta$ **then**

    $[G]$=BRUTE(X,k)

else

    $[X_1,X_2]$=DIVIDE(X,α)

    $[G_1]$ = ANN(X_1,k,α,β)

    $[G_2]$ = ANN(X_2,k,α,β)

    $[G]$ = CONQUER(G_1,G_2,k)

end

Chapter 6

Data Fusion

As discussed in Chapter 1, data fusion is a technique that can encompass many aspects of data analysis. For example, In the remote sensing community, data fusion is referred to as many different types of data analysis pan-sharpening or band sharpening, [27, 107, 71], is a technique where two sources of data, one high spectral and low spatial fidelity, and another source of low spectral and high spatial fidelity are combined to approximate a high spectral and high spatial data source. This technique allows the user to improve the spatial fidelity of the high spectral fidelity data source without developing a more expense sensor. Another place where data fusion is used in the remote sensing community is in image registration, see, e.g., [90, 49, 130]. This problem involves finding pixel-to-pixel registration between two data sets, possibly taken with different sensors or temporally separated. The problem of classification, discussed earlier in Chapter 2, can also be viewed as a fusion problem. Here we are fusing a set of partially labeled pixels with a spectral data set describing all the pixels. Yet another example of data fusion in the remote sensing community is the color selection problem, see, e.g., [78, 84, 60]. Here an analyst is required to select, from a multidimensional data source, the three most important channels, or a 3 dimensional feature space representation of the data, to view in a traditional RGB or HSV color space.

In this chapter we shall look at a more traditional type of data fusion with respect to remote sensing data sources: the fusion of two or more disparate types of information into a common product. We shall apply the harmonic analysis concepts developed in the previous chapters to help to accomplish this task. First, we shall look at the problem of spatial-spectral data fusion. Spatial-spectral fusion has been researched over the past decade as a means to improve image classification, anomaly detection, change detection, and target detection results. This fusion method improves the results by making use of the inherent spatial information contained in the scene acquired by a hyperspectral sensor. We will only study fusion in regards to image classification, but we will provide justification for our fusion technique applications to the other problems of remote sensing in Section 6.5. Next, we move on to the fusion of LIDAR and hyperspectral data and then multiple hyperspectral data sources. These are both newer problems in the remote sensing community due to the proliferation of different types of sensors. Next, we explore how to accomplish these fusions without an a priori point registration. Finally, we will look at a novel application of SE analysis to hyperspectral data with limited ground truth as a means to fuse hyperspectral data with expert analysis. The types of fusion we are exploring here, spatial/spectral, LIDAR/HS, multiple HSIs, are just an example of the types of data we can study. The focus of our fusion methodology is building techniques that are not data dependent. By relying on a graph theoretic representation of the data sources, we can readily move our algorithms to different problem domains; the only limitation is developing a measure of similarity for defining the graph. We show an example of developing such a similarity measure, the Minimum

Path Gradient Distance, in reference to LIDAR data.

6.1 Spatial-Spectral Fusion

State of the art methods of analyzing hyperspectral data have primarily focused on studying the spectral information contained in the $N \times D$ data set. However, spatial information is intrinsically included within hyperspectral images. Often, spatial information has largely been ignored in hyperspectral image analysis and dimension reduction algorithms which treat data as belonging to a point cloud.

Recent work has sought to incorporate this spatial information into a coherent product that improves classification results compared to a single sensor by means of data fusion [66]. Current strategies to integrate spatial and spectral information include the use of wavelet packets [16, 15, 68], modified distance metrics for graph construction [104], combining spectral classification with image segmentation results [125], spatially weighted normalized cuts [72], and adding spatial information in spectral band grouping [94]. Other approaches include using iterative information spreading on the data-dependent graphs with SVM to label unknown classes [31, 30, 32], using spatially motivated morphological profiles [64], creating end member sets which are spatially weighted [111], multilevel segmentation [28] or combining segmentation with pixel classification maps [126].

It is within the dimension reduction process that we see opportunities at the graph level, the operator level, and the feature space level, to fuse the multiple data sources in a unique way, which shall improve upon the state of the art. Our

methods, by design, do not rely on pre- and post-processing of the data and use simple classifiers. By developing algorithms which seek to create a new unified representation of the data, our techniques can be easily combined with different classifiers, majority voting, output smoothing, cloud and vegetation masks, all of which can further improve the numerical outputs. By treating the data studied broadly, i.e. without using pre- and post- processing, data specific tools, we believe that our methods can be applied to not just the remote sensing field but to entirely different scientific fields. For example early forms of our ideas were used in the study of gene expressions combined with chromosomal locations [112]. Our goal is to find meaningful ways of including spatial information, which is acquired with no additional cost, with the hyperspectral data in Laplacian Eigenmaps framework. To accomplish this, we studied the effect of introducing spatial information in each phase of the Laplacian Eigenmaps algorithm, separately and simultaneously. Initial work in this domain has been published as [13, 14, 62].

6.1.1 Laplacian Eigenmap Analysis of Hyperspectral Data

Recall that we can think of a $m \times n \times D$ hyperspectral image, Im_{hsi}, as being a collection of $mn = N$, D-dimensional vectors. If we form a matrix X, such that each row corresponds to a hyperspectral pixel (D-dimension vector) then we have a $N \times D$ matrix. The rows of X correspond to the pixels in Im_{hsi}, and the columns correspond to the discretely sampled light spectrum that the hyperspectral sensor recorded.

As a baseline for our analysis, we first look at Spectral Laplacian Eigenmaps which have been previously used to analyze hyperspectral data in [135]. We are defining Spectral Laplacian Eigenmaps to be the standard Laplacian Eigenmaps algorithm where the L_2 distance between two pixels' spectra is used to determine the nearest neighbors and the weights of the kernel matrix. For Section 6.1 we set the number of nearest neighbors, k to be 20 for all three scenes. This number was chosen to ensure that the graph was connected. We also set d, the intrinsic dimension, to 25 for Pavia University and Pavia Centre and 50 for Indian Pines. These dimensions were chosen to roughly represent the material complexity in the scenes and be the number of eigenimages such that any additional eigenimages would not change the classification results. We choose Indian Pines to have a greater intrinsic dimension because the ground truth had classes which were spectrally very close together, i.e., corn and soybeans as different maturity levels.

In Figures 6.1, 6.2, and 6.3 we can see several of the eigenimages that are produced by the Spectral Laplacian Eigenmaps algorithms on three test data sets, Pavia University, Pavia Centre, and Indian Pines. Eigenimages correspond to the eigenvectors produced by the Laplacian Eigenmaps algorithm which are reshaped to form a matrix corresponding to the original dimensions of the hyperspectral image. So for an eigenvector, ϕ_r, which represents the rth dimension of the lower dimensional embedding found by the LE algorithm, we reshape ϕ_r to form an image Im_r as such:

$$\mathrm{Im}_r = \begin{bmatrix} \phi_r(1) & \phi_r(m+1) & \phi_r(2m+1) & \dots & \phi_r((n-1)m+1) \\ \phi_r(2) & \phi_r(m+2) & \phi_r(2m+2) & \dots & \phi_r((n-1)m+2) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \phi_r(m) & \phi_r(2m) & \phi_r(3m) & \dots & \phi_r(nm) \end{bmatrix}.$$

All the eigenimages, Im_r, for $r = 1, \dots, d$, can now be collected to form a $m \times n \times d$ matrix Y. The matrix Y can be thought of as the lower dimensional representation of X. The eigenimages are colored using a linear scale determined by the minimum and maximum value present in the eigenimage; colors only hold meaning in that two pixels which share the same color have been determined to be *close* by the LE algorithm.

The eigenimages that are presented here seem to work well as a feature extraction algorithm, as we know Laplacian Eigenmaps does well in practice. In Figure 6.4 and Table 6.1 we see the classification results for the Spectral Laplacian Eigenmap analysis.

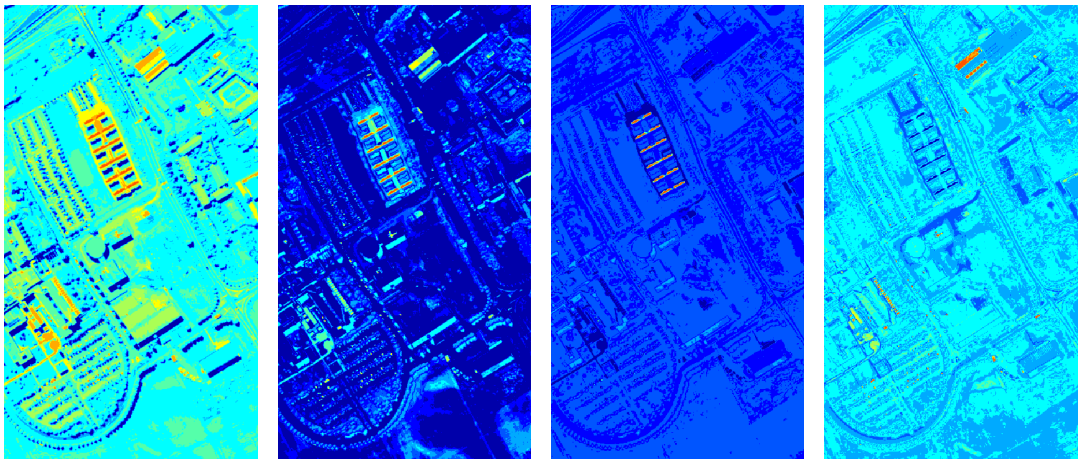


Figure 6.1: *Several eigenimages from Pavia University scene.*

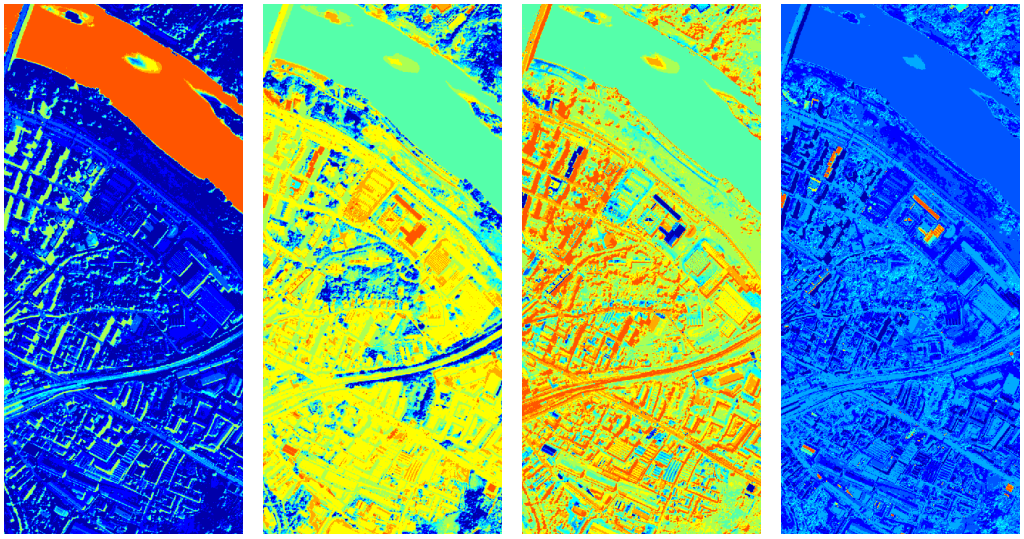


Figure 6.2: *Several eigenimages from Pavia Centre scene.*

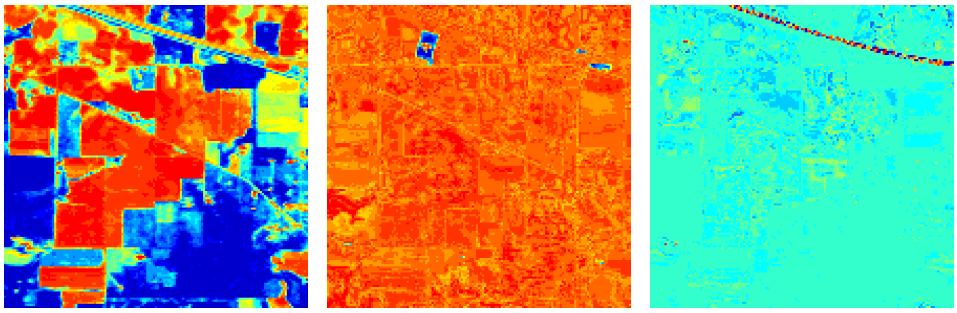
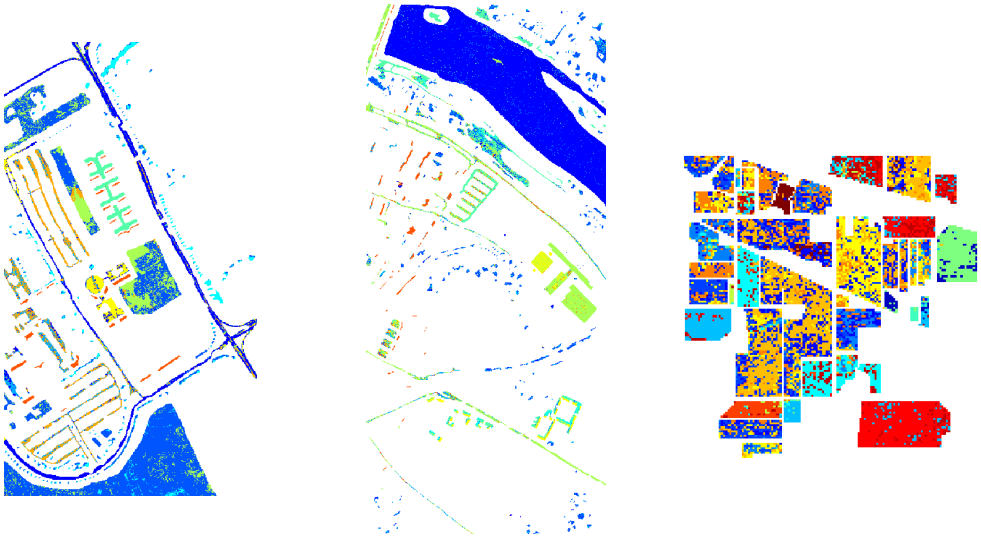


Figure 6.3: *Several eigenimages from Indian Pines scene.*



(a) **(b)** **(c)**

Figure 6.4: *Class maps for (a) Pavia University scene, (b) Pavia Centre scene, and (c) Indian Pines scene, using spectral Laplacian Eigenmaps.*

6.1.2 Spatial Laplacian Eigenmap Analysis of Hyperspectral Data

We first motivate how we can improve on results of the Spectral Laplacian Eigenmaps analysis by looking at the results of running Spatial Laplacian Eigenmaps on X_s, the $N \times 2$ array of purely spatial information, i.e., if a pixel x is located in the ith row and jth column of Im_{hsi}, then let:

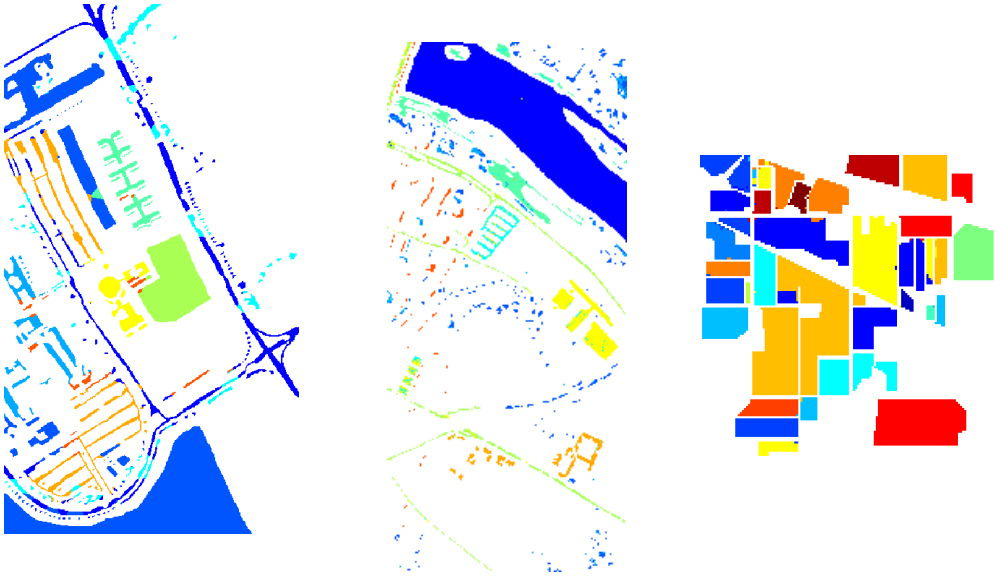
$$X_s(i + (j-1)n, 1) = i \text{ and } X_s(i + (j-1)n, 2) = j. \tag{6.1}$$

Spatial Laplacian Eigenmaps, like Spectral Laplacian Eigenmaps, uses the standard Laplacian Eigenmaps algorithm but with the L_2 distance between pixel locations on the $m \times n$ positive integer grid, where the origin is selected to be the upper left corner of the image, to select nearest neighbors and the weights in the kernel matrix.

The Spatial Laplacian Eigenmaps results provide a means of measuring to what degree the ground truth classes are spatially interspersed. In Figure 6.5 and Table 6.1, we can see the classification results for Spectral Laplacian Eigenmaps and Spatial Laplacian Eigenmap analysis of our three test images.

		Spectral	Spatial
Pavia University	OA	73.26	89.94
	AA	72.15	82.07
	κ	0.6440	0.8593
Pavia Centre	OA	81.16	93.89
	AA	57.25	80.97
	κ	0.6736	0.8948
Indian Pines	OA	60.41	78.78
	AA	55.52	71.61
	κ	0.5491	0.7577

Table 6.1: *Spatial and Spectral Laplacian Eigenmaps Classification Results*



(a) **(b)** **(c)**

Figure 6.5: *Class maps for (a) Pavia University scene, (b) Pavia Centre scene and (c) Indian Pines scene, using Spatial Laplacian Eigenmaps.*

6.1.3 Feature Space Stacking

A first attempt at fusion, and perhaps the most naive, would be in the feature space. Here we can combine eigenvectors obtained by our two initial methods, Spectral and Spatial Laplacian Eigenmaps, by stacking the resulting eigenvectors and then applying a classifier. In Table 6.2 we present the classification results obtained by stacking different ratios of the Spatial Laplacian Eigenmap eigenvectors with the Spectral Laplacian Eigenmap eigenvectors. To arrive at a better comparison with our other methods, we force the total the number of stacked feature vectors to be consistent with the chosen intrinsic dimensions which are 25 for the Pavia scenes and 50 for the Indian Pines scene. In Table 6.2 the columns indicate the ratio of

primary, i.e. the ordering which minimize (3.11), spatial eigenvectors to spectral eigenvectors chosen for this experiment.

		[100p%] Primary Spatial and [100$(1-p)$%] Primary Spectral Vectors							
		0	0.08	0.16	0.4	0.6	0.84	0.92	1.00
Pavia University	OA	73.26	89.91	92.24	95.99	97.48	98.22	96.44	89.44
	AA	72.15	87.39	91.08	95.06	96.84	97.63	95.09	82.07
	κ	0.6440	0.8652	0.8970	0.9465	0.9664	0.9764	0.9525	0.8593
Pavia Centre	OA	81.16	84.46	86.70	92.64	93.77	94.39	94.53	93.89
	AA	57.25	61.16	68.47	82.31	83.05	84.15	84.26	80.97
	κ	0.6736	0.7278	0.7669	0.8722	0.8925	0.9035	0.9059	0.8948
Indian Pines	OA	0.6041	78.78	84.77	93.01	96.46	98.17	98.44	97.95
	AA	55.52	71.61	79.27	92.35	95.84	97.92	97.88	95.83
	κ	0.5491	0.7577	0.8261	0.9204	0.9596	0.9791	0.9822	0.9767

Table 6.2: *Feature space fusion overall classification percentage with varying percentage of principle spatial feature vectors retained.*

The outcome of this analysis is very promising, as the results in Table 6.2 indicate. We can see that after appropriate dimension reduction, choosing a balanced selection of spatial and spectral eigenvectors allows to significantly increase the classification rates as compared to the Spectral Laplacian Eigenmap analysis. Further we see, even including a small amount of spatial information (Column 1 vs Column

2 in Table 6.2) that overall classification scores increase by over 15 percentage points in the Pavia University and Indian Pines scenes. We also see that a purely spatial classification scheme, which should provide meaningless results but relatively high classification scores, does not perform the best in Table 6.2. This confirms that Spectral information is also crucial to the data fusion process. The reason for this improvement and for the fact that such an arbitrage works is the difference between the spatial and spectral class maps, which can be exploited via data integration.

