ABSTRACT

Title of dissertation: Dimensionality Reduction for Hyperspectral Data

David P. Widemann, Doctor of Philosophy, 2008

Dissertation directed by: Professor John Benedetto
Department of Mathematics

Professor Wojciech Czaja
Department of Mathematics

This thesis is about dimensionality reduction for hyperspectral data. Special emphasis is given to dimensionality reduction techniques known as kernel eigenmap methods and manifold learning algorithms. Kernel eigenmap methods require a nearest neighbor or a radius parameter be set. A new algorithm that does not require these neighborhood parameters is given. Most kernel eigenmap methods use the eigenvectors of the kernel as coordinates for the data. An algorithm that uses the frame potential along with subspace frames to create nonorthogonal coordinates is given. The algorithms are demonstrated on hyperspectral data. The last two chapters include analysis of representation systems for LIDAR data and motion blur estimation, respectively.

Dimensionality Reduction for Hyperspectral Data

by

David P. Widemann

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
2008

Advisory Committee:
Professor John Benedetto, Chair/Advisor
Professor Wojciech Czaja, Co-Advisor
Professor Kasso Okoudjou
Professor Konstantina Trivisa
Professor Denny Gulick

For Mariam and Tarahneh

Acknowledgments

I would like to thank my advisor, John Benedetto, not only for teaching me harmonic analysis but also giving me the opportunity to work on two interesting, applied harmonic analysis projects. John's guidance and support were integral to my graduate school experience.

I would also like to thank my co-advisor, Wojciech Czaja, the associate director of the Norbert Wiener Center for Harmonic Analysis and its Applications, Ioannis Konstantinidis, and Diego Maldonado for our many mathematical conversations.

Thanks are due to Professors Kasso Okoudjou, Konstantina Trivisa, and Denny Gulick for serving on my committee.

I would like to thank my fellow coders at the Norbert Wiener Center, Chris Flake, Matt Hirn, and Chris Miller. Sharing a laboratory with these guys has been an experience.

Thanks are also given to: Juan Romero for introducing me to John and being a friend, Ji Hui for being a friend and quite clever, Prashant Athavale and I-kuen Chen for their friendship, support, and conversation.

I would like to acknowledge financial support from the Department of Defense for the hyperspectral and LIDAR projects discussed herein.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AIS	Airborne Imaging Spectrometer
DCT	Discrete Cosine Transform
DEM	Data Elevation Matrix
DWT	Discrete Wavelet Transform
HLLE	Hessian Locally Linear Embedding
HYDICE	Hyperspectral Digital Imagery Collection Experiment
LIDAR	Light Detection and Ranging
LLE	Locally Linear Embedding
MDS	Multidimensional Scaling
NGA	National Geospatial-Intelligence Agency
NWC	Norbert Wiener Center for Harmonic Analysis and Applications
PCA	Principal Component Analysis
SNR	Signal to Noise Ratio
SSIM	Structural Similarity Index
SVD	Singular Value Decomposition
TSSIM	Terrain Structural Similarity Index

Chapter 1

Introduction

1.1 Problem overview

In many applications of machine learning, data mining, and image processing, high dimensional data is collected. Although the collected data is high dimensional, often it is the case that the data is intrinsically low dimensional. We think of this data as lying in a subspace or a manifold[1] embedded in the larger space. Dimensionality reduction is the transformation of this data lying in a high dimensional space to a low dimensional space. Hopefully, the transformation creates a representation of the data that is low dimensional yet still preserves certain properties of the original data. Often in dimensionality reduction, the goal is to find the minimum number of parameters (dimensions) necessary to classify and account for the data.

[1]Formally, a *topological manifold* is a second countable Hausdorff space that is locally homeomorphic to Euclidean space. However, if it helps, just think of a manifold as a space in which every point has a neighborhood that looks like $\mathbb{R}^n$, although globally, the manifold may be curved.

1.2 Examples

1.2.1 Face recognition

As an example of dimensionality reduction, Belkin and Niyogi consider a camera rotating around its subject while simultaneously taking a picture [2]. If each picture has n^2 pixels, then each image can be considered a point in $\mathbb{R}^{n^2}$. In this example, the data collected lies in $\mathbb{R}^{n^2}$ yet intrinsically it has a dimension of one, this being the one parameter necessary to describe the motion of the camera. Ideally, a dimensionality reduction algorithm would be able to take these images as high dimensional input and then represent and classify these images according to their corresponding angle of rotation, i.e., a one dimensional space. This can be seen in Figure 1.1, in which seven distinct rotations of a face are randomly given and then the data is organized by angle of rotation.

Figure 1.1: Capturing the angle of rotation of a face.