6.1.4 Distance Modification

Inspired by the Spatial Local Linear Embedding (SLLE) algorithm [104], we look to modify it and apply the concept to our LE analysis. The SLLE algorithm adds spatial information to each pixel, by forming a super pixel, containing the immediate spatially defined neighborhood's spectral information. Once the super pixel data set is created the standard LLE algorithm is executed on the modified data set. As mentioned earlier and Section 3.3, under some assumptions, LE and LLE are very similar algorithms, so a straightforward application is possible. We propose here a modification of the algorithm to allow for different neighborhood arrangements.

Let us first define a C-neighborhood:

Definition 6.1.1 (C-Neighborhood)**.** Given a non boundary pixel $p \in X$, called the source pixel, located in a $n \times m$ spatial grid corresponding to the number of rows and columns of X, the pixels, $\{p_i\}_{i=1}^{C}$, which have minimum L_2 spatial distance from

p are the C-Connectivity pixels. This collection of pixels form the C-Neighborhood. We can break ties which occur by either choosing randomly or choosing the smallest spectral distance to the source pixel.

Remark 6.1.2. We will here consider two general cases of C-neighborhoods, $C = 4, 8$. When $C = 4$, the neighborhood will be the pixels immediately to the north, south, east, and west. When $C = 8$, the neighborhood will be the pixels immediately to the north, south, east, west, north-east, north-west, south-east, and south-west.

Using the C-Neighborhood it is possible to construct a super pixel for each non boundary pixel in the scene by concatenating the source pixel with is C-Neighborhood pixels. Now, for our standard D dimensional data set, we have pixels which are $(C + 1)D$ dimensional. When distances between super pixels are measured, the distances amongst the local patches are also measured. This forces pixels that have similar local neighborhoods to be close in the final embedding, not just similar single pixels. Boundary pixels can be addressed by either using a reflexive boundary or ignoring them all together as the border of an image contains a small fraction of the total pixels.

The method discussed in [104] does not however give consideration to the arrangement of the local neighborhoods. For example, two pixels separated in the image could have very similar neighborhoods up to a rotation or permutation. This would be missed by the SLLE algorithm as the super pixel does not take this rotation or permutation into account. For the special case of $4-$ and $8-$Neighborhoods we will consider the 4 and 8 rotations, respectively, for the neighborhood. The

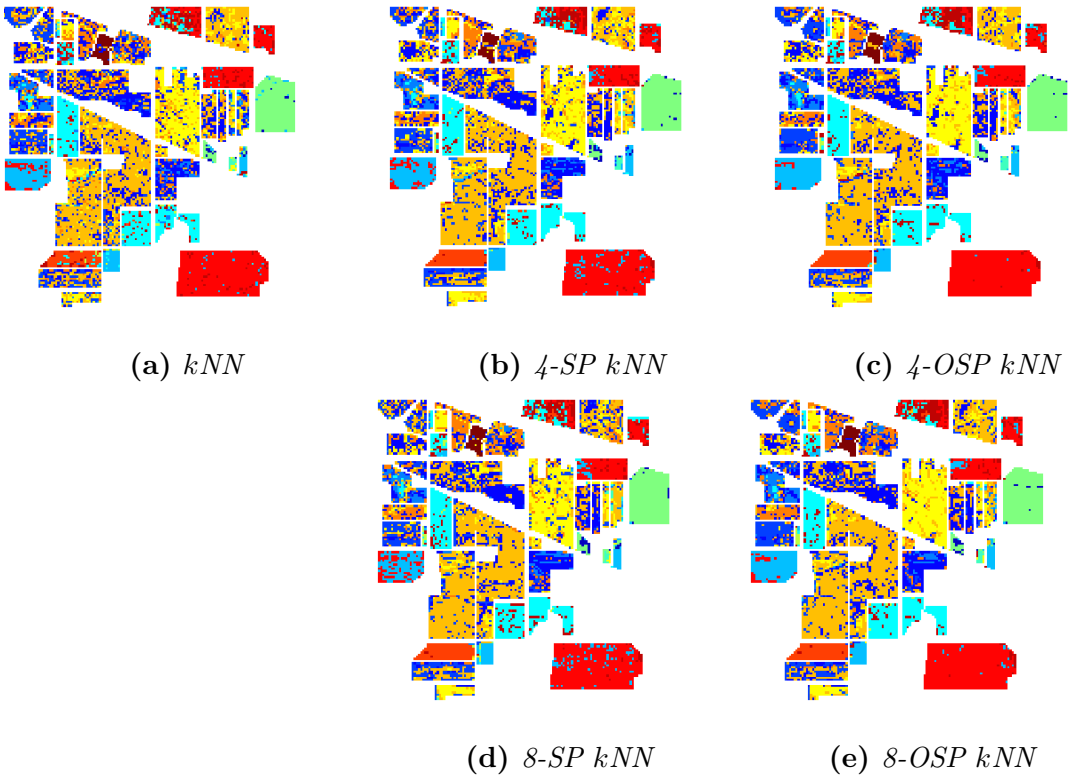
classification results for the Indian Pines and Pavia Univeristy data set, with normal kNN, the super pixel kNN from [104], and our optimized super pixel kNN can be found in Tables 6.3, 6.4, 6.5, and6.6. The corresponding class maps can be found in Figures 6.6 and 6.7. We see that Super Pixel kNN improves upon the standard kNN for LE analysis just as was show for LLE. We also see that optimizing the arrangement of the super pixels, in the Optimized Super Pixel kNN, improved the classification results as compared with the naive approach. When moving from 4-Connectivity to 8-Connectivity we notice that the optimized arrangement showed classification improvement where the naive approach had a decrease in classification performance. This shows the superiority of the optimized method as increasing the spatial patch should improve upon results.

	kNN	Super Pixel kNN	Optimized Super Pixel kNN
OA	60.41	62.23	66.58
CA	55.52	60.55	64.49
κ	0.5491	0.5702	0.6189

Table 6.3: *Classification results for 4-Connectivity super pixels kNN construction for LE analysis of Indian Pines.*

	kNN	Super Pixel kNN	Optimized Super Pixel kNN
OA	60.41	61.30	66.91
CA	55.52	58.06	65.58
κ	0.5491	0.5595	0.6236

Table 6.4: *Classification results for 8-Connectivity super pixels kNN construction for LE analysis of Indian Pines.*



(a) *kNN* **(b)** *4-SP kNN* **(c)** *4-OSP kNN*

(d) *8-SP kNN* **(e)** *8-OSP kNN*

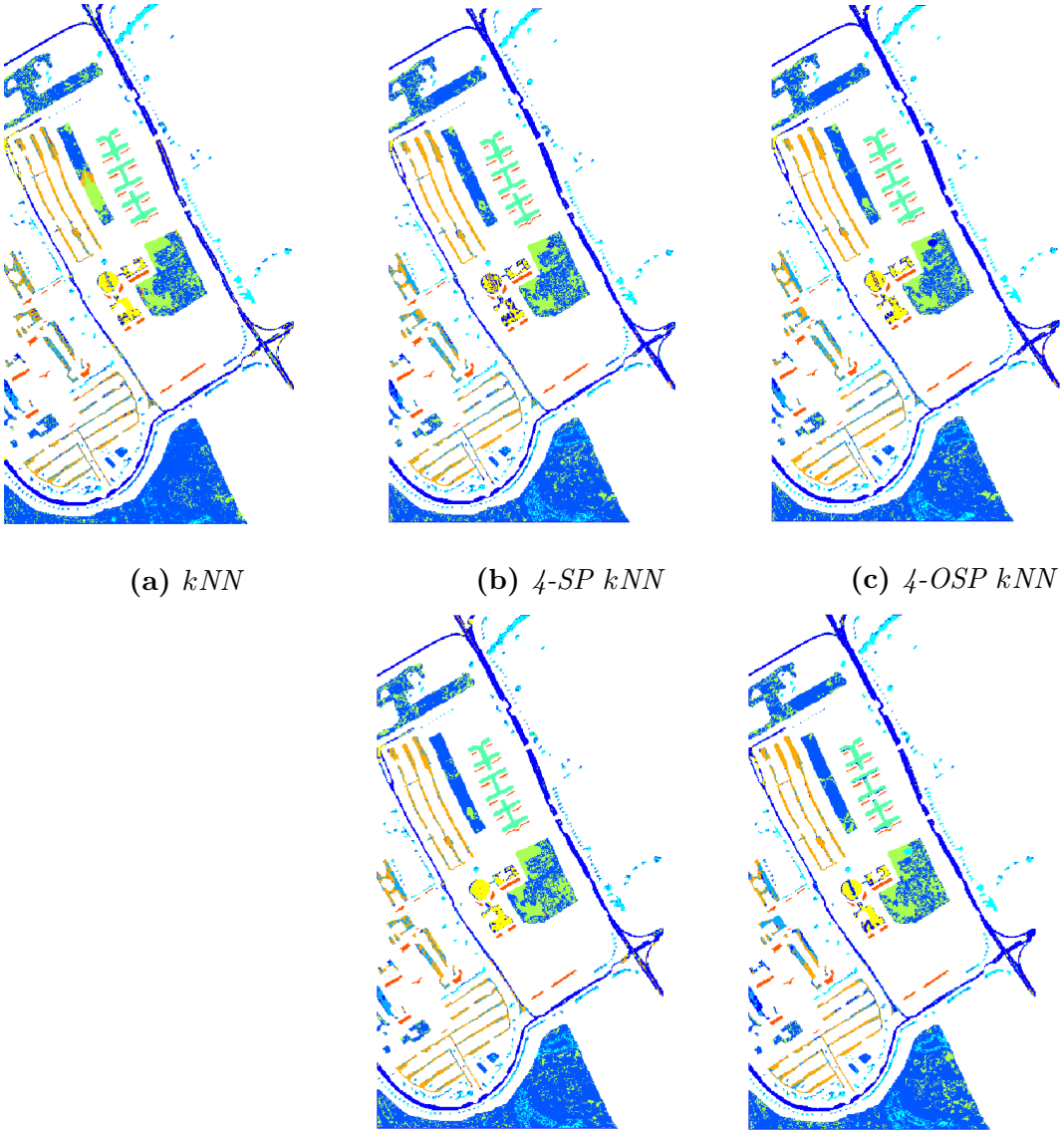
Figure 6.6: *Class maps for Indian Pines after performing LE analysis on the graphs constructed from (a)standard kNN, (b) 4-Connectivity Super Pixels, (c) 4-Connectivity Optimized Super Pixels, (d) 8-Connectivity Super Pixels, and (e) 8-Connectivity Optimized Super Pixels.*

	kNN	Super Pixel kNN	Optimized Super Pixel kNN
OA	73.26	74.45	76.45
CA	72.15	72.73	75.12
κ	0.6440	0.6711	0.6872

Table 6.5: *Classification results for 4-Connectivity super pixels kNN construction for LE analysis of Pavia University.*

	kNN	Super Pixel kNN	Optimized Super Pixel kNN
OA	73.26	76.36	77.81
CA	72.15	75.89	77.67
κ	0.6440	0.6883	0.7074

Table 6.6: *Classification results for 8-Connectivity super pixels kNN construction for LE analysis of Pavia University.*



(a) *kNN* **(b)** *4-SP kNN* **(c)** *4-OSP kNN*

(d) *8-SP kNN* **(e)** *8-OSP kNN*

Figure 6.7: *Class maps for Pavia University after performing LE analysis on the graphs constructed from (a)standard kNN, (b) 4-Connectivity Super Pixels, (c) 4-Connectivity Optimized Super Pixels, (d) 8-Connectivity Super Pixels, and (e) 8-Connectivity Optimized Super Pixels.*

Remark 6.1.3. We would like to allow for any of the possible permutation of the

local neighborhood and then find the minimum distance but this is combinatorially hard; for the case of 8-connectivity there are 8!=40,320 arrangements to check. This problem is akin to the assignment problem in combinatorial optimization which has a $O(C^3)$ solution in the form of the Hungarian Algorithm [83]. For our setup this would require $O(C^3N^2)$ operations.

6.1.5 Graph Based

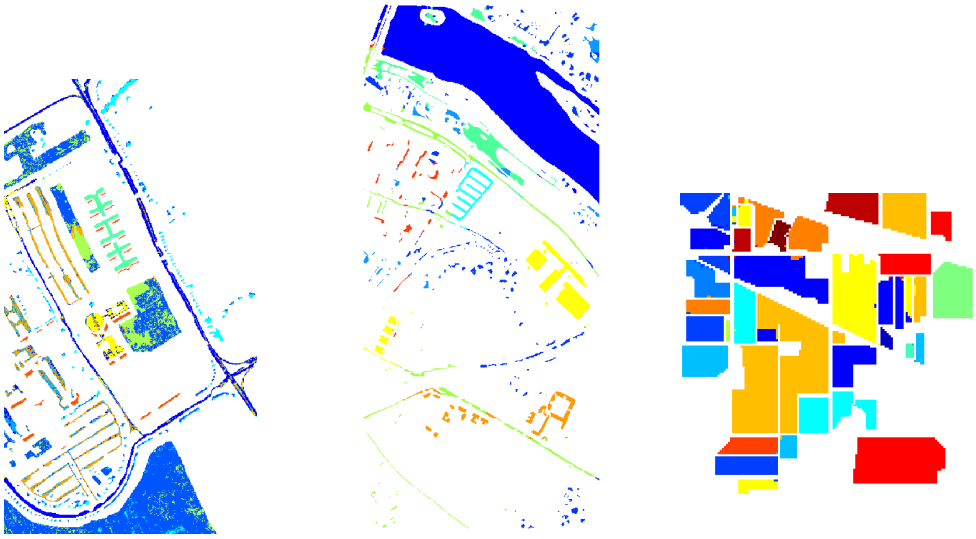
Another avenue to explore for data fusion is combining spatial and spectral information in the graph construction phase of the LE algorithm. Let G_f be the kNN graph computed using the spectral metric and let W_f be the weight matrix constructed for a given kNN graph using weights determined by 3.10. Let G_s be the kNN graph constructed using the spatial metric:

$$\|x_i - x_j\|_s = \|s(i) - s(j)\|_2,$$

and let W_s be the weight matrix obtained by replacing the spectral weights with their spatial analogues. In this spatial-spectral fusion attempt, a global structure is first defined by the initial construction of G_f using spectral information. Later, by replacing the weights in the kernel matrix with spatial distances a priority in the embedding will be given toward preserving distances, between spatially close pixels. However, since the connections in G_f were spectrally defined, materials with similar spectra will still be linked in the embedding. We show class maps for our test data sets in Figure 6.8 and classification results in Table 6.7. .

		Mixed
Pavia University	OA	73.34
	AA	72/24
	κ	0.6450
Pavia Centre	OA	95.58
	AA	87.64
	κ	0.9240
Indian Pines	OA	60.49
	AA	55.94
	κ	0.5502

Table 6.7: *Mixing spectral neighborhoods with spatial weights*



(a) **(b)** **(c)**

Figure 6.8: *(a) Pavia University scene, (b) Pavia Centre (c) Indian Pines Scene using spectral neighborhood construction with spatial kernel.*

6.1.6 Operator Based

Having already explored how to fuse data in the graph construction phase of the Laplacian Eigenmaps, we move onto looking at how to fuse in the operator phase of the LE algorithm. In this section, we fix $G = G_f$. We choose to fix the construction G based on spectral information as to preserve a global perspective to the embedding. Otherwise, a very local and patchy embedding may result. Spatial information is introduced by modifying the Laplacian operator, L, in three different ways. Let L_{ff} be the operator using spectral information for the graph construction and graph weights and, let L_{fs} be the operator using spectral information for the graph construction and spatial information for the graph weights. We define:

- $L_1(\sigma, \eta) = L_{ff}(\sigma) \cdot L_{fs}(\eta)$, where $\cdot$ denotes entry-wise multiplication,
- $L_2(\sigma, \eta) = L_{ff}(\sigma) + L_{fs}(\eta)$,
- $L_3(\sigma, \eta) = G \cdot (L_{ff}(\sigma) \times L_{fs}(\eta))$, i.e., we take the usual matrix product of the two Laplacians and then overlay the sparsity structure determined by the adjacency matrix.

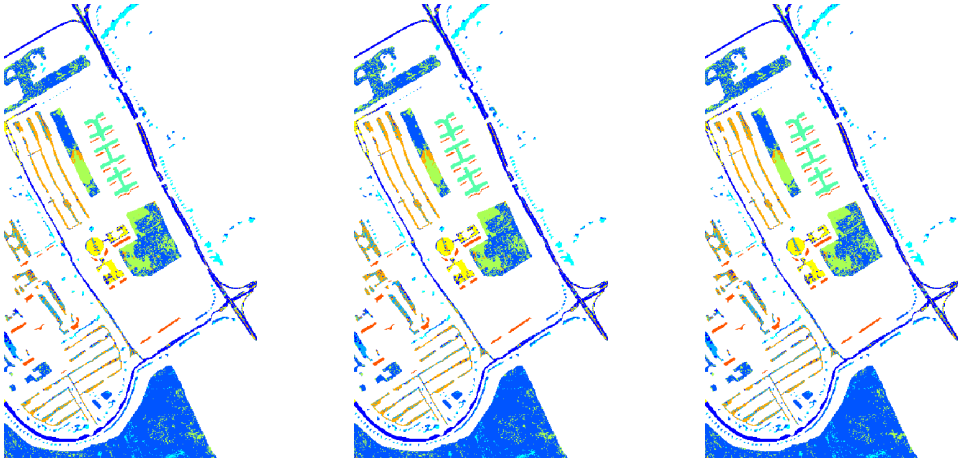
In this fusion operator based approach to adding spatial information, three separate methodologies are considered based on how the fused diffusion weight matrix is defined. $L_1(\sigma, \eta)$ is perhaps the simplest and is akin to defining a new distance metric which weights spectral and spatial information differently for the diffusion weights matrix. $L_2(\sigma, \eta)$, by adding the two diffusion weight matrices separately allows for pixels are both close spectrally and spatially to dramatically dominate the imbedding while allowing pixels which are only close in one regard to have approximately equal importance in the imbedding. $L_2(\sigma, \eta)$ can viewed as a general for of the Schrödinger operator, where the L_{fs} operator acts like a potential. $L_3(\sigma, \eta)$ seeks to take the fusion deeper by considering weights derived from common neighborhoods amongst pixels. By further requiring the original sparsity structure be preserved keeps the problem computationally on scale with the fusion operator approaches.

Across all three data sets (Pavia University, Pavia Centre, and Indian Pines) we found that a value 0.2 for η and 0.8 for σ produced the best results, σ was set to 0.8. We performed a simple grid optimization search, varying σ and η between 0.1 and 1.0 in intervals of width 0.1 and choose the pair which gave the heights

classification accuracy; we did however notice that there was not a large amount of variation caused by the parameter selection. In Figures 6.9, 6.10 and 6.11 we provide the class maps and in Table 6.8 the classification accuracy for for the three test data sets.

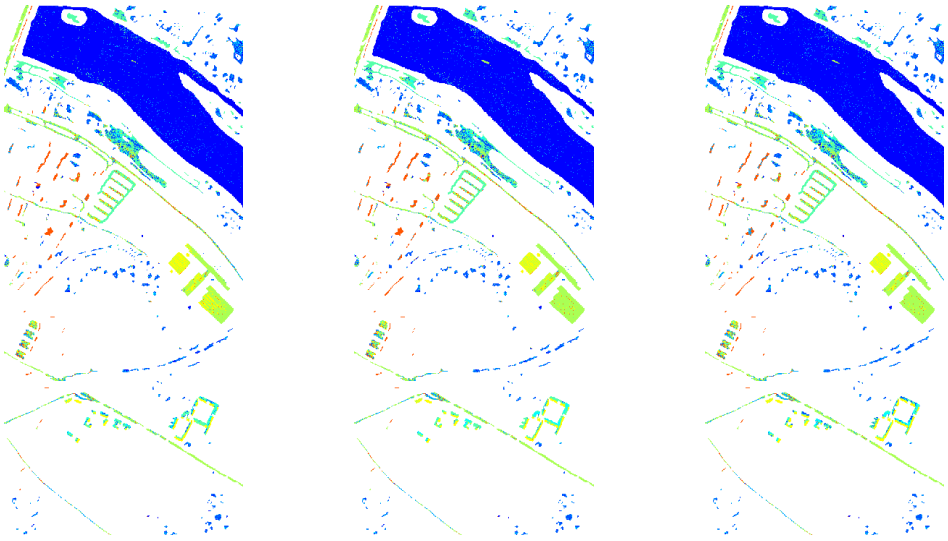
		L_1	L_2	L_3
Pavia University	OA	73.92	73.90	73.92
	AA	72.53	72.28	72.59
	κ	0.6482	0.6525	0.6528
Pavia Centre	OA	95.84	95.66	95.80
	AA	88.40	87.88	88.20
	κ	0.9285	0.9524	0.9279
Indian Pines	OA	61.30	52.62	64.73
	AA	56.56	47.23	61.82
	κ	0.5593	0.4587	0.5983

Table 6.8: *Operator Fusion Classification Results*



(a) $L_1(\sigma, \eta)$ **(b)** $L_2(\sigma, \eta)$ **(c)** $L_3(\sigma, \eta)$

Figure 6.9: *Pavia University class maps using spectral neighborhood construction and Fusion Operators kernel matrix construction.*



(a) $L_1(\sigma, \eta)$ **(b)** $L_2(\sigma, \eta)$ **(c)** $L_3(\sigma, \eta)$

Figure 6.10: *Pavia Centre class maps using spectral neighborhood construction and Fusion Operators kernel matrix construction.*

(a) $L_1(\sigma, \eta)$ **(b)** $L_2(\sigma, \eta)$ **(c)** $L_3(\sigma, \eta)$

Figure 6.11: *Indian Pines class maps using spectral neighborhood construction and Fusion Operators kernel matrix construction.*

6.1.7 Combining Graph and Operator Fusion

Another way spatial and spectral information can be combined is to adjust the norm used to determine the knn-graph and the weight matrix. In this manner we are claiming that by carefully adjusting the norms used to construct the graph we can inject important spatial information into the optimization phase and thus favor solutions (eigenimages) that are spatially and spectral motivated. For $\gamma > 0$, we define

$$\|x_i - x_j\|_\gamma = \sqrt{\|x_i - x_j\|_f^2 + \gamma \|x_i - x_j\|_s^2}.$$

Let L_γ be the Laplacian operator defined using the kNN graph and weight matrix determined by $\|\cdot\|_\gamma$.

The parameter γ allows the user to control how much spatial information to include in the analysis. Our goal was to choose γ so that the contribution of spatial information to $\|\cdot\|_\gamma$ reflects the relative importance of spatial information to spectral

information inherent in the image. By choosing such a γ we would guarantee that two pixels in the image would be considered close with respect to $\|\cdot\|_\gamma$ if and only if they are close spectrally and "close enough" spatially. Motivated thusly, let γ be determined as follows.

Let $\{N_i\}_{i=1}^N$ be the set of indices for the spectral kNN of the pixel x_i. Define γ_i to be the ratio of the spectral and spatial spread of this neighborhood:

$$\gamma_i = \frac{\sum_{j\in N_i} \|x_i - x_j\|_f^2}{\sum_{j\in N_i} \|x_i - x_j\|_s^2}.$$

Define γ to be the global average of these local weights:

$$\gamma = \frac{1}{N}\sum_{i=1}^{N} \gamma_i.$$

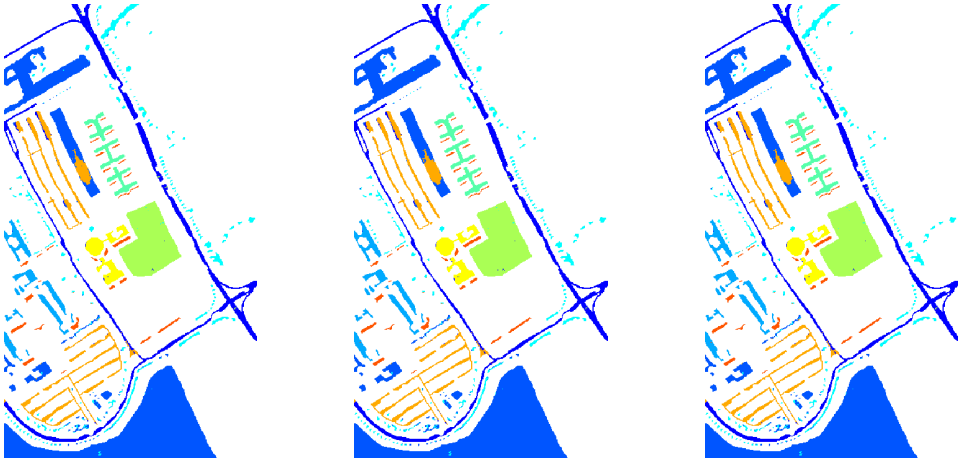
In Figures 6.12, 6.13, and 6.14 we present the class maps produced using this metric with the fusion operators from the previous section. In Table 6.9 we also present the classification accuracies for this method.

To capture the essence of the results presented in this section we refer again to Table 6.9, which provides the overall and average classification rates for the three examples of hyperspectral imagery we have chosen to analyze. Comparing to the most recent results in the field of spatial-spectral integration for hyperspectral imagery, we clearly show that our results are in line with those presented recently in literature, or better, see e.g., in [66]. Moreover, we showed that across the range of various methods presented in this section, the standard deviation is negligible, and as such all methods perform equally well. This is due to the fact that varying the graph constructions with a fixed method for computing the Laplacian, or the varying the construction of the Laplacian on a fixed graph, amounts to similar results, if done

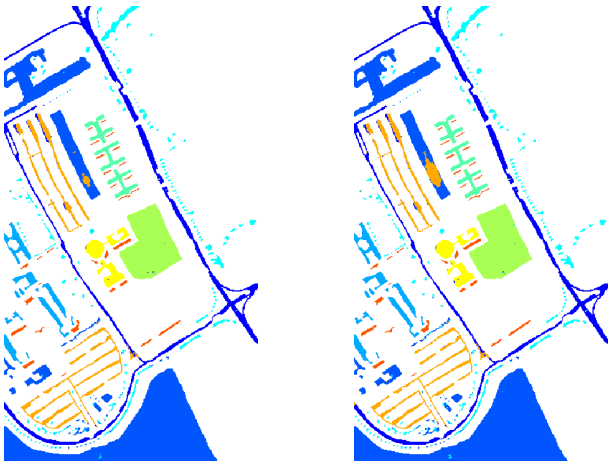
correctly. At the same time, we emphasize that we have introduced a completely novel approach to spatial-spectral fusion, which possibly can be combined with the state of the art classification techniques such as those in [131, 22, 127] to provide additional improvements.

		L_f	L_1	L_2	L_3	L_γ
Pavia University	OA	97.15	96.99	97.03	**98.15**	97.14
	AA	97.47	97.39	97.39	**97.81**	97.46
	κ	0.9623	0.9603	0.9607	**0.9754**	0.9622
Pavia Centre	OA	95.57	95.81	95.68	95.77	**97.28**
	AA	87.62	88.26	87.94	88.11	**92.91**
	κ	0.9239	0.9280	0.9258	0.9273	**0.9532**
Indian Pines	OA	98.81	**98.87**	98.82	87.95	98.81
	AA	98.52	**98.56**	98.53	78.89	98.52
	κ	0.9864	**0.9872**	0.9865	0.8625	0.9864

Table 6.9: *Fusion Metrics Classification Results*

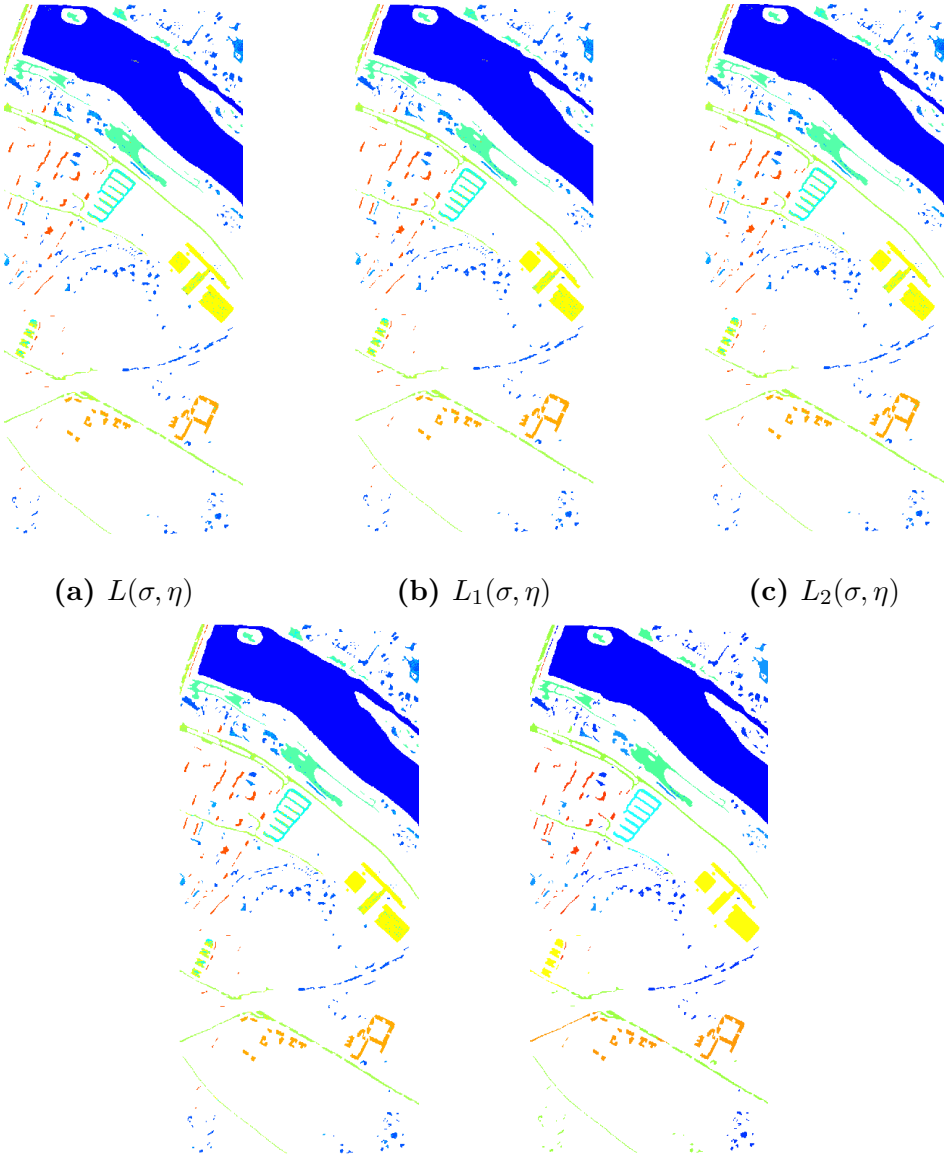


(a) $L(\sigma, \eta)$ **(b)** $L_1(\sigma, \eta)$ **(c)** $L_2(\sigma, \eta)$



(d) $L_3(\sigma, \eta)$ **(e)** $L_\gamma(\sigma, \eta)$

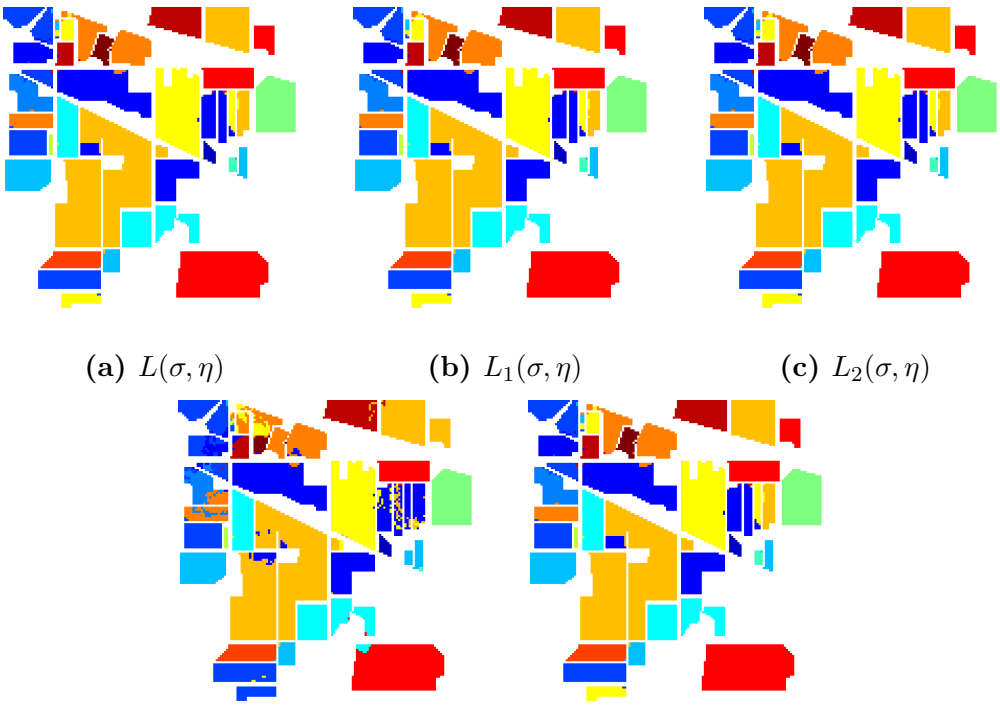
Figure 6.12: *Pavia University class maps using fusion metric neighborhood construction.*



(a) $L(\sigma, \eta)$ **(b)** $L_1(\sigma, \eta)$ **(c)** $L_2(\sigma, \eta)$

(d) $L_3(\sigma, \eta)$ **(e)** $L_\gamma(\sigma, \eta)$

Figure 6.13: *Pavia Centre class maps using fusion metric neighborhood construction.*



(a) $L(\sigma, \eta)$ **(b)** $L_1(\sigma, \eta)$ **(c)** $L_2(\sigma, \eta)$

(d) $L_3(\sigma, \eta)$ **(e)** $L_\gamma(\sigma, \eta)$

Figure 6.14: *Pavia University class maps using fusion metric neighborhood construction.*

6.2 Fusing Hyperspectral and LIDAR Data

Recall from Chapter 2 that LIDAR data represents the height of an object relative to the earths' surface. Along withe height information provided by the LIDAR we will also infer spatial coordinates upon LIDAR data in the same fashion as we did in 6.1 to create a 3 dimensional set of coordinates for each pixel. In this section, we will assume that we two data sets, hyperspectral and LIDAR, that have been co-registered such that the (i, j)th pixel from one data set matches the (i, j)th pixel from the other. Our goal here, like with the spatial-spectral fusion, is to use the

spatial information inherent in the LIDAR data to improve the classification results of the hyperspectral data. Previous approaches to this problem have attempted to fuse the data using sparse modeling [33], decision trees [26], diffusion bases [102], Shape and Spectral Integration (SSI) [121], and Support Vector Machines (SVMs) [50, 82]. To solve this difficult problem we rely on representation theory of data dependent operators instead. We build on the previous fusion methods in this Chapter by introducing LIDAR and spatial information, not only spatial information, to the hyperspectral data. We also develop new concepts in feature space rotation between distinct sensor outputs. We will first develop a distance measure for LIDAR data which will help us build a graph representation of the data. Next, we will introduce a new type of fusion which we will call Feature Space Rotation fusion based on the ideas of [41]. Finally, we will provide numeric results for two hyperspectral and LIDAR data sets.

6.2.1 Minimum Path Gradient Distance Algorithm

We introduce here the Minimum Path Gradient Distance (MPGD) as a means to measure the three dimensional distance between pixels in such a manner that local structure is preserved. In theory, we want to consider all possible paths between two pixels in an image and find the one which minimizes the numerical gradient along the path. This provides a valuable metric by which pixels belonging to a common class and close in MPGD measure, can be differentiated from pixels from different classes which are spatially close, or from pixels of the same class, which

are spatially separated. Moreover, MPGD provides a feasible way to fuse spatial and LIDAR data with hyperspectral data. As minimizing over all possible paths is combinatorially hard, we rely on an approximate solution using spatial clustering and Dijkstra's algorithm. Given a $m \times n$ LIDAR image $\mathrm{Im_L}$:

1. Let $m_L = \min_{i,j}\{\mathrm{Im_L}(i,j)\}$ and $M_L = \max_{i,j}\{\mathrm{Im_L}(i,j)\}$. Define a set of N_L levels between m_L and M_L, i.e. $L = \{(M_L - m_L)k + m_L\}_{k=0}^{N_L-1}$.

2. Label pixels of $\mathrm{Im_L}$ as belonging to the same class if they are spatially connected (using 8-Connectivity) and their values belong to the same interval $L_i = [L(i), L(i+1)]$ for $i = 0, \ldots, N_L$ (in essence, a form of quantization). Denote clusters $\mathcal{C} = \{c_i\}$ assigned by the function C with corresponding values $R = \{r_i\}$ and $\mathrm{Im_L^{N_L}}$ as the approximate or quantized image.

3. Let G_L be the graph with vertices indexed by $\mathcal{C}$ found in the previous step:

$$G_L(i,j) = \begin{cases} (r_i - r_j)^2, & \text{if } c_i \text{ and } c_j \text{ are connected spatially} \\ 0, & \text{otherwise} \end{cases}$$

4. To find MPGD between two pixels, $D_M(x_i, x_j) \approx D_M(C(x_i), C(x_j))$, use Dijkstra's algorithm on the graph G_L.

Dijkstra's algorithm, [57], reduces the complexity of finding the shortest path between two vertices in a graph from the brute force case of $O(N^2)$, to $O(N \log N)$. It accomplishes this by spreading out from the source vertex via the edge connections of the graph and updating the distance from the source vertex to all other vertices while traveling. By only visiting each vertex once, and always traveling in the direction of

the closest vertex, which has not been visited yet, from the source vertex, Dijkstra's algorithm is able to drastically reduce the computational burden. Using Dijkstra's algorithm thus allows us to efficiently calculate the MPGD. As the number of levels, N_L, increases the approximation, $\mathrm{Im}_{\mathrm{L}}^{N_L}$, closer to that of the actual LIDAR image Im_{L}. The parameter N_L thus allows us to balance computational performance and accuracy.

We have developed the MPGD because we need a robust distance for LIDAR which can easily return a distance between two requested pixels in LIDAR space. This distance will be especially powerful in discerning dissimilarity in urban scenes. For example, given two buildings which are close together spatially, have approximately the same spectral properties, and are same hight using a traditional LIDAR distance measure would group the pixels belonging to the two buildings together in the LIDAR graph representation. Also, since the buildings have approximately the same spectral properties the hyperspectral derived graph would connect the two buildings. The MPGD, on the other hand, would only connect the pixels belonging to each building independently in the graph representation.

The MPGD, unlike morphological profiles, is shape independent in that it can learn shapes of contiguous objects since their gradient is not likely to change dramatically.

Remark 6.2.1. We see three possible extensions to the MPGD. The first of which is to weigh the gradient along the path by the length of the path. This would constrain the solution to be even more locally focused. The second would be to

also consider hyperspectral information during the MPGD calculation. Here we would be accomplishing our fusion solely during the calculation of the MPGD. This could be accomplished easily by calculating $D_M(C(x_i), C(x_j)) + d(x_i, x_j)$, where $d(x_i, x_j)$ is the L_2 distance between the pixels' spectra. Third we augment the hyperspectral data set by adding additional channels derived by the MPGD. In this scheme we would choose pixels at random or by some deterministic algorithm, see the landmarking discussion in Chapter 8, and calculate the MPGD to all other pixels. This $N \times 1$ vector of distances would now be added as another channel to the hyperspectral data set with an appropriate weight. Laplacian Eigenmap analysis, or any of the developed fusion methods, can now be applied to the augmented hyperspectral data set.

6.2.2 Feature Space Rotation

In [41], Coifman and Hirn, develop a method which we will call here, Feature Space Rotation. In their paper they develop their concept for Diffusion Maps but we will show that LE can be used with it as well. The basic concept of the method it to learn a rotation from one feature space to another given some amount of pixel registration between the two embeddings. We will present this method with the fusing of two data sets, but it can be easily extended to fuse any number of data sets.