1.2.2 Manifolds

Saul and Roweis gave a more geometric explanation of dimensionality reduction using the Swiss roll [41]. Here, we have the Swiss roll, a two dimensional manifold embedded in $\mathbb{R}^3$. This is Figure 1.2A. In practice we do not know the

dimensionality of the manifold. All that we have is our data consisting of samples lying in $\mathbb{R}^3$, Figure 1.2B. Our goal in trying to reduce the dimensionality of the Swiss roll is to transform the data into two dimensions in such a way that the atlases of the manifold are preserved. Figure 1.2C shows how the manifold in $\mathbb{R}^3$ is taken into $\mathbb{R}^2$ in such a way that preserves the neighborhood structure.

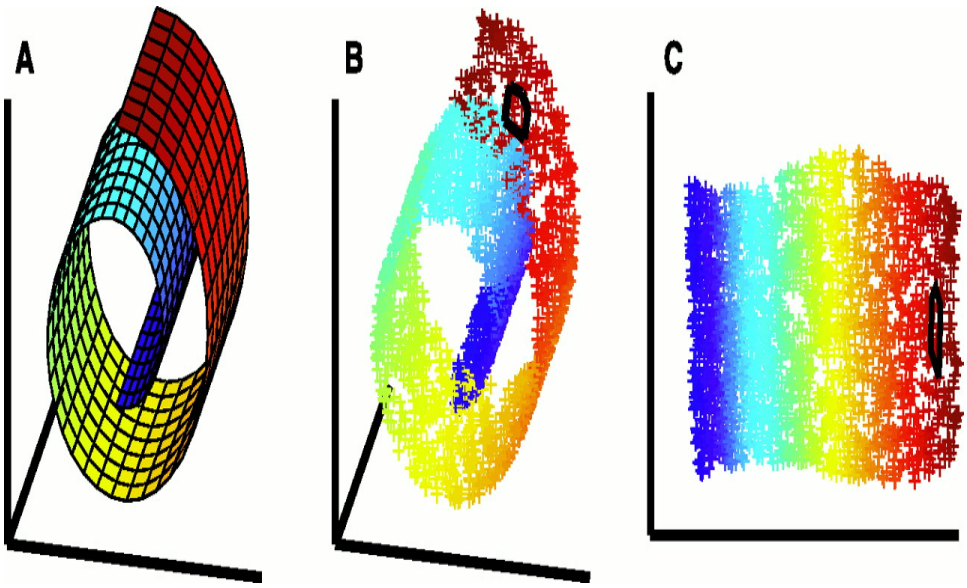


Figure 1.2: An artificial example. Reducing the Swiss roll from three to two dimensions.(Courtesy Saul and Roweis)

1.2.3 Hyperspectral data

Hyperspectral sensors and data will be discussed in more detail later. However, for now, it suffices to consider hyperspectral data as an image stacked on top of itself D times. Here, we use D to represent the spectral dimension of the data. The only difference between each image is the spectrum of light from which data are collected, e.g., for dimension 1 the data may come from light with a wavelength of approximately 400 nano-meters whereas for dimension 200 the data may come from

light with a wavelength of approximately $2,500$ nano-meters. The hyperspectral data forms a data cube which can be seen in Figure 1.3.

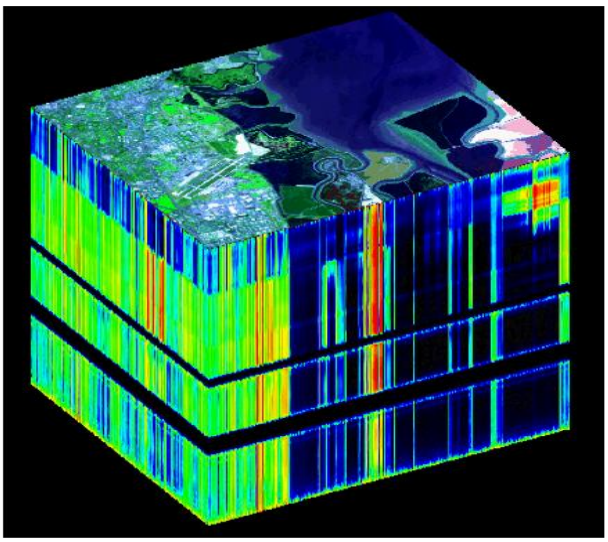


Figure 1.3: A hyperspectral data cube. (Courtesy NEMO Project Office, United States Navy)