Let us first assume that we are given two data sets, X_1 and X_2. In our applications, X_1 will be a hyperspectral data set and X_2 will be a LIDAR image in

Sections 6.2.3 and 6.2.4 or another hyperspectral data set in 6.3.1 and 6.3.2.

Let us also assume that we preformed an LE analysis of both X_1 and X_2; thus resulting in the feature space representation of Y_1 and Y_2 for X_1 and X_2 resultant from mappings Φ_1 and Φ_2. Let the dimension of the feature space, i.e. the number of retained eigenvectors from the LE minimization phase, be d_1 and d_2. So we have

$$\Phi_1 : X_1 \mapsto Y_1 \text{ and } \Phi_2 : X_2 \mapsto Y_2,$$

and

$$\Phi_1(x) = [\phi_1^{(1)}(x), \ldots, \phi_1^{(d_1)}(x)] \text{ and } \Phi_2(x) = [\phi_2^{(1)}(x), \ldots, \phi_2^{(d_2)}(x)].$$

We now embed Y_1 and Y_2 into one joint representation space. Without this, each embedding would have to be considered independently. For this reason, we define a rotation operator $\mathcal{O}_{2\to1} : Y_2 \to Y_1$ by

$$\mathcal{O}_{2\to1}x = \left(\sum_{j=1}^{d_2} x_j \langle \phi_1^{(i)}, \phi_2^{(j)} \rangle \right)_{i=1}^{d_1}. \tag{6.2}$$

The final step is to fuse the data expressed in Y_1 and Y_2 into a common representation. For any $x \in X_1$, we concatenate to form:

$$\tilde{x} = [y_1; \mathcal{O}_{2\to1}y_2],$$

We have now defined a fused feature space where any type of classification algorithm can therefore be applied. In Figure 6.20 we see a diagram summarizing the feature space rotation fusion method applied to two data sets, X_1 and X_2.

In the next two subsections, 6.2.3 and 6.2.4, we use the MPGD in conjunction with the fusion methodologies developed earlier and the feature space rotation fusion

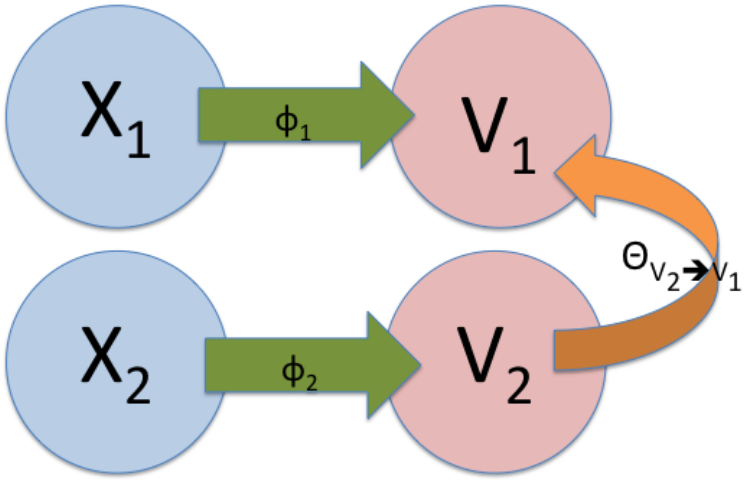


Figure 6.15: *Fusion of X_1 and X_2 via feature space rotation.*

to study the Houston and Gulfport scenes. In Section 6.2.3, we again solve the image classification problem for the Houston data set. In Section 6.2.4, we will introduce endmember extraction then produce the endmembers and abundance maps on the fused Gulfport data sets instead of performing image classification, as there is no available ground truth.

6.2.3 Image Classification

Recall the Houston data set discussed in Section 2.2.1; this data set contains a LIDAR and hyperspectral image with corresponding ground truth. To establish a baseline we used standard LE analysis on the HSI and HSI stacked with the LIDAR image, i.e., [HSI; (γ LIDAR)], where γ is an optimized multiplier for the LIDAR band chosen to give the best classification results.

The results for Overall Accuracy (OA) and Average Accuracy (AA) for the operator fusion, graph fusion, and feature space rotation fusion are shown in Table

6.10. Our best class map, by percent classification, was found using the feature space rotation fusion, can be seen in Figure 6.16. We also include the graph fusion classification map as it gave comparable results, see Figure 6.17. As seen in Table 6.10, our methods all outperformed the naive approach of stacking the heterogeneous data. It should be noted that the presence of a cloud in the scene affected the classification in the surrounding area; this is a situation where the application of a pre-processing algorithm, i.e. a cloud removal mask, would have improved our results.

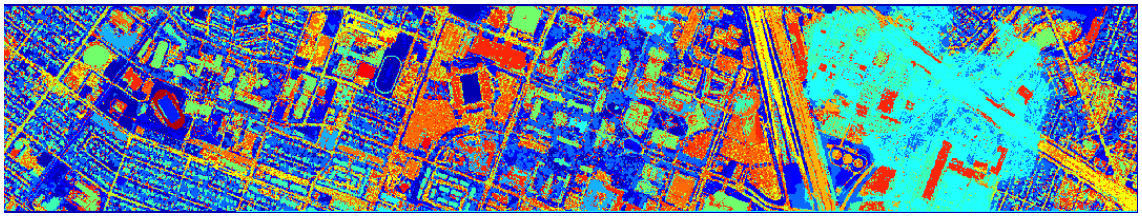


Figure 6.16: *Class map obtained using Feature Space Fusion*

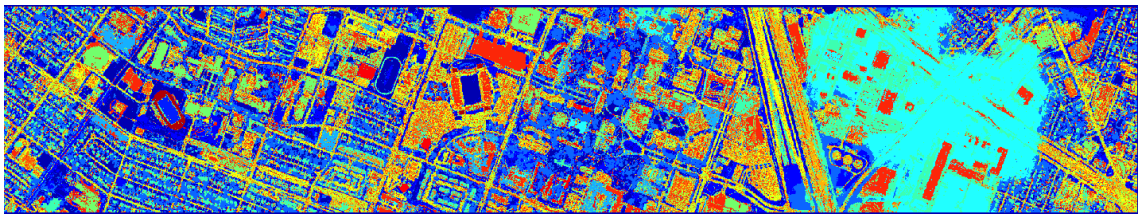


Figure 6.17: *Class map obtained using Graph Fusion*

6.2.4 Endmember Extraction

It is not always the case that data sets come with available ground truth to evaluate an algorithms effectiveness. Without ground truth, a unsupervised approach must be undertaken to perform classification on the image. For this exper-

Method	% OA	% AA
LE HSI	74.237	74.252
LE [HSI; (γ LIDAR)]	80.345	80.303
Graph Fusion	85.270	85.772
$L_1(\sigma, \eta)$	81.284	81.285
$L_2(\sigma, \eta)$	83.591	83.590
$L_3(\sigma, \eta)$	84.616	84.719
Feature Space Rotation	86.374	86.437

Table 6.10: *Summary of Classification Results for Houston Data Set*

iment we shall make use of endmember extraction algorithms to evaluate visually the effectiveness of our data fusion algorithms.

An endmember can be thought of as a representative or pure sample from an image. Using the linear mixture model [79], every pixel $x_i \in X$, can be represented as a linear collection of s end members, e_i, with coefficients, $c_{i,j}$, and noise vector N_{x_i},

$$x_i = \sum_{j=1}^{s} c_{i,j} e_j + N_{x_i}.$$

For simplicity it is usually assumed that the coefficients are non-negative, $c_{i,j} \geq 0$ for $i = 1, \ldots, N$ and $j = 1, \ldots, s$, and sum to 1, $\sum_{j=1}^{s} c_{i,j} = 1$ for $i = 1, \ldots, N$. The

coefficients, $c_{i,j}$, are chosen to minimize the reconstruction error,

$$c_{i,\cdot} = \underset{c_{i,j}\geq 0, \sum_{j=1}^{s} c_{i,j}=1}{\arg\min} \left\| x_i - \sum_{j=1}^{s} c_{i,j} e_j \right\|_2 .$$

A common extension is to add sparsity [34] via a penalty term τ_i,

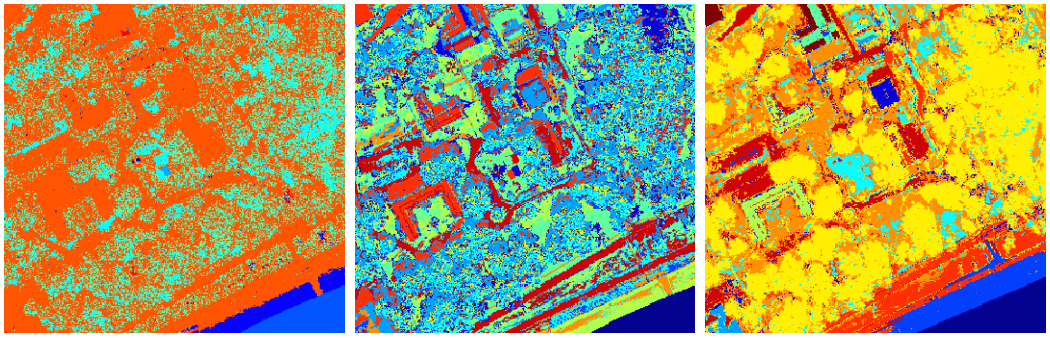
$$c_{i,\cdot} = \underset{c_{i,j}\geq 0, \sum_{j=1}^{s} c_{i,j}=1}{\arg\min} \left\| x_i - \sum_{j=1}^{s} c_{i,j} e_j \right\|_2 + \tau_i \left\| c_{i,\cdot} \right\|_1 .$$

The process of finding the coefficients is called spectral unmixing [79], as we are finding each pixels' abundance contribution from the set of endmembers. Once the set of endmembers and coefficients are found, an abundance map can be produced for each endmember indicating the percentage of each pixel that corresponds to that endmember. It also possible to classify the data in an unsupervised format (similar to clustering) based on the coefficients, by labeling each pixel belonging to the endmember class of greatest contribution.

For this work we shall make use of the Vertex Component Analysis (VCA) algorithm [105]. The VCA algorithm extracts a set $E = \{e_1, \ldots, e_s\}$, of endmembers from a data set by finding the s vertices of a simplex that encompasses the point cloud. Other algorithms exist, for example, N-FINDR [137], Pixel Purity Index (PPI) [25] and Support Vector Data Description (SVDD) [128], but we choose to use the VCA algorithm for its presence in the literature and simple implementation.

For this experiment we use the MUUFL Gulfport data set. We choose to use $k = 30, \sigma = .5$, and $d = 30$ for both the hyperspectral, with L_2 distance, and LIDAR, with MPGD, datasets. After both data sets had feature space realizations we used the feature space rotation of Section 6.2.2. Again, as there was no included ground truth with this data set, we choose to compute the endmembers using VCA

algorithm and then create an abundance map. For the number of endmembers to select we choose to follow the results of [33] which indicated that 11 was the ideal number for this scene. We did this process on the original hyperspectral data set and on the fused hyperspectral and LIDAR data set. In Figure 6.18 we see the image classified by the produced abundance maps. We notice that first the LE and feature space rotation fusion produced a much better abundance map than the unprocessed hyperspectral. This is expected because in the higher dimensional space it is more difficult to pick representative signatures. The difference between the LE analysis and the feature space rotation fusion is much closer in performance but we will claim the fusion results is superior. We note that the grass and tree areas in the fusion result are better separated. In the LE result these classes look very noisy. This can been seen as a direct result of the LIDAR information, as trees and grass, though spatially similar, give very different LIDAR returns. We also notice in the fusion result that roof tops rendered much cleaner and the roads are separated from the roof tops.



(a) **(b)** **(c)**

Figure 6.18: *The abundance maps resultant from the VCA endmember extraction with 11 endmembers on the (a) HSI dataset, (b) LE embedding on the hyperspectral dataset, and (c) feature space rotation fusion of hyperspectral and LIDAR*

Thus, we have shown the validity of our approach to the problem of LIDAR and hyperspectral data fusion and we have created a broad paradigm for the fusion of multimodal data, and not just LIDAR and hyperspectral data. In the next section we move to the fusion of multiple hyperspectral data sources.

6.3 HSI-HSI Fusion

The fusion of multiple hyperspectral data sources is the next problem which we will study. This problem, usually phrased for change detection or pan sharpening, here will be defined as the combination of two hyperspectral images not necessarily acquired from the same sensor.

We will study two different setups for this fusion. In Section 6.3.1 we will present a fusion method where the pixels of two images are registered on a subset

of their pixels [39]. In the 6.3.2 we will present a fusion where there is no known registration between the two data sets. Both of these setups will use the feature space rotation fusion methodology. When there is partial overlap we will split our data into three sets, one containing the overlap, and the other two containing the non overlap from each data set. After embedding each of the three sets separately we will learn a rotation into the overlap embedding. For the second setup, where there is no pixel registration, we will use a graph matching algorithm to develop an approximate pixel registration in the feature space then perform a rotation from one feature space into another.

6.3.1 Partial Overlap

If the data sets have pixel registration on only a subset of each of their total number pixels then we shall make a slight modification to the Feature Space Rotation algorithm, see Figure 6.19. For simplicity let A and B be two heterogeneous data sets which we wish to fuse and let C be their intersection (i.e., the set of registered pixels which have been concatenated so as to contain the full spectral information from each sensor). We shall now find a feature space embedding for A, B, and C, denoted F_A, F_B, and F_C, via the Laplacian Eigenmaps algorithm. Now a rotation, $\Theta_{F_A \to F_C}$, is learned from F_A to F_C using the Feature Space Rotation algorithm. Similarly, a rotation, $\Theta_{F_B \to F_C}$, is learned from F_B to F_C. Now A and B are represented in a common feature space that was learned using only a subset of the total pixels in A and B.

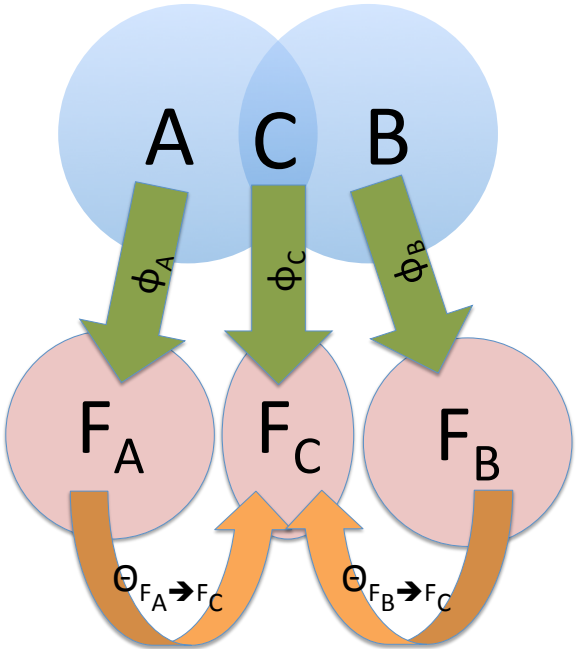


Figure 6.19: *Data sets A and B with overlap $C = A \cap B$ realized as feature space representations F_A, F_B, and F_C, by Laplacian Eigenmaps and then rotated into a common space via rotations $\Theta_{F_A \to F_C}$ and $\Theta_{F_B \to F_C}$*

We look at two different ways to select the overlapping and non-overlapping data points. In Experiment #1 we select the partition at random but preserve complete lines; we choose columns of the image for convience. This procedure equates to simulating data lost during transmission from two separate sensors. In Experiment #2, we simulate the situation in which two different sensors both observe the same scene but there is only a small amount of overlap between the observed pixels. In this experiment the overlap set is chosen at random with no spatial continuity conditions applied. Experiment #2 reflects a situation where two different sensors made measurements over the same geographic area within a short time frame but due to the orientation of the sensors only a fraction of the recorded

pixels corresponded spatially within the margin of the error of the sensors spatial resolutions. For both Experiment #1 and #2, we split the available spectrum from our real data set into two parts, even and odd channels, for the data sets A and B as denoted in Figure 6.19.

As a metric for how well the fusion process performed we shall use classification based upon a known set of the ground truth for the Urban and Indian Pines data sets. Classification will be performed using the simple 1NN algorithm between the training and validation data. We shall use 25% of the ground truth, chosen as by class, to train the classifier and retain the rest of the ground truth in a validation set. The training data will be picked at random and to remove irregularities in the selection of the training data, we shall repeat the classification ten times and average the results. Aside from our measures of classification accuracy, OA and CA, we will also look at the comparison metric, CM,

$$CM_O := \frac{\left\| \Theta_{F_A \to F_B} F_A \Big|_O - \Theta_{F_B \to F_A} F_B \Big|_O \right\|_F}{|O|}. \tag{6.3}$$

In (6.3) we measure how close F_A restricted to the overlap O and rotated into F_B, denoted as $\Theta_{F_A \to F_B} F_A \Big|_O$, is to F_B restricted to the overlap O and rotated into F_A, denoted as $\Theta_{F_B \to F_A} F_B \Big|_O$.

The classification percentages and comparison metric results can be found in Tables 6.11, 6.12, 6.13, and 6.14. First we see that in both scenes the p% overlap results show a drop in classification performance as compared with running LE on the complete data set. This result makes sense as the graph underlying the embeddings with p% overlap are learned on only half of the spectral information

and a small performance drop is expected. In a real situation it would not be possible to have the LE on HSI result the way we did here as the concatenation and weighting of the channels would not be trivial. We also see a general downward trend in classification accuracy as p decreases from 100 to 25; this too is to be expected. What we find encouraging is the drop off is small. The results from the CM metric shows that as the amount of overlap between the two data sets decreases, the difference between mapping F_A into F_B as compared with mapping F_B into F_A, restricted to the overlap, increases. This too is to be expected and this can be used as a means to devise a metric as to how well the fusion process will work in the future on data sets that don't have ground truth.

	OA	CA	CM
LE on HSI	82.02	77.55	·
100% overlap	81.36	76.49	0.0504×10^{-4}
75% overlap	81.40	77.15	0.0686×10^{-4}
50% overlap	81.15	77.24	0.0722×10^{-4}
25% overlap	79.94	73.05	0.1240×10^{-4}

Table 6.11: *Classification and CM Results for Urban - Experiment #1. LE on HSI is classification on the complete data cube after LE algorithm and p% overlap is the spatial overlap between sensors.*

	OA	CA	CM
LE on HSI	61.28	56.84	·
100% overlap	59.82	54.98	0.1226×10^{-3}
75% overlap	58.52	53.40	0.1410×10^{-3}
50% overlap	57.97	50.00	0.1606×10^{-3}
25% overlap	57.36	51.76	0.1948×10^{-3}

Table 6.12: *Classification Results and CM for Indian Pines - Experiment #1. LE on HSI is classification on the complete data cube after LE algorithm and p% overlap is the spatial overlap between sensors.*

	OA	CA	CM
LE on HSI	82.02	77.55	·
100% overlap	81.36	76.49	0.0504×10^{-4}
75% overlap	81.56	77.27	0.0626×10^{-4}
50% overlap	80.31	75.81	0.0778×10^{-4}
25% overlap	79.78	74.12	0.0943×10^{-4}

Table 6.13: *Classification and CM Results for Urban - Experiment #2. LE on HSI is classification on the complete data cube after LE algorithm and p% overlap is the spatial overlap between sensors.*

	OA	CA	CM
LE on HSI	61.28	56.84	·
100% overlap	59.82	54.98	0.1226×10^{-3}
75% overlap	59.39	55.26	0.1439×10^{-3}
50% overlap	57.51	54.05	0.1663×10^{-3}
25% overlap	57.24	51.13	0.2028×10^{-3}

Table 6.14: *Classification Results and CM for Indian Pines - Experiment #2. LE on HSI is classification on the complete data cube after LE algorithm and p% overlap is the spatial overlap between sensors.*

6.3.2 Without Point Registration

We will now discuss how we can avoid requiring an a prior point registration. For the other examples which we have studied the data has been perfectly co-registered or the there has been some amount of overlap between the two data sets. We seek to avoid making this assumption in this section as it is impractical in the real world to have aligned or registered data sets without some amount of preprocessing. The co-registration of remote sensing data sets is another research topic in its own right with rich mathematics, see, e.g., [89]. We shall here build a new technique which uses graph matching algorithms to find a registration between two embedded data sets. After a registration is known, the feature space rotation fusion will be used to fuse the data sets. To summarize, in this section we will only

assume that the two data sets were acquired over roughly the same area; the sensor types, number of channels, and number of recorded data points can all be different.

We will start with a brief introduction to graph matching, for a more complete introduction see [44]. There are two main types of graph matching, exact and inexact. Exact methods provide a 1:1 matching between two graphs and thus require a common number of vertices, we would like to avoid this constraint, so instead we will focus on inexact graph matching. These methods relax the 1:1 requirement and instead require only 1:many relationship which allows for matching betweens graphs of unequal number of nodes. This is much more practical situation, as real world sensors record at different spatial fidelities. In this thesis we will focus on the methods presented in [100, 81] as they rely on eigenvectors of the graph Laplacian which we have calculated already during the LE analysis.

Suppose first that we have two data sets, X_1 and X_2. Suppose also that we have analyzed these data sets using LE, thus we have formed a graph representation, G_1 and G_2, a graph Laplacian, $L_1 = D_1 - G_1$ and $L_2 = D_2 - G_2$, and a lower dimensional embedding Y_1 and Y_2. Recall that Y_1 and Y_2 are the eigenvectors corresponding to the respective graph Laplacians. Also, recall that the Y_1 and Y_2 are composed of the eigenvectors corresponding to the smallest eigenvalues, i.e., they contain the low frequency information of the graph. It works out well that LE requires this information as well so there is no overhead is this regard. The reason we focus on the low-frequency information is that we are trying to find point correspondences between different graphs. Low-frequency information is fairly robust to small deformations and structure changes, thus making it perfect for learning point

correspondences between two data sets.

For now, we shall concentrate on the first K eigenvectors of Y_1 and Y_2. The first K eigenvectors are chosen since the primary eigenvectors should describe the dominate features of the scene with following eigenvectors describing smaller features and generally being more noisy. It is thus desirable to match only based on the primary dominate features. Our method for learning a rotation between these eigenvectors is originally proposed in [100] for the purposes of shape matching. Due to the nature of eigenvalue/eigenvector pairs, the feature extraction algorithms is contingent on a combinatorially hard matching process. First, eigenvectors can match up to a sign difference. Second, the eigenvectors which represent the feature space cannot be directly matched, as order in general is not illustrative. So when attempting to align K feature vectors, we must compare $2^K K!$ K-tuples.

To eliminate problems associated with unequal sample sizes, it is desirable to look at the histograms belonging to each of the feature vectors sets, H_1 and H_2, for a fixed number bins b. We also normalize the histograms to be in the range of $[-1, 1]$ to aid in comparison. Now let the distance, D, between the sets of histograms be:

$$D(s, p, H_1, H_2) = \sum_{i=1}^{K} \|s(i)H_1^{p(i)} - H_2^i\|_2^2, \tag{6.4}$$

where s is a vector representing the signs (+,-) and p is a vector representing the permutations to the ordering of the histograms. To improve the histogram matching process one can apply a smoothing window to each histogram to eliminate the influence of outliers. As finding the best matching is still a complicated calculation we can employ the Hungarian algorithm, which was mentioned in Chapter 5, to

efficiently find the matching which minimizes the distance D.

Remark 6.3.1. To improve results, it might also be advantageous to modify (6.4) to match based only on the smallest $m \leq n$ distances, i.e.:

$$D_m(s, p, H_1, H_2) = \sum_{i \in M} d(s(i)H_1^{p(i)}, H_2^i),$$

where M is an indicator function for the smallest m feature vector distances.

Now that we know the correspondences between the histograms, and by extension between the sets of eigenvectors, let us rearrange the eigenvectors of Y_2 and apply any sign changes necessary to match those of Y_1. To find the registration between Y_1 and Y_2 we can employ an Expectation Maximization (EM) algorithm algorithm. This procedure calls for letting the Y_1 eigenvectors act as cluster centroids of normally distributed clusters and Y_2 eigenvectors act as observations. During the designed EM procedure the likelihood that each $y_j \in Y_2$ belongs to the cluster $y_i \in Y_1$ is found, for details please see [100]. These probabilities, then properly weighted, can be used as a set of weights which we denote as S. Now that we have a set of weights to express one sets' data points in terms of weighted combinations of the other data sets' data points, we use this setup the registration which is needed for the feature space rotation method. In Figure 6.20 we see a diagram representing the proposed method.

To test the effectiveness of this proposed fusion method we require two hyperspectral data sets which were acquired over the same approximate territory. As access to such a data set is problematic, we modify the Indian Pines data set to suit our needs. Let X be the $145 \times 145 \times 200$ hyperspectral data cube for Indian Pines.

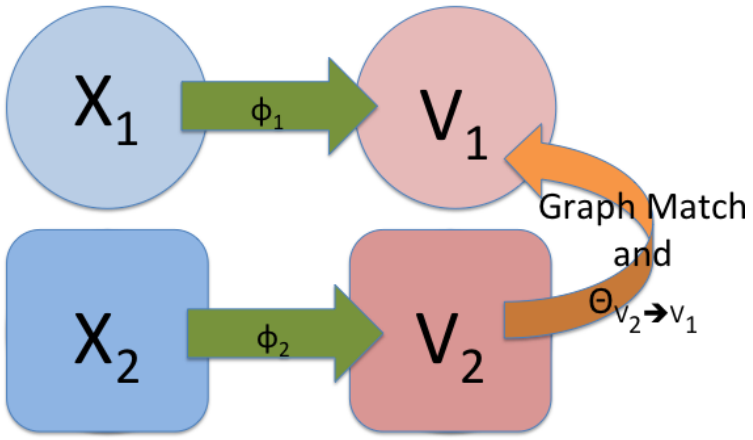


Figure 6.20: *Fusion of X_1 and X_2 without any a priori point registration via graph matching and feature space rotation.*

We can form subsets of X, denoted X_1 and X_2, where X_1 collects the odd columns and odd channels of X, and X_2 collects the even columns and even channels of X. We are thus left with a $145 \times 73 \times 100$ data cube X_1 and a $145 \times 72 \times 100$ data cube X_2. By their construction, X_1 and X_2, have no pixel registration (or overlap), share no common spectral channels and have a different number of data points. Since the data sets do not have common spectral channel centers direct measurements between the data sets is not possible without interpolation. As was outlined above we first embed X_1 and X_2 into a feature space by finding their lower dimension representations, Y_1 and Y_2, using the LE algorithm; see Figure 6.21 and 6.22 for a collection of the eigenimages for each data set.

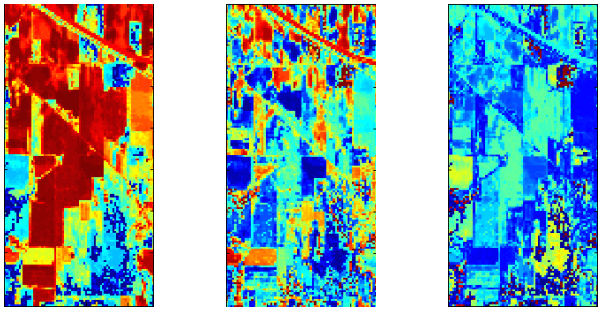


Figure 6.21: *Several eigenimages from X_1 data set.*

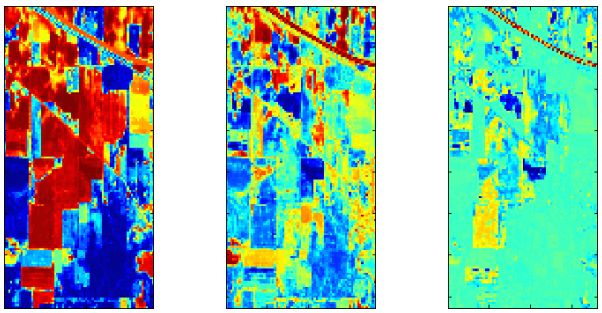


Figure 6.22: *Several eigenimages from X_2 data set.*

For a visualization, we can plot the leading two eigenvectors contained in Y_1 and Y_2, see Figure 6.23.

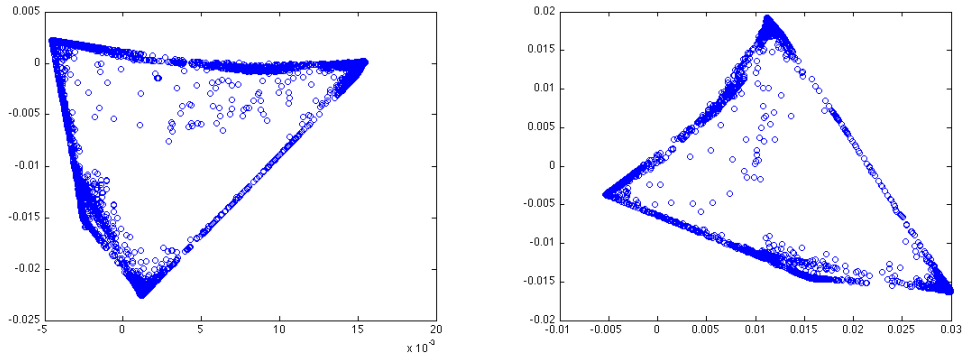
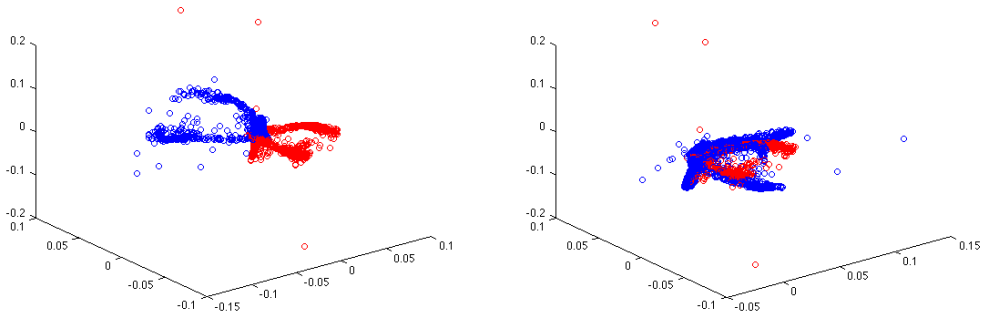


Figure 6.23: *The leading two eigenvectors contained in Y_1 (left) and Y_2 (right) plotted as x-y coordinates. Notice that the shape is approximately equivalent but is orientated in different ways.*

For the histogram matching process we choose to use the first three eigenvectors from each data set and 100 bins. Both of these constants were chosen by visual inspection of the embedding. We could have chosen these constants algorithmically by looking at entropy in the eigenimages to chosen how many eigenvectors to match and smoothness of the normalized histograms over a set of number of bins to chose the number of bins. With the now matched eigenvectors between the two embeddings we represent the coordinates described by the first three eigenvectors from Y_2 as combinations of the coordinates described by the first three eigenvectors from Y_1 using an EM algorithm. We find the rotation from Y_2 to Y_1:

$$\mathcal{O}_{Y_2 \mapsto Y_1} = (Y_1^T S Y_2)^T \tag{6.5}$$

where S expresses the relationship between the coordinates described by Y_1 and Y_2. Note the similarity between (6.5) and (6.2); if S is the identity matrix these are equivalent. We denote the now rotated data set $Y_{2 \mapsto 1}$, see Figure 6.24.



(a) **(b)**

Figure 6.24: *(a) The leading three eigenvectors contained in Y_1 (red) and Y_2 (blue) plotted as x-y-z coordinates. (b) Y_2 matched and rotated into the feature space formed by X_1.*

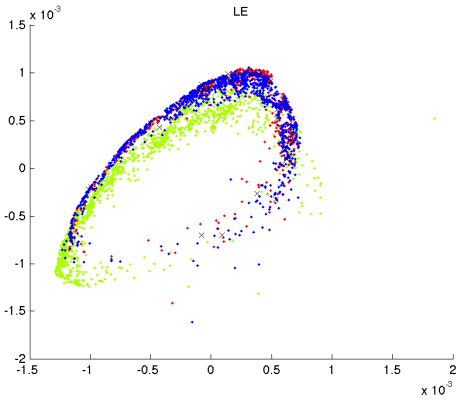
To judge whether the fusion was successful or not we compare the classification scores between Y_1 and the concatenated data set $[Y_1; Y_{2\mapsto 1}]$. A positive test here is any improvement in the classification scores, because, as we we theorized above this fusion process should allow for the transfer of spectral information from one heterogeneous unaligned data set to another. Here we restrict the ground truth to only data points that were contained in X_1 and use 20% of the remaining ground truth to train the 1NN classifier; the classification results can be seen in Table 6.15. We notice that the fused data set outperforms X_1 by almost 13%. This is evidence that the fusion methodology was able to correctly pass the additional spectral information, which was only contained in X_2, to the fused product. This extra spectral information was thus able to improve the classification score by giving added material discernability power.

Data Set	% OA	% AA
Y_1	60.15	59.32
$[Y_1; Y_{2\mapsto 1}]$	67.89	64.22

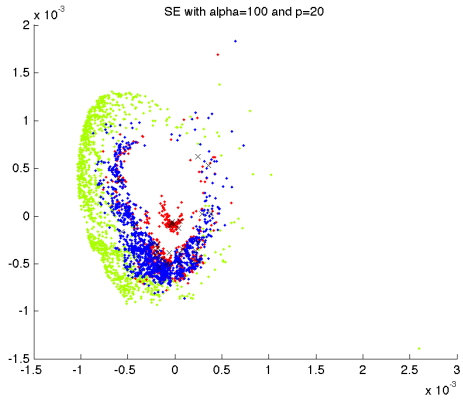
Table 6.15: *The results from non overlapping heterogeneous data set X_1 and X_2. derived from Indian Pines scene, using Feature Space Rotation fusion.*

6.4 A Priori Knowledge Fusion

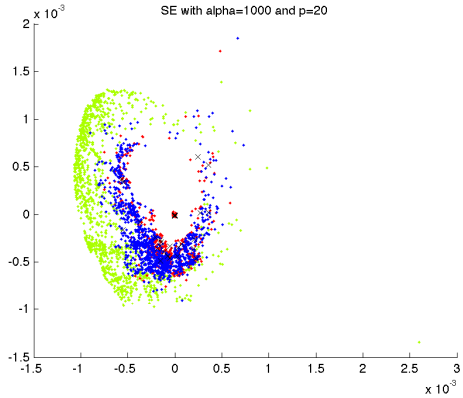
In this section we propose the use of the potentials defined in Section 3.4 as a means to fuse spectral information with expert knowledge. The idea is to have an expert choose pixels prior to the embedding process and label them with a potential term. Recall that we can choose a barrier potential or a cluster potential. In Figure 6.25 we assign a barrier potential to one of three classes. As we would expect this class, as α increases, is pulled toward the origin. A problem that arises is that while being pulled to the origin, the class with the barrier potential is pulled through other classes thus interfering with a classification algorithm.



(a) $\alpha = 0$



(b) $\alpha = 100$

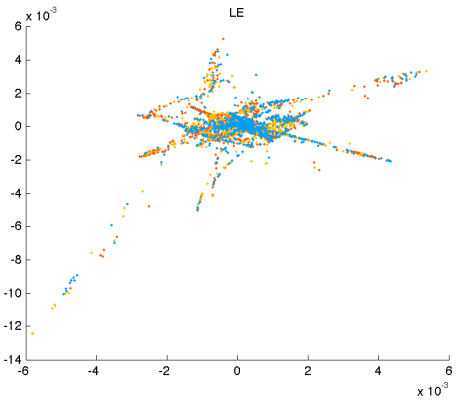


(c) $\alpha = 1,000$

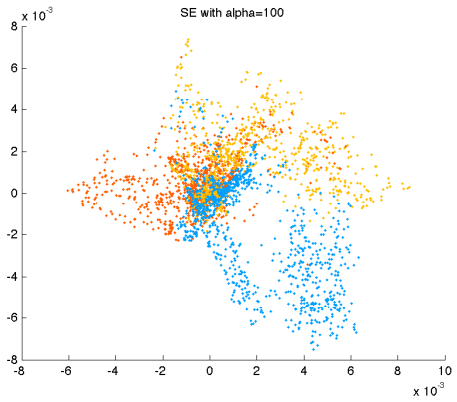
Figure 6.25: *Dimensions 4 and 5 of the embedding for classes 1, 3, and 7 with barrier potential placed on small percentage of class 1's (red) pixels from the Pavia University data set. (a) is Laplacian Eigenmaps (b) and (c) are Schrödinger Eigenmaps with* $\alpha = 100$ *and* 1000 *respectively.*

A better solution is to use the cluster potentials. Here we identify several pixels from each class prior to the embedding and create a potential which sequentially identifies them. As it is difficult to obtain such expert knowledge, we propose here to cluster the scene with k-means and use the learned clusters to build the cluster

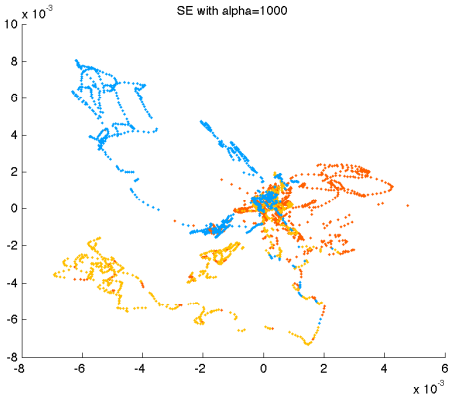
potential. We can also grid the image and perform k-means on small subimages thus adding spatial information. In Figure 6.26 we see the result of this procedure. Here we can very clearly see the spatially and spectrally formed clusters which are learned by the SE analysis.



(a) $\alpha = 0$



(b) $\alpha = 100$



(c) $\alpha = 1,000$

Figure 6.26: *SE Cluster Potentials for Indian Pines plotted with dimensions 17 and 22 of the embedding for classes 2, 3, and 10 from the Indian Pines data set. (a) is Laplacian Eigenmaps (b) and (c) are Schrödinger Eigenmaps with* $\alpha = 100$ *and* 1000 *respectively.*

6.5 Extensions to Other Remote Sensing Problems and Future Work

This thesis looked at the problem of image classification but the other central problems of remote sensing; anomaly, target, and change detection may also be studied with these methods.

Anomaly detection is perhaps the easiest problem to address with our methods. For anomaly detection, we attempt to rank all pixels in a scene with a number from 0 to 1 based on how anomalous it is to all other pixels. Since our methods are data representation methods at heart, the output of the methods developed here can be passed directly to any of a number of different anomaly detection algorithms, for example RX [113] or Topological Anomaly Detection (TAD) [5]. A simpler solution is perhaps to cluster the outputs of the algorithms and use the largest clusters by pixel count to characterize the background of the image. Once the background has been characterized, the distance between the background average to each pixel can be used as an anomalousness measure. This procedure can be done iteratively, in case the background is not very uniform or varied at different spatial scales, to capture anomalies of different sizes.

Target detection can be looked at in two different ways: in situ and out of scene. In situ target detection involves finding a target from a known source in the scene; this involves simply finding a distance from all pixels to the known target and then deciding on a cutoff between positive and negative target detection. Our methods are well suited for this type of target detection in the same way they are for image classification. Out of scene target detection is a harder problem because

it involves finding a target given a lab derived spectrum. Assuming there is a target in the scene corresponding to the lab spectrum, due to the effects of the atmosphere the lab spectrum will not match pixels in the scene. This problem is traditionally solved by applying atmospheric correction codes, see, e.g., [101] and [23]. Our methods, by representing the various sources of the data in the feature space, make applying correction codes difficult after the embedding. Instead, as suggested in [141], it is possible to inject the target spectrum in the graph representation of the data. A subgraph can be constructed from the target pixel by varying several of the parameters used in the MODTRAN atmospheric modeling code base [21]. This will create a target subspace which then can be modeled by a complete graph. Edges between the created subgraph and the data graph then can be found in the traditional nearest neighbor way. In the feature space, the problem of out of scene target detection then becomes an in situ target detection problem.

Change detection is the problem of finding what, if any anything, has changed between two temporally separated data collections from the same area. Calculating a distance between the two images is problematic because the images may have been acquired using different sensors with unequal recorded wavelengths or nonaligned wavelength centers. To adapt our methods we would make use of the feature space rotation methodology introduced in [41]. Once both data sets have been realized in the feature space, a rotation can be calculated. Now in a common feature space a direct measurement between pixels is possible. If the data sets are unaligned then the graph matching techniques we mentioned can be used to align the different data sets in the feature space.

Aside from other problems in remote sensing our methods can also be adapted easily to study the fusion of other types of remote sensing data. We are particularly interested in the fusion of SAR or multiple view angle images with HSI and LIDAR. Our methods should also not be limited to remote sensing, e.g., in the case of biomedical imaging, to the fusion of Optical Coherence Tomography (OCT) with multispectral retinal imagery. To summarize, at the heart of our methods lies the representation of multiple data sets in a common space, with classification of remote sensing data being just one of the many possible applications for this new representation.

Chapter 7

Nyström Method

A significant portion of the computational cost of any dimension reduction routine is the eigendecomposition of the resulting kernel matrix. To perform it exactly, results in a computational complexity of $O(N^3)$. For our methods developed in Chapter 6 to be practical, we must lower the complexity of this eigendecomposition. Thus, we see the numerical acceleration techniques to be directly tied to the developed data fusion methods as to fuse large data sets can only be accomplished if the complexity is lowered. To improve this, we first look to take advantage of the real positive semi-definite structure that is inherent in the dimension reduction kernel. Cholesky Factorization offers a starting point to quickly diagonalize the kernel. If we now also consider that the kernel will be sparse to a known level (related to the number of nearest neighbors chosen) we can again realize a reduction in compute time. Iterative eigensolvers, namely Subspace Iteration or more complicated methods like Krylov Subspace methods (Implicitly Restarted Arnoldi or Lanczos), Jacobi Davidson, and Preconditioned Conjugate Gradient Rayleigh Quotient minimization or random matrix decomposition have been developed to take advantage of the structure of sparse kernels. Another eigensolver based approach is to make use of the large number of eigensolvers which have parallel implementations. All of these computational improvements however are limited in that their complexity

is based on the size of the kernel or the number of vertices in the graph. To get a greater reduction in compute time we will look at matrix eigendecomposition approximation schemes, which rely on sampling the kernel matrix, i.e., methods that find an eigenbasis for a subset of the kernel and extend these eigenvectors to unseen entries.

The leading solutions for sampling the kernel include: Random SVD, CUR, and the Nyström method. We have chosen to concentrate on the Nyström method, and will discuss it at length in the following paragraphs, for this thesis because of its rich mathematical content and number of citations in leading literature. The CUR algorithm [98], taking its name from the sub matrices C, U, R which give us $A \approx CUR$, is a general low rank matrix approximation scheme. The CUR algorithm however does not respect the symmetric, positive, semi-definite property that is inherent in the kernel matrices which we are studying and has a higher complexity than the Nyström method. Random SVD uses Random Projections with the SVD algorithm, which we discussed in Section 5.1.

Using the Nyström method to aid in the computation of kernels relating to dimension reduction was proposed in [69, 136, 7], we are able to find the approximate eigendecomposition of the kernel related to dimension reduction methods. This method has the advantage, over the other methods mentioned above, in that it is designed to operate on symmetric, positive, semi-definite matrices. We now provide a detailed discussion of this technique.

The Nyström method, developed by Evert Nyström in 1930 [108], arose from

the solution of Fredholm-type differential equations [55, 59] which have the form:

$$\int_a^b K(x,y)\phi(y)dy = \lambda\phi(x), x \in [a,b], \tag{7.1}$$

where K is a kernel function. We first find the solution to (7.1) on a set of N uniformly sampled quadrature node points, $\xi_1, \ldots, \xi_n \in [a,b]$, using the quadrature rule:

$$\int_a^b f(y)dy = \sum_{i=1}^{N} w_i f(\xi_i), \tag{7.2}$$

where $\{w_i\}_{i=1}^N$ are the set of quadrature weights usually chosen to uniformly sample $[a,b]$. Now we can use (7.2) to approximate the solution of (7.1),

$$\int_a^b K(x,y)\phi(y)dy \approx \sum_{i=1}^{N} w_i k(x,\xi_i)\hat{\phi}(\xi_i), \tag{7.3}$$

where k is the kernel matrix. (7.3) can be posed as eigenvalue problem:

$$\sum_{i=1}^{N} w_i k(x,\xi_i)\hat{\phi}(\xi_i) = \hat{\lambda}\hat{\phi}(x), \tag{7.4}$$

where $\hat{\phi}$ and $\hat{\lambda}$ are an approximate eigenvalue-eigenfunction pair. Now the Nyström method is the solution for $\hat{\lambda}$ and $\hat{\phi}$ by creating a system of N equations by letting $x \in \{x_i\}_{i=1}^N$:

$$\sum_{i=1}^{N} w_i k(x_i,\xi_i)\hat{\phi}(\xi_i) = \hat{\lambda}\hat{\phi}(x_i), i = 1, \ldots, N \tag{7.5}$$

Definition 7.0.1 (Nyström Point)**.** The Nyström points are the discrete set of points, $x = \{x_i\}_{i=1}^{N}$, chosen in the range $[a, b]$ which sample (7.4) to produce a system of N equations in (7.5).

Definition 7.0.2 (Landmark Point)**.** A landmark is synonymous with a Nyström point but it is more common to refer to a Nyström point in the context of numerical integration of integral equations and a landmark in the context of discrete data dependent kernels.

For simplicity, let the Nyström points and the quadrature points be equivalent, as this guarantees that if K is symmetric then k will also be symmetric. Then, for $\hat{\lambda}_m \neq 0$ the exact eigenfunction $\hat{\phi}_m$ on the Nyström points can be extended to $\bar{\phi}_m$, i.e. the Nyström extension, on $[a, b]$ by using (7.4):

$$\hat{\lambda}_m \bar{\phi}_m(x) = \sum_{i=1}^{N} w_i k(x, \xi_i) \hat{\phi}_m(\xi_i),$$

or equivalently,

$$\bar{\phi}_m(x) = \frac{1}{\hat{\lambda}_m} \sum_{i=1}^{N} w_i k(x, \xi_i) \hat{\phi}_m(\xi_i). \tag{7.6}$$

Definition 7.0.3 (Nyström Extension)**.** The function, $\bar{\phi}_m$ in (7.6) is the Nyström extension of an eigenvector $\hat{\phi}_m$.

Now if we consider the discrete analogue of the above problem for a kernel matrix A (symmetric, positive, semi definite) and a set of landmarks L (for simplicity assume that $L = \{1, 2, \ldots, \ell\}$, $\ell < N$), i.e., a set of rows or columns of A, we can

rewrite A as:

$$A = \begin{bmatrix} A_L & B \\ B^T & C \end{bmatrix}. \tag{7.7}$$

In (7.7) $A_L \in \mathbb{R}^{\ell \times \ell}$ is the submatrix formed by taking the rows and columns corresponding to the set L, i.e. the weights amongst the landmark points, $B \in \mathbb{R}^{(N-\ell)\times \ell}$ is the sub matrix that contains the weights from landmark to non-landmark points and $C \in \mathbb{R}^{(N-\ell)\times(N-\ell)}$ is the submatrix that contains the weights between the non-landmark points.

If we choose $\ell << N$, the eigendecomposition of $A_L = U_L \Lambda_L U_L^T$ will be dramatically cheaper than that of A. Now we note that with some simple linear algebra we can find an approximate eigendecomposition of A, denoted as $\tilde{A}$:

$$\begin{aligned} \tilde{A} &= \tilde{U}\tilde{\Lambda}\tilde{U}^T \\ &= \begin{bmatrix} U \\ B^T U \Lambda^{-1} \end{bmatrix} \begin{bmatrix} \Lambda \end{bmatrix} \begin{bmatrix} U^T & \Lambda^{-1} U^T B \end{bmatrix} \\ &= \begin{bmatrix} U\Lambda U^T & B \\ B^T & B^T A_L^{-1} B \end{bmatrix} \\ &= \begin{bmatrix} A_L & B \\ B^T & B^T A_L^{-1} B \end{bmatrix}. \end{aligned}$$

Thus the Nyström method (extension) gives us eigenvectors as the columns of the $\ell \times N$ matrix,

$$\tilde{U} = \begin{bmatrix} U \\ B^T U \Lambda^{-1} \end{bmatrix}. \tag{7.8}$$

We note that the forms of (7.6) and (7.8) are equivalent. It is also possible to find the Nyström extension for an arbitrary matrix, see [59]. We will, however, restrict ourselves to extending $N \times N$ symmetric, positive, semi-definite kernel matrices as they are the form guaranteed by our Dimension Reduction algorithms, so (7.8) will suffice.

The error involved in approximating A by $\tilde{A}$ can be measured by how well C is approximated by $B^T A_L^{-1} B$, or as how well the rows of B span C. We can quantify this error, $\mathcal{E}_N$, by the norm of the Schur complement,

$$\mathcal{E}_N = \|C - B^T A_L^{-1} B\|_F, \tag{7.9}$$

where $\|\cdot\|_F$ is the Frobenius norm (for a matrix M, $\|M\|_F^2 = \sum_{i,j} |M(i,j)|^2$). If instead of looking at the norm of the Schur Complement, (7.9), we look at the norm squared of the Schur complement, which is referred to as the trace norm, $\|\cdot\|_{tr}$, we notice that:

$$\begin{aligned} \|C - B^T A_L^{-1} B\|_{tr} &= \|C - B^T A_L^{-1} B\|_F^2 \\ &= \text{trace}(C) - \text{trace}(B^T A_L^{-1} B). \end{aligned} \tag{7.10}$$

This alternative error measure can be useful, as it has been shown that among all unitary invariant norms $\|\cdot\|$, $\|\cdot\|_{tr} \geq \|\cdot\|$. Thus the trace norm offers an upper bound on the error associated with the Nyström extension.

Until now we have not specified how the landmark points are to be chosen for use in the Nyström method; essentially there are two means for their selection, either deterministically or randomly.

Random methods offer the advantage of lower computational cost (basically just the expense needed to generate the sampling distribution) and some proven bounds on the error (unlike deterministic methods). On the other hand, deterministic methods, which either are iterative greedy methods or minimize a cost function, offer the advantage of reproducible results and smaller landmark sets that can be designed to concentrate on certain qualities of the data.

We will now explore some of the error bounds and variations on the Nyström method.

We will then look at the Nyström method in relation to Frame Theory and Sigma-Delta Quantization Section 7.2.

7.1 Landmark Selection Methods and Error Bounds

To optimize the Nyström method can be thought of as selecting a group of landmarks which will minimize the error in (7.9) or (7.10).

It might be illustrative to first look for when a set of landmarks gives perfect reconstruction, i.e. $\mathcal{E}_N = 0$. This can occur when $L = \{1, \ldots, N\}$, i.e. the full set of data points; in this situation the error in (7.9) and (7.10) are both 0. This can also occur, in a nontrivial way, when A has rank less than the number of landmarks. It is also required, as shown in [7], that for exact reconstruction, $\text{trace}(A_L) \neq 0$. In a similar vein, the authors in [139] note that the Nyström extension of the kernel matrix $A(x_i, x_j)$ will be exact if there exists landmark points l_p and l_q such that $l_p = x_i$ and $l_q = x_j$.

If we sample the landmarks at random uniformly, [8] provides the following theorem on the bounded expected value of the error in the trace norm.

Theorem 7.1.1 (Belabbas and Wolf, 2009)**.** *Let A be a symmetric, positive, semi-definite matrix with Nystrom extension $\tilde{A}$ with landmark set L, $|L| = \ell$. Averaging over this choice we have*

$$E[\|A - \tilde{A}\|_{tr}] \leq \frac{N - \ell}{N} \operatorname{trace}(A).$$

In [8] the authors introduce annealed determinantal sampling as an improvement over uniform sampling. This sampling scheme assigns probabilities to each column of A, for a fixed parameter $s \geq 0$ and landmark set size ℓ:

$$p^s(L) \propto det(A_L)^s.$$

For $s = 0$ the annealed determinantal sampling gives the uniform sampling case. However for, $s > 0$, the error measured by (7.10) will be 0. The difficulty lies in the combinatorially complex sampling of the distribution $p^s(L)$, as there are $\binom{N}{\ell}$ possibilities. For the special case of $s = 1$ the authors are able to show that the expected value of the trace norm error can be bounded by the spectrum of A:

Theorem 7.1.2 (Belabbas and Wolfe, 2009)**.** *Let A be a symmetric, positive, semi-definite matrix with Nyström extension $\tilde{A}$ where the landmark set L, $|L| = \ell$ was chosen with the annealed determinantal sampling scheme with parameter $s = 1$. Then,*

$$E[\|A - \tilde{A}\|_{tr}] \leq (\ell + 1) \sum_{i=\ell+1}^{N} \lambda_i,$$

where λ_i is the ith largest eigenvalue of A.

In [59] the authors look at probabilistic bounds for creating a optimal rank k approximation of A given a set of landmarks of size ℓ chosen with sampling probabilities equal to:

$$p_i = \frac{A(i,i)^2}{\sum_{i=1}^{N} A(i,i)^2}. \tag{7.11}$$

The algorithm they developed, which we will call here the Weighted Column Selection Nyström algorithm, selects ℓ columns of A i.i.d. with replacement using the probabilities defined in (7.11) to form the landmark set L. The $N \times \ell$ matrix T is assembled by the selected the columns corresponding to L in A and dividing them by $\sqrt{\ell p_i}$. The $\ell \times \ell$ matrix W is then created such that $W(i,j) = A(i,j)/(\ell\sqrt{p_i p_j})$. Finally the approximation $\tilde{A}_k = TW_k^+T^T$ is found where W_k^+ is the pseudo inverse of the best rank k approximation of W. The algorithm can be simplified by using a uniform distribution but affects the tightness of the bound in the following theorem.

Theorem 7.1.3 (Drineas and Mahoney, 2005)**.** *Suppose A is a $N \times N$ symmetric, positive, semi-definite matrix, let $k \leq \ell$ be a rank parameter and let $\tilde{A}_k = T$ be constructed using the Weighted Column Selection Nyström algorithm by sampling ℓ columns of A with probability $\{p_i\}_{i=1}^N$ defined by* (7.11)*. Let $r = \text{rank}(W)$ and ket A_k be the best rank k approximation to A. In addition let $\epsilon > 0$ and $\nu = 1+\sqrt{9\log(1/\delta)}$. If $\ell \geq 64k/\epsilon^4$, then*

$$E[\|A - \tilde{A}_k\|_F] \leq \|A - A_k\|_F + \epsilon \sum_{i=1}^{N} A(i,i)^2$$

and if $\ell \geq 64k\nu^2/\epsilon^4$ then with probability at least $1-\delta$,

$$\|A - \tilde{A}_k\|_F \leq \|A - A_k\|_F + \epsilon \sum_{i=1}^{N} A(i,i)^2.$$

The authors in [85] introduce Ensemble Nyström which is based on a weighted mixture of Nystrom approximations. In Ensemble Nyström, like in the standard Nyström algorithm a set of landmark points, L is first selected randomly without replacement. Let $|L| = \ell p$, where ℓ and p are positive integers. The algorithm now calls for partitioning L into p subsets, $L_1, \ldots, L_p$, where $|L_i| = \ell$. Each of these subsets, L_i, with the Nyström method, produces an approximation of A, denoted as $\tilde{A}_i$. Now given a set of weights, $\mu = \{\mu_i\})i = 1^p$, the Ensemble Nystrom approximation of A is, $\tilde{A}^{ens}$:

$$\tilde{A}^{ens} = \sum_{i=1}^{p} \mu_i \tilde{A}_i.$$

The authors discuss three methods for choosing the mixture weights, uniform, exponential, or ridge regression. First let V be a subset of the columns of A and let A_V by the submatrix formed from the columns and rows denoted by V. Uniform weights give equal weight to each $\tilde{A}_i$, i.e. $\mu_i = 1/p$. Exponential weights set $\mu_i = \exp(-\nu\hat{\epsilon}_i)/Z$ where $\nu > 0$ is a spreading parameter and Z is a normalization factor ensuring the weights belong to the simplex $\{\mu \in \mathbb{R}^p : \mu \geq 0, \sum_{i=1}^{p} \mu_i = 1\}$ in $\mathbb{R}^p$. $\hat{\epsilon}_i$ is the error associated in approximating the kernel A_V with $\tilde{A}_i^V$ via the Nyström extension with the landmarks coming from landmark set L_i. The ridge regression weights use V to train μ to optimize the regression objective function:

$$\min_{\mu} \lambda \|\mu\|_2^2 + \left\| \sum_{i=1}^{p} \mu_i \tilde{A}_i^V - A_V \right\|_F^2,$$

where $\lambda > 0$.

The Ensemble Nyström method has complexity that is approximately p times that of the standard Nyström method (more if using the exponential or ridge regression weight method) but is easily parallelizable. For the improved error bounds we refer to [85].

The Max-Min [53] algorithm chooses the landmark set iteratively by first choosing a seed number, $1 \leq s < \ell$, of landmarks at random, then adding them to the set L and removing the corresponding data points from the data set X. Then for each iteration till the set L contains the required number of points the algorithm chooses the new landmark point to add as the data point which maximizes the minimum distances to all other previously selected landmark points. Again, as in the seeding phase, this point is added to L and removed from X.

Sparse Matrix Greedy Approximation (SMGA) [122] works by finding an optimal approximation to the matrix A using a weighted combination of the columns of A which act as basis elements. This procedure can be accomplished via a greedy optimization algorithm. The authors suggest using the Matching Pursuit algorithm which iteratively finds the best column to add to the basis set then projects out that column from the remaining columns in A. Since the number of possible basis sets is equal to $\binom{N}{\ell}$ the search space is limited each iteration to a small subset of the total number of columns.

The Incomplete Cholesky Decomposition (ICD) [67] builds a low rank approximation of the kernel matrix A using the ICD algorithm and then uses the Woodbury formulas to find a decomposition of A. Since any positive definite matrix can be represented by its Cholesky factorization, we can let $A^{(icl)k} = QQ^T$, where Q is

lower triangular and of rank k. If $k = \ell$ then [Bach Jordan 2005] proved that $A^{(icl)}k$ is identical to the Nyström approximation with landmarks equal to the Cholesky pivots.

In [139, 138], the authors introduce Density weighted Nyström, a method designed to remove the flawed assumption that all data points are equally important under the Nyström extension. The authors propose to assign a density $p(\cdot)$ to the landmarks. The major difference now is during the extension phase of the Nyström method a probability density function will appear:

$$\bar{\phi}_m(x) = \frac{1}{c\hat{\lambda}_m} \sum_{i=1}^{N} p(\xi) w_i k(x, \xi_i) \hat{\phi}_m(\xi_i),$$

where $c = \sum_{i=1}^{N} p(\xi_i)$. The authors also propose a block-quantization scheme for the kernel matrix, which can be shown to be a special case or the weighted Nyström method. The data set is partitioned into disjoint clusters S_k, $k = 1, \ldots, \ell$, each with size $|S_k|$ and landmark point or cluster representative s_k. So for $x_i \in S_p$ and $x_j \in S_q$, $A(x_i, x_j) \approx A(s_p, s_q)$. A then can be quantized, if ordered correctly, to produce $\bar{A}$, a blockwise constant matrix of the distances between landmarks. A_L can be formed from the landmark distances now and decomposed as per the standard Nyström method, the weighting will be proportional to the cluster sizes. The authors offer the following bound on the approximation error.

Theorem 7.1.4 (Zhang and Kwok, 2009)**.** *If the data set X is partitioned into ℓ clusters with centers $L = \{\ell_i\}_{i=1}^{\ell}$, $c(i)$ center indicator function, and the blockwise constant kernel is obtained by replacing each x_i with its corresponding cluster center.*

Then using the stationary gaussian kernel $k(x,y) = k(\|x-y\|^2/\sigma^2)$,

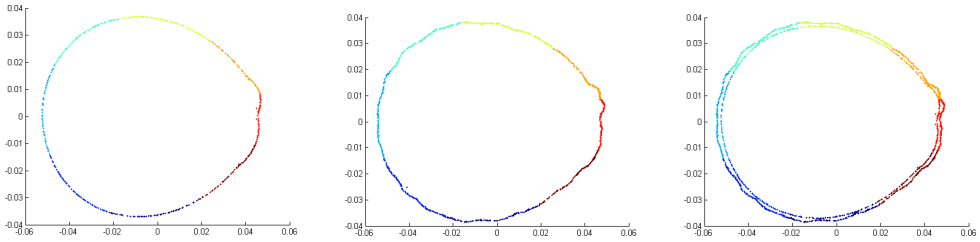
$$\begin{aligned}\|A-\tilde{A}\|_F &\leq 8\frac{\xi^2}{\sigma^4}(NR^2\overline{D^{(2)}} + nR\overline{D^{(3)}} + \frac{1}{4}n\overline{D^{(4)}} + R^2(\overline{D^{(1)}})^2 \\ &\quad + \frac{3}{4}(\overline{D^{(2)}})^2 + 3R\overline{D^{(2)}D^{(1)}} + \overline{D^{(3)}D^{(1)}}),\end{aligned}$$

where R *is the max pairwise distance, and* $\xi = \max_x |k'(x)|$ *(from mean value theorem), and* $\overline{D^{(k)}} = \sum_{i=1}^{N} \|x_i - \ell_{c(i)}\|^k$.

Using the k-means algorithm to form the set L is cheap but results in a non-deterministic algorithm. Sequential sampling is used to seed the k-means algorithm to fix this problem. Ouimet and Bengio have a similar approach called greedy sampling.

The Adaptive Sampling [56] improves upon the uniform random sampling. It updates the probability distribution over the columns of A each iteration. The probabilities are updated each iteration such that they are proportional to the distance squared from the span of the previously selected columns. A later refinement of Adaptive Sampling, Adaptive Partial Sampling [86], lowers the computational burden by measuring the reconstruction error using the partially assembled sub matrix $[A_L; B^T]$ and using this to update the sampling probabilities across the remaining columns.

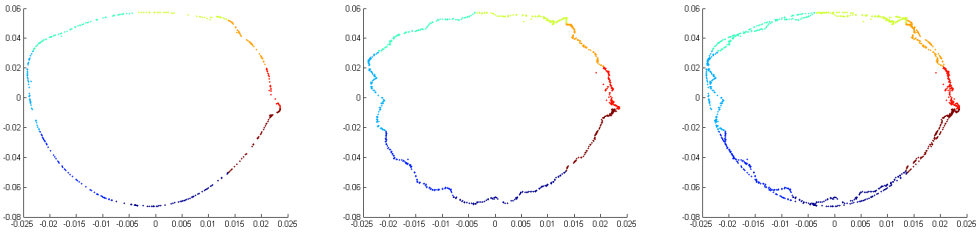
We can now look at the choice of different landmark sets and how it will effect the final embedding. For all the different landmark selection methods we will discuss here we have chosen to use 25% of the total number of data points. Recall the Helix data set from Figure 3.1. First in Figure 7.1 we construct a set of landmarks perfectly uniformly spaced on the data set. Here we are using the a priori knowledge that the



(a) **(b)** **(c)**

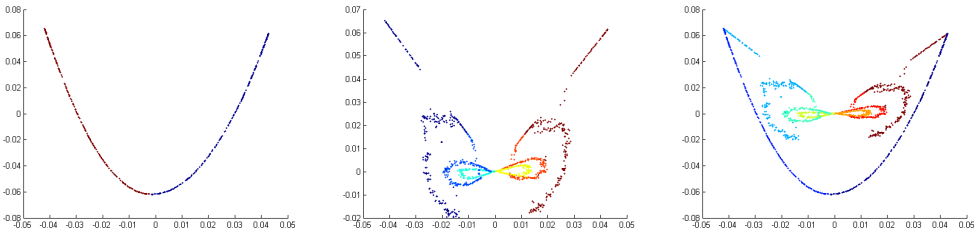
Figure 7.1: *Landmarks are chosen knowing intrinsic structure of the lower dimensional structure, (a) is the embedding of the landmarks, (b) is the Nyström extension for the non landmarks, and (c) is the embeddings from (a) and (b) plotted together*

the two dimensional representation is a circle. We are thus able to sample the circle in uniformly spaced intervals. These chosen indices are then chosen from the Helix. We notice that when sampling knowing the underlying structure of the data the Nyström extension works very well. In Figure 7.2 we sample the landmark uniform randomly with no a priori information. We notice that the final lower dimensional embedding is not as good as 7.1 but the embedding is still close to the ideal circle. Finally, in 7.3, we show the effect of choosing a poor landmark set. In this example we sample all the points on a 1/4 of the arc of circle. This landmark set thus does not provide a good descriptor for the full data set. When the Nyström extension is calculated then there is no good collection of landmark points to represent the non landmark points.



(a) **(b)** **(c)**

Figure 7.2: *Landmarks are chosen with uniform random variable, (a) is the embedding of the landmarks, (b) is the Nyström extension for the non landmarks, and (c) is the embeddings from (a) and (b) plotted together*



(a) **(b)** **(c)**

Figure 7.3: *Landmarks are chosen with a poor sampling function, (b) is the Nyström extension for the non landmarks, and (c) is the embeddings from (a) and (b) plotted together*

7.2 Frames and the ΣΔ Quantization

It is perhaps also illustrative to look at the theory of Nyström's method and selection of landmarks in the language of frames and Sigma-Delta (ΣΔ) Quantization. We will start with a brief summary of both frame theory and ΣΔ Quantization then give motivations for their use with the Nyström method and remote sensing data in general.

7.2.1 Frame Theory

In the following we will present a brief introduction to the theory of frames, for a more complete review see [17, 37, 76].

Coming from the initial work of Duffin and Schaeffer [61] a frame is defined by:

Definition 7.2.1 (Frame)**.** A countable family of elements $F = \{f_k\}_{k\in\Lambda}$ in a Hilbert space $\mathbb{H}$ is a frame for $\mathbb{H}$ if there exits constants A and B, $0 < A \leq B < \infty$ such that for all $x \in H$, $A\|x\|^2 \leq \sum_{k\in\Lambda} |\langle x, f_k\rangle|^2 \leq B\|x\|^2$.

The constants A and B are known as frame bounds. A tight frame is a frame where $A = B$. A normalized frame is a frame where for all k, $\|f_k\| = 1$. Though a frame can be viewed as having a countable number of vectors, for the rest of this treatment we will assume that we have a finite frame of M vectors.

The analysis operator, $\mathcal{F} : H \mapsto \ell_2$, of the frame F is defined by

$$\mathcal{F}(f) \quad = \quad \{\langle f, f_i\rangle\}_{i=1}^{M}.$$

The synthesis operator, which is the adjoint of the analysis operator, $\mathcal{F}^* : \ell_2 \mapsto H$, is defined by

$$\mathcal{F}^*(\{c_i\}_{i=1}^M) = \sum_{i=1}^M c_i f_i.$$

The frame operator is the positive, self-adjoint, invertible operator $\mathcal{S} : H \mapsto H$, defined by,

$$\mathcal{S} = \mathcal{F}^*\mathcal{F},$$

or,

$$\mathcal{S}(f) = \sum_{i=1}^M \langle f, f_i \rangle f_i,$$

for all $f \in H$. It can be shown that $\mathcal{S}$ satisfies the frame bound:

$$AI \leq \mathcal{S} \leq BI,$$

where I is the identity operator on the H. Since $\mathcal{S}$ is invertible, $\mathcal{S}^{-1}$, or the dual frame satisfies,

$$B^{-1}I \leq \mathcal{S}^{-1} \leq A^{-1}I.$$

We can also form a frame from the dual frame as such.

Theorem 7.2.2 (Reconstruction Formula)**.** *Let $F = \{f_i\}_{i=1}^M$ be a frame with frame bounds A and B and let $\mathcal{S}$ be the frame operator then $\{\mathcal{S}^{-1}f_i\}_{i=1}^M$ is also a frame with frame bounds $1/B$ and $1/A$ and for all $x \in H$,*

$$x = \sum_{i=1}^M \langle x, f_i \rangle (\mathcal{S}^{-1} f_i) = \sum_{i=1}^M \langle x, \mathcal{S}^{-1} f_i \rangle f_i.$$

Theorem 7.2.2 will play an important role in using frames for Quantization.

For finite normalized tight frames in $\mathbb{R}^D$ the lower frame constant is M/D. For a normalized tight frame, the lower frame bound, A, measures the redundancy of F. If $A = 1$ then F is a orthonormal basis and thus has no redundancy; when $A \geq 1$ there exists redundancy in F. Frames thus give us the opportunity to robustly represent a signal, which is why they are favored in in signal processing because they deal well with partial data loss, additive noise, and quantization.

7.2.2 $\Delta\Sigma$ Quantization

Quantization is a mathematical technique to obtain a representation of a signal that is ideal for practical purposes, i.e. storage or transmission of a signal. Quantization achieves this by representing a signal x as a weighted sum of atoms of a dictionary $\mathcal{D}$ where $|\mathcal{D}| = \mathcal{D}_0$:

$$x = \sum_{i=1}^{\mathcal{D}_0} c_i d_i, \tag{7.12}$$

where d_i is an atom and c_i is a real or complex number. For our treatment here we will only consider dictionaries of finite size but in theory they can be of countable size. The only problem with (7.12) is that the coefficients, c_i, are not suited for use in the digital environments because they do not have ready binary representations. Thus a second step is used, the quantization step, which forces the continuous range of coefficients to be reduced to a discrete alphabet, $\mathcal{A}$. This second step now offers an approximate representation of x,

$$\tilde{x} = \sum_{i=1}^{\mathcal{D}_0} q_i d_i, \tag{7.13}$$

where $q_i \in \mathcal{A}$. The approximation error, $\mathcal{E}_q$, incurred by the quantization, can now be measured as the difference between (7.12) and (7.13),

$$\mathcal{E}_q = \|x - \tilde{x}\|.$$

A rule must be established that governs how the quantization occurs from the continuous range of c_i's to the discrete collection of q_i's; we will call this the quantization rule. This rule has to predictably quantize each coefficient because once transmitted or stored the signal will need to be recovered by apply the inverse rule.

The basic quantization rule is called Pulse Code Modulation (PCM) [109] and it guarantees if $\mathcal{D}$ is a orthonormal basis then $\mathcal{E}_q$ is minimized for all $x \in H$. Let $\mathcal{D}$ be a normalized tight frame for $\mathbb{R}^D \subset H$, such that for all $x \in \mathbb{R}^D$,

$$\begin{aligned} x &= \frac{D}{\mathcal{D}_0} \sum_{i=1}^{D_0} \langle x, d_i \rangle d_i \\ &= \frac{D}{\mathcal{D}_0} \sum_{i=1}^{D_0} x_i d_i. \end{aligned} \tag{7.14}$$

For a width-parameter $\delta > 0$, the $2\left\lceil \frac{1}{\delta} \right\rceil$ level PCM quantizer, where $\lceil \cdot \rceil$ is the ceiling function, is:

$$q_i = \begin{cases} \delta\left(\left\lceil \frac{x_i}{\delta} \right\rceil - \frac{1}{2}\right) & \text{if } |x_i| < 1, \\ \delta\left(\left\lceil \frac{1}{\delta} \right\rceil - \frac{1}{2}\right) & \text{if } x_i \geq 1, \\ -\delta\left(\left\lceil \frac{1}{\delta} \right\rceil - \frac{1}{2}\right) & \text{if } x_i \leq -1. \end{cases} \tag{7.15}$$

Thus by the PCM quantization rule defined in (7.15), which is essentially rounding to the nearest grid point, we get the quantization version of x, $\tilde{x}$:

$$\tilde{x} = \frac{D}{\mathcal{D}_0} \sum_{i=1}^{\mathcal{D}_0} q_i d_i. \tag{7.16}$$

We can now calculate $\mathcal{E}_q$ using (7.14) and (7.16):

$$\begin{aligned} \|x - \tilde{x}\| &= \|\frac{D}{\mathcal{D}_0}\sum_{i=1}^{D_0} x_i d_i - \frac{D}{\mathcal{D}_0}\sum_{i=1}^{\mathcal{D}_0} q_i d_i\| && (7.17)\\ &= \frac{D}{\mathcal{D}_0}\|\sum_{i=1}^{D_0}(x_i - q_i)d_i\| && (7.18)\\ &\leq \frac{\delta}{2}\frac{D}{\mathcal{D}_0}\sum_{i=1}^{\mathcal{D}_0}\|d_i\| && (7.19)\\ &= \frac{\delta D}{2}, && (7.20) \end{aligned}$$

where (7.19) is from (7.15) and (7.20) is from the frame being normal. PCM does not make use of the desinged redundancy in the frame, so will need a new technique to improve upon (7.20).

Sigma-Delta ($\Sigma\Delta$) Quantization, sometimes also referred to as Delta-Sigma ($\Delta\Sigma$) Quantization, is an improvement because it allows for the redundancy found in the frame representation. We will reference the basic notions of $\Sigma\Delta$ Quantization from [19].

We will start our discussion of $\Sigma\Delta$ Quantization by developing the quantization rule:

Definition 7.2.3. Given $K \in \mathbb{N}$, $\delta > 0$, and the midrise alphabet,

$$\mathcal{A}_K = \{(-K + 1/2)\delta, (-K + 3/2)\delta, \ldots, (K - 3/2)\delta, (K - 1/2)\delta\}, \qquad (7.21)$$

and the $2 - K$ level mid rise uniform scalar quantizer with step size δ,

$$Q(u) = \arg\min_{q \in \mathcal{A}_K^\delta} |u - q|. \qquad (7.22)$$

Let $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$, and let p be a permutation of $\{1, \ldots, N\}$. The first order quan-

tizer is defined by the interative scheme:

$$\begin{aligned} u_i &= u_{N-1} + c_{p(i)} - q_i, \\ q_i &= Q(u_{N-1} + x_{p(i)}), \end{aligned}$$

where u_0 is a constant.

The midrise alphabet, (7.21), creates a set of uniform gridpoints between $-K$ and K spaced at width δ and in (7.22), the quantizer $Q(u)$ is essentially choosing the member of the alphabet which is closest to the u.

While order is usually not important when considering frames or frame expansions the order of the dictionary while quantizing is very important. If the elements of a dictionary are permuted once a quantization has occurred the recoverability will no longer be guaranteed. We thus need to fix an order to the frame elements, one means of accomplishing this is via frame variation.

Definition 7.2.4 (Frame Variation)**.** Let $F = \{f_i\}_{i=1}^N$ be a frame for $\mathbb{R}^D$ and let p be a permutation of $\{1, \ldots, N\}$. Then the frame variation of F with respect to p, $\sigma(F, p)$, is

$$\sigma(F, p) = \sum_{i=1}^{N-1} \|f_{p(i)} - f_{p(i+1)}\|.$$

Now that the there exists a method to order the frame elements, the authors in [19] were able to show bounds on the approximation for general frames in Theorem 7.2.5 and for tight frames, which are better for applications, in Theorem 7.2.6.

Theorem 7.2.5 (Benedetto, Powel, and Yılmaz, 2006)**.** *Given the* $\Sigma\Delta$ *Quantization scheme from 7.2.3. Let* $F = \{f_i\}_{i=1}^N$ *be a normalized frame for* $\mathbb{R}^D$ *with frame bounds*

$A = \frac{N}{D} \leq B$ and inverse frame operator $\mathcal{S}^{-1}$, let p be a permutation of $\{1, \ldots, N\}$, let $|u_0| \leq \delta/2$, and let $x \in \mathbb{R}^D$ satisfy $\|x\| \leq (K - 1/2)\delta$. Then,

$$\mathcal{E}_q(x) \leq \|\mathcal{S}^{-1}\| \left((\delta/2)\sigma(F, p) + |u_N| + |u_0|\right).$$

Theorem 7.2.6 (Benedetto, Powel, and Yılmaz, 2006)**.** *Given the $\Sigma\Delta$ Quantization scheme from 7.2.3. Let $F = \{f_i\}_{i=1}^N$ be a normalized tight frame for $\mathbb{R}^D$ with frame bounds $A = \frac{N}{D} \leq B$, let p be a permutation of $\{1, \ldots, N\}$, let $|u_0| \leq \delta/2$ and let $x \in \mathbb{R}^D$ satisfy $\|x\| \leq (K - 1/2)\delta$, then*

$$\mathcal{E}_q(x) \leq \frac{D}{N}\left(\frac{\delta D}{2N}(\sigma(F, p) + 2)\right).$$

If $\sum_{i=1}^N f_i = 0$ and $u_0 = 0$ then,

$$\mathcal{E}_q(x) \leq \begin{cases} \frac{\delta D}{2N}\sigma(F, p) & \text{if } N \text{ is even,} \\ \frac{\delta D}{2N}(\sigma(F, p) + 1) & \text{if } N \text{ is odd.} \end{cases}$$

We discussed here linear reconstruction but there are non-linear reconstruction methods that can be explored as well, see, [74]. These, in the future, might prove to work even better with nonlinear data sets such as Hyperspectral. We also limited the discussion to first order methods but it is also possible to consider higher order schemes [18].

7.2.3 Frames and $\Sigma\Delta$ Quantization Applied to the Nyström Method

Now that we described the basic properties of frames and $\Sigma\Delta$ quantization we will discuss their application to hyperspectral data, graph Laplacians and Nyström extensions.

First, in relation to hyperspectral data, frames can be used, much like endmembers, to represent pixels as linear combination of frame vectors as suggested in [77]. Building the set of frame vectors can be accomplished either by random sampling or greedy methods, if the frame elements are coming from the data itself. Random methods, much like those discussed in the landmark selection problem could either be computed truly randomly or via assigning a density function on the data. Greedy methods could also accomplish this by building the frame library by using selection methods such as the Sparse Greedy Matrix Approximation algorithm or the Max-Min algorithm. These algorithms would thus try to iteratively choose frame members outside of the span of the current frame members. If we desire frame elements which are not in the data, then a more traditional endmember extraction algorithm could be used. Once a collection of frame elements has been chosen it is simple to check if a collection of vectors, $\{f_i\}_{i=1}^{M}$, form a tight frame in $\mathbb{R}^D$, with lower frame bound A, by verifying

$$\mathcal{S} = AI_D,$$

where I_D is the $D \times D$ identity matrix. Applying a level of sparsity is also advisable when building the frame as we would want the reconstruction coefficients to be concentrated amongst a few members of the frame.

We may also consider creating a dictionary, in essence a frame, which minimizes the $\mathcal{E}_q$ iteratively. We can start with an initial frame, F_0, which is created at random from the data points. Now once a quantization has been performed, those data points which have a high error can be added to the F_0 to create F_1;

the quantization error would now be 0 on these high error points. The frame will now be reordered such that its frame variation is minimized, a second quantization pass now may occur. We know that this second round of quantization will produce lower (at worst the same) $\mathcal{E}_q$ for the data. The number of vectors in the frame, the number of iterations in the frame building procedure, or the maximum allowable error in each pixels quantized representation, can be used as a stopping procedure in the algorithm.

If we consider the kernel matrix associated with our dimension reduction schemes now, we have A_L which are affinities expressed between landmark points and B which are the affinities expressed between landmark and non landmark points. The construction of A_L and B are not only dependent on the selection of the landmark set L but also on the initial graph representation of the data. As the graph is sparse, not all landmarks interact directly with a specific non landmark point. If we view our landmark set L, arising from a frame then the affinities expressed in B can be thought of as frame representations of the non landmark points. The weights which were set in the graph construction phase of the LE or SE algorithm can thus be viewed as frame coefficients.

The selection of landmarks, as discussed in this chapter is important for designing an efficient Nyström extension which minimizes $\mathcal{E}_N$. In [123] the authors use vector quantization to choose a set of landmark points. Vector Quantization operates similar to the k-means clustering landmark selection method. We propose here a similar method using $\Sigma\Delta$ Quantization over Vector Quantization. The landmark set would essentially be chosen in a manner so as to minimize the quantization error

produced. As with the coding of hyperspectral data discussed above, this would have to be done iteratively with a random seeding. If we realize a landmark set which minimizes $\mathcal{E}_q$ then the non landmarks would be represented well by the landmarks.

We can also use Quantization to develop weights for the graph. Given a frame on our data set, we can choose the the nearest neighbors from the collection of frame elements. The nearest neighbor would be chosen as a subset of the frame elements which produce the smallest $\mathcal{E}_q$.

Bibliography

[1] P. Arias, G. Randall, and G. Sapiro. Connecting the out-of-sample and pre-image problems in kernel methods. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[3] C. M. Bachmann, T. L. Ainsworth, and R. A. Fusina. Exploiting manifold geometry in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):441–454, 2005.

[4] R. G. Baraniuk and M. B. Wakin. Random projections of smooth manifolds. *Foundations of Computational Mathematics*, 9(1):51–77, 2009.

[5] B. Basener, E. J. Ientilucci, and D. W. Messinger. Anomaly detection using topology. In *Defense and Security Symposium*, pages 65650J–65650J. International Society for Optics and Photonics, 2007.

[6] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[7] M. Belabbas and P. Wolfe. On landmark selection and sampling in high-dimensional data analysis. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4295–4312, 2009.

[8] M.-A. Belabbas and P. J. Wolfe. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, 106(2):369–374, 2009.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 1396:1373–1396, 2003.

[10] M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. In *Learning theory*, pages 486–500. Springer, 2005.

[11] M. Belkin and P. Niyogi. Convergence of Laplacian eigenmaps. *Advances in Neural Information Processing Systems*, 19:129, 2007.

[12] R. E. Bellman. *Adaptive Control Processes: A Guided Tour*, volume 4. Princeton University Press, 1961.

[13] J. J. Benedetto, W. Czaja, J. Dobrosotskaya, T. Doster, K. Duke, and D. Gillis. Integration of heterogeneous data for classification in hyperspectral satellite imagery. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVIII, Proc. SPIE,*, volume 8390, pages 8390–78. International Society for Optics and Photonics, 2012.

[14] J. J. Benedetto, W. Czaja, J. Dobrosotskaya, T. Doster, K. Duke, and D. Gillis. Semi-supervised learning of heterogeneous data in remote sensing imagery. In *Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering X, Proc. SPIE*, volume 8401, pages 8401–03. International Society for Optics and Photonics, 2012.

[15] J. J. Benedetto, W. Czaja, and M. Ehler. Wavelet packets for time frequency analysis of multispectral data. *International Journal on Geomathematics*, 4(2):137–154, 2013.

[16] J. J. Benedetto, W. Czaja, M. Ehler, C. Flake, and M. Hirn. Wavelet packets for multi- and hyper-spectral imagery. *Wavelet Applications in Industrial Processing VII, Proc. SPIE*, 7535:7535–08, 2010.

[17] J. J. Benedetto and M. W. Frazier, editors. *Wavelets: Mathematics and Applications*, volume 13. CRC press, 1993.

[18] J. J. Benedetto, A. M. Powell, and Ö. Yılmaz. Second-order Sigma-Delta ($\Sigma\Delta$) quantization of finite frame expansions. *Applied and Computational Harmonic Analysis*, 20(1):126–148, 2006.

[19] J. J. Benedetto, A. M. Powell, and Ö. Yılmaz. Sigma-Delta ($\Sigma\Delta$) quantization and finite frames. *IEEE Transactions on Information Theory*, 52(5):1990–2005, 2006.

[20] Y. Bengio, J.-F. Paiement, and P. Vincent. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, pages 177–184. MIT Press, 2003.

[21] A. Berk, G. Anderson, P. Acharya, J. Chetwynd, L. Bernstein, E. P. Shettle, M. W. Matthew, and S. Adler-Golden. *MODTRAN4 User's Guide*. Spectral Sciences, Inc. and U.S. Air Force Research Laboratory, April 2000.

[22] K. Bernard, Y. Tarabalka, J. Angulo, J. Chanussot, and J. A. Benediktsson. Spectral–spatial classification of hyperspectral data based on a stochastic minimum spanning forest approach. *IEEE Transactions on Image Processing*, 21(4):2008–2021, 2012.

[23] L. Bernstein, S. Adler-Golden, R. Sundberg, R. Levine, T. Perkins, A. Berk, A. Ratkowski, G. Felde, and M. Hoke. A new method for atmospheric correction and aerosol optical property retrieval for VIS-SWIR multi- and hyperspectral imaging sensors: QUAC (QUick atmospheric correction). In *IEEE International Geoscience and Remote Sensing Symposium Proceedings*, 2005.

[24] T. Bıyıkoglu, J. Leydold, and P. F. Stadler. Laplacian eigenvectors of graphs. *Lecture notes in mathematics*, 1915, 2007.

[25] J. W. Boardman. Geometric mixture analysis of imaging spectrometry data. In *Geoscience and Remote Sensing Symposium, 1994. IGARSS'94. Surface and Atmospheric Remote Sensing: Technologies, Data Analysis and Interpretation., International*, volume 4, pages 2369–2371. IEEE, 1994.

[26] A. Brook, E. Ben-Dor, and R. Richter. Fusion of hyperspectral images and LIDAR data for civil engineering structure monitoring. In *2nd Workshop Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–5, June 2010.

[27] B. V. Brower and C. A. Laben. Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening, Jan 2000. US Patent 6,011,875.

[28] L. Bruzzone and L. Carlin. A multilevel context-based system for classification of very high spatial resolution images. *IEEE Transactions on Geoscience and Remote Sensing,*, 44(9):2587–2600, 2006.

[29] J. B. Campbell. *Introduction to Remote Sensing*. Guilford Press, 2nd edition, 1996.

[30] G. Camps-Valls, T. V. Bandos Marsheva, and D. Zhou. Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3044–3054, 2007.

[31] G. Camps-Valls, L. Gómez-Chova, M.-M. Jordi, J. Vila-Francés, and J. Calpe-Maravilla. Composite kernels for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 3(1):93–97, 2006.

[32] G. Camps-Valls, L. Gómez-Chova, J. Muñoz-Marí, J. L. Rojo-Álvarez, and M. Martínez-Ramón. Kernel-based framework for multitemporal and multisource remote sensing data classification and change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 46(6):1822–1835, 2008.

[33] A. Castrodad, T. Khuon, R. Rand, and G. Sapiro. Sparse modeling for hyperspectral imagery with LiDAR data fusion for subpixel mapping. In *2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 7275–7278. IEEE, 2012.

[34] A. S. Charles, B. A. Olshausen, and C. J. Rozell. Learning sparse codes for hyperspectral imagery. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):963–978, 2011.

[35] J. Chen, H.-R. Fang, and Y. Saad. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *The Journal of Machine Learning Research*, 10:1989–2012, 2009.

[36] A. Cheriyadat and L. M. Bruce. Why principal component analysis is not an appropriate feature extraction method for hyperspectral data. In *Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International*, volume 6, pages 3420–3422. IEEE, 2003.

[37] O. Christensen. *Frames and Bases: An Introductory Course*. Springer, 2008.

[38] F. R. Chung. *Spectral Graph Theory*, volume 92. American Mathematical Society, 1997.

[39] A. Cloninger, W. Czaja, and T. Doster. A case study on data fusion with overlapping segments. In *Applied Imagery Pattern Recognition Workshop: Sensing for Control and Augmentation, 2013 IEEE (AIPR*, pages 1–11. IEEE, 2013.

[40] A. Cloninger, W. Czaja, and T. Doster. The pre-image problem for laplacian eigenmaps utilizing L_1 regularization with applications to data fusion. In preparation, 2014.

[41] R. R. Coifman and M. J. Hirn. Diffusion maps for changing data. *Applied and Computational Harmonic Analysis*, 36(1):79–107, 2014.

[42] R. R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, 2005.

[43] P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

[44] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[45] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, et al. *Introduction to Algorithms*, volume 2. MIT press Cambridge, 2001.

[46] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. 1: Theory*, volume 41. SIAM, 2002.

[47] J. C. Curlander and R. N. MacDonough. *Synthetic Aperture Radar: Systems and Signal Processing*. Monograph, 1991.

[48] W. Czaja and M. Ehler. Schroedinger eigenmaps for the analysis of biomedical data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1274–1280, 2013.

[49] X. Dai and S. Khorram. A feature-based image registration algorithm using improved chain-code representation combined with invariant moments. *Geoscience and Remote Sensing, IEEE Transactions on*, 37(5):2351–2362, 1999.

[50] M. Dalponte, L. Bruzzone, and D. Gianelle. Fusion of hyperspectral and lidar remote sensing data for classification of complex forest areas. *IEEE Transactions on Geoscience and Remote Sensing*, 46(5):1416–1427, 2008.

[51] 2013 IEEE GRSS Data Fusion Contest. `http://www.grss-ieee.org/community/technical-committees/data-fusion/`.

[52] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[53] V. De Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

[54] F. Dell'Acqua, P. Gamba, and A. Ferrari. Exploiting spectral and spatial information for classifying hyperspectral data in urban areas. In *Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International*, volume 1, pages 464–466. IEEE, 2003.

[55] L. Delves and J. Mohamed. *Computational Methods for Integral Equations*. Cambridge University Press, 1988.

[56] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1117–1126. ACM, 2006.

[57] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[58] D. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[59] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[60] Q. Du, N. Raksuntorn, S. Cai, and R. J. Moorhead. Color display for hyperspectral imagery. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(6):1858–1866, 2008.

[61] R. J. Duffin and A. C. Schaeffer. A class of nonharmonic fourier series. *Transactions of the American Mathematical Society*, pages 341–366, 1952.

[62] K. Duke. *A Study of the Relationship between Spectrum and Geometry through Fourier Frames and Laplacian Eigenmaps*. PhD thesis, University of Maryland, College Park, 2012.

[63] C. Elachi. *Spaceborne Radar Remote Sensing: Applications and Techniques*. IEEE Press, 1988.

[64] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. Sveinsson. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 46(11):3804–3814, 2008.

[65] M. Fauvel, J. Chanussot, and J. A. Benediktsson. Kernel principal component analysis for the classification of hyperspectral remote sensing data over urban areas. *EURASIP Journal on Advances in Signal Processing 2009*, 2009.

[66] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. Tilton. Advances in spectral–spatial classification of hyperspectral images. *Proceeding of the IEEE*, 101(3):652–675, 2013.

[67] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264, 2002.

[68] J. C. Flake. *The Multiplicative Zak Transform, Dimension Reduction, and Wavelet Analysis of LIDAR Data*. Ph.d thesis, University of Maryland, College Park, 2010.

[69] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[70] P. Gader, A. Zare, R. Close, J. Aitken, and G. Tuell. MUUFL Gulfport hyperspectral and LIDAR airborne data set. Technical report, University of Florida, Gainesville, FL, 2013.

[71] A. Garzelli and F. Nencini. Interband structure modeling for pan-sharpening of very high-resolution multispectral images. *Information Fusion*, 6(3):213–224, 2005.

[72] D. Gillis and J. Bowles. Hyperspectral image segmentation using spatial-spectral graphs. In *SPIE Defense, Security, and Sensing*, pages 83901Q–83901Q. International Society for Optics and Photonics, 2012.

[73] Y. Goldberg, A. Zakai, D. Kushnir, and Y. Ritov. Manifold learning: The price of normalization. *The Journal of Machine Learning Research*, 9:1909–1939, 2008.

[74] V. K. Goyal, M. Vetterli, and N. T. Thao. Quantized overcomplete expansions in $\mathbb{R}^N$: analysis, synthesis, and algorithms. *IEEE Transactions on Information Theory*, 44(1):16–31, 1998.

[75] A. Halevy. *Extensions of Laplacian Eigenmaps for Manifold Learning*. Ph.d thesis, University of Maryland, College Park, 2011.

[76] C. Heil. *A Basis Theory Primer: Expanded Edition*. Springer, 2010.

[77] M. J. Hirn. *Enumeration of Harmonic Frames and Frame Based Dimension Reduction*. Ph.d thesis, University of Maryland, College Park, 2009.

[78] X. Jia and J. A. Richards. Segmented principal components transformation for efficient hyperspectral remote-sensing image display and classification. *IEEE Transactions on Geoscience and Remote Sensing*, 37(1):538–542, 1999.

[79] N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57, 2002.

[80] D. H. Kim and L. H. Finkel. Hyperspectral image processing using locally linear embedding. In *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, pages 316 – 319, 2003.

[81] D. Knossow, A. Sharma, D. Mateus, and R. Horaud. Inexact matching of large and sparse graphs using Laplacian eigenvectors. In *Graph-Based Representations in Pattern Recognition*, pages 144–153. Springer, 2009.

[82] B. Koetz, F. Morsdorf, S. van der Linden, T. Curt, and B. Allgöwer. Multi-source land cover classification for forest fire management based on imaging spectrometry and LIDAR data. *Forest Ecology and Management*, 256(3):263 – 271, 2008.

[83] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[84] S. Kumar, J. Ghosh, and M. M. Crawford. Best-bases feature extraction algorithms for classification of hyperspectral data. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(7):1368–1379, 2001.

[85] S. Kumar, M. Mohri, and A. Talwalkar. Ensemble Nyström method. In *Neural Information Processing Systems*, volume 7, page 223, 2009.

[86] S. Kumar, M. Mohri, and A. Talwalkar. Sampling techniques for the Nyström method. In *Conference on Artificial Intelligence and Statistics*, pages 304–311, 2009.

[87] J. T. Kwok and I. W. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15(6):1517–25, Nov 2004.

[88] D. Langrebe. Indiana's Indian Pines 1992 data set. `http://dynamo.ecn.purdue.edu/biehl/MultiSpec/documentation.html`, 1992.

[89] J. Le Moigne, N. S. Netanyahu, and R. D. Eastman. *Image Registration for Remote Sensing*. Cambridge University Press, 2011.

[90] J. Le Moigne and I. Zavorine. Use of wavelets for image registration. In *AeroSense 2000*, pages 99–108. International Society for Optics and Photonics, 2000.

[91] Y. LeCun and C. Cortes. The MNIST database of handwritten digits, 1998.

[92] C. Lee and D. A. Landgrebe. Analyzing high-dimensional multispectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 31(4):792–800, 1993.

[93] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[94] M. Lee, L. Bruce, and S. Prasad. Concurrent spatial-spectral band grouping: Providing a spatial context for spectral dimensionality reduction. In *3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4, June 2011.

[95] A. J. Lewis and F. Henderson. Geomorphic and hydrologic applications of active microwave remote sensing. *Principles and applications of imaging radar. Manual of Remote Sensing*, pages 567–629, 1998.

[96] J. Lintz and D. S. Simonett. *Remote Sensing of Environment*. Addison-Wesley, 1976.

[97] L. Ma, M. M. Crawford, and J. Tian. Generalised supervised local tangent space alignment for hyperspectral image classification. *Electronics Letters*, 46(7):497–498, 2010.

[98] M. W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[99] D. Manolakis, D. Marden, and G. Shaw. Hyperspectral image processing for automatic target detection applications. *Lincoln Laboratory Journal*, 14(1):79–116, 2003.

[100] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[101] M. W. Matthew, S. M. Adler-Golden, A. Berk, G. W. Felde, G. P. Anderson, D. Gorodetzky, S. E. Paswaters, and M. Shippert. Atmospheric correction of spectral imagery: Evaluation of the FLAASH algorithm with AVIRIS data. In *AeroSense 2003*, pages 474–482. International Society for Optics and Photonics, 2003.

[102] J. E. Mesina. *Urban Classification Techniques Using the Fusion of LIDAR and Spectral Data*. PhD thesis, Monterey, California. Naval Postgraduate School, 2012.

[103] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel pca and de-noising in feature spaces. In *NIPS*, volume 11, pages 536–542, 1998.

[104] A. Mohan, G. Sapiro, and E. Bosch. Spatially coherent nonlinear dimensionality reduction and segmentation of hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 4:206–210, 2007.

[105] J. M. Nascimento and J. M. Bioucas Dias. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(4):898–910, 2005.

[106] J. M. Norman and F. Becker. Terminology in thermal infrared remote sensing of natural surfaces. *Agricultural and Forest Meteorology*, 77(3):153–166, 1995.

[107] J. Nunez, X. Otazu, O. Fors, A. Prades, V. Pala, and R. Arbiol. Multiresolution-based image fusion with additive wavelet decomposition. *IEEE Transactions on Geoscience and Remote Sensing*, 37(3):1204–1211, 1999.

[108] E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.

[109] B. M. Oliver, J. R. Pierce, and C. E. Shannon. The philosophy of PCM. *Proceedings of the IRE*, 36(11):1324–1331, 1948.

[110] K. Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[111] A. Plaza, P. Martinez, R. Pérez, and J. Plaza. Spatial/spectral endmember extraction by multidimensional morphological operations. *IEEE Transactions on Geoscience and Remote Sensing*, 40(9):2025–2041, 2002.

[112] V. Rajapakse, W. Czaja, Y. Pommier, W. Reinhold, and S. Varma. Predicting expression-related features of chromosomal domain organization with network-structured analysis of gene expression and chromosomal location. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, BCB '12, pages 226–233, New York, NY, USA, 2012. ACM.

[113] I. S. Reed and X. Yu. Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distribution. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(10):1760–1770, 1990.

[114] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[115] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*, volume 158. SIAM, 1992.

[116] A. Schaum and A. Stocker. Hyperspectral change detection and supervised matched filtering based on covariance equalization. In *Proceedings of SPIE*, volume 5425, pages 77–90, 2004.

[117] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks,*, 10(5):1000–1017, 1999.

[118] B. Scholkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Advances in kernel methods-support vector learning*, 1999.

[119] J. Schott. *Remote Sensing: The Image Chain Approach*. Oxford University Press, New York, 1997.

[120] D. W. Scott and J. R. Thompson. Probability density estimation in higher dimensions. In *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, volume 528, pages 173–179, 1983.

[121] M. Shimoni, G. Tolt, C. Perneel, and J. Ahlberg. Detection of vehicles in shadow areas using combined hyperspectral and LIDAR data. In *2011 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4427–4430, 2011.

[122] A. Smola and B. Scholkopf. Sparse greedy matrix approximation for machine learning. In *International Conference on Machine Learning*, pages 911–918, 2000.

[123] W. Sun, A. Halevy, J. J. Benedetto, W. Czaja, C. Liu, H. Wu, B. Shi, and W. Li. UL-Isomap based nonlinear dimensionality reduction for hyperspectral imagery classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 89:25–36, 2014.

[124] P. H. Swain and S. M. Davis. *Remote Sensing: The Quantitative Approach*. McGraw-Hill, 1978.

[125] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot. Spectral-spatial classification of hyperspectral imagery based on partitional clustering techniques. *IEEE Transactions on Geoscience and Remote Sensing*, 47:2973–2987, August 2009.

[126] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot. Spectral-spatial classification of hyperspectral imagery based on partitional clustering techniques. *IEEE Transactions on Geoscience and Remote Sensing*, 47(8):2973–2987, 2009.

[127] Y. Tarabalka, J. Tilton, J. A. Benediktsson, and J. Chanussot. A marker-based approach for the automated selection of a single segmentation from a hierarchical set of image segmentations. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(1):262–272, 2012.

[128] D. M. Tax and R. P. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[129] P. Thenkabail, R. Smith, and E. De Pauw. Hyperspectral vegetation indices and their relationships with agricultural crop characteristics. *Remote sensing of Environment*, 71(2):158–182, 2000.

[130] P. Thévenaz and M. Unser. Optimization of mutual information for multiresolution image registration. *IEEE Transactions on Image Processing,*, 9(12):2083–2099, 2000.

[131] J. Tilton, Y. Tarabalka, P. Montesano, and E. Gofman. Best merge region growing with integrated region object classification. *IEEE Transactions on Geoscience and Remote Sensing*, 50:4454–4467, 2012.

[132] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[133] US Army Topographic Engineering Center. `http://www.tec.army.mil/Hypercube`.

[134] F. Van Der Meer. Analysis of spectral absorption features in hyperspectral imagery. *International Journal of Applied Earth Observation and Geoinformation*, 5(1):55–68, 2004.

[135] D. Widemann. *Dimensionality reduction for hyperspectral data*. Ph.d thesis, University of Maryland, College Park, 2008.

[136] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, pages 682–688, 2001.

[137] M. E. Winter. N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 266–275. International Society for Optics and Photonics, 1999.

[138] K. Zhang and J. T. Kwok. Density-weighted Nyström method for computing large kernel eigensystems. *Neural Computation*, 21(1):121–146, 2009.

[139] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved Nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.

[140] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26:313–338, 2002.

[141] A. K. Ziemann, D. W. Messinger, and J. A. Albano. Target detection performed on manifold approximations recovered from hyperspectral data. In *SPIE Defense, Security, and Sensing*, pages 874319–874319. International Society for Optics and Photonics, 2013.