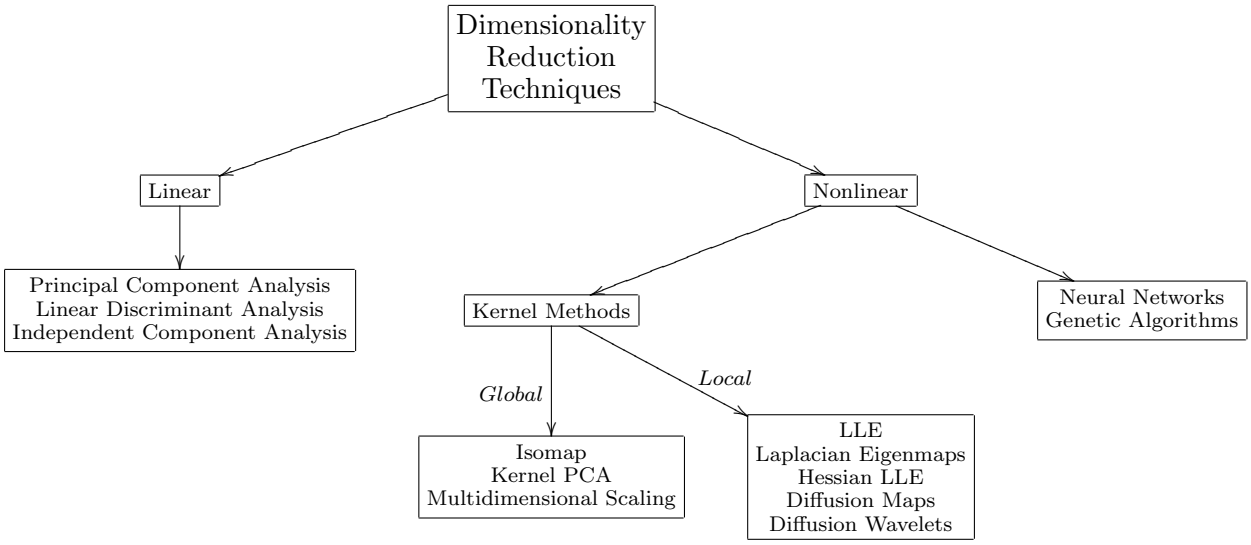
Each pixel in the hyperspectral cube has its own spectral signature. One of the goals of hyperspectral imaging is to identify the materials in the image by looking at the spectral signatures of the pixels. The goal of dimensionality reduction for hyperspectral data is to reduce the number of spectral bands from D to $d \leq D$ and still have spectral signatures that allow for image clustering and classifcation. This is an oversimplification of hyperspectral data and there are a host of problems associated with performing dimensionality reduction on hyperspectral data that will be discussed in more detail later.

1.2.4 Data mining and search engines

Dimensionality reduction is also used for classifying and retrieving documents. The basic idea is that each document corresponds to a vector of length n. Here n is the cardinality of the set of keywords that is being considered. Each entry in a vector is the number of times the keyword occurs in the document, e.g., v_i is the number of times that the i^{th} keyword occurs in the document. Different metrics are used for these vectors; however, one of most common metrics is vector angle. Vector angle computes the angle between the two vectors using the Euclidean inner product. Once the documents are given coordinates in $\mathbb{R}^n$ and the distance between the documents is computed, it becomes possible to store and retrieve documents that are related to each other in an efficient manner.

1.3 Various techniques

Because of the nonlinear nature of the Swiss roll and many other data sets, linear methods for dimensionality reduction such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) perform poorly on this type of data. As a consequence of this, many nonlinear dimensionality reduction methods have been created. A team of researchers at Maastricht University in the Netherlands has done a comparative review of dimensionality reduction techniques and created a MATLAB toolbox that implements twenty-seven different dimensionality reduction algorithms [46]. Dimensionality reduction techniques range from neural networks, genetic algorithms, statiscal approaches to kernel methods.



The kernel dimensionality reduction algorithms can mostly be divided into two categories. There are techniques that preserve the global properties of the manifold and there are techniques that preserve the local properties of the manifold.[2] One subfamily of local dimensionality reduction techniques is known as kernel methods. My work with dimensionality reduction involves using local kernel methods to reduce the dimension of hyperspectral data.

1.4 Mathematical formulation

The problem of dimensionality reduction is as follows. Given n data samples in $\mathbb{R}^D$, x_i for $i = 1, ..., n$, they form a D x n matrix, X. Dimensionality reduction algorithms transform this data, X, to a new set of data, Y. Y is a d x n matrix where $d \leq D$. Column y_i of Y corresponds to the data point x_i which is the i^{th} column of X. Ideally, this transformation would find the intrinsic dimensionality of

[2]For some methods the distinction between being global or local depends on the choice of the kernel or parameters used to create the kernel

the data and preserve the geometry or relevant properties of the manifold.

1.5 My contribution

The main contribution of this thesis is a new method for constructing a graph from the data in chapter 7 and the use of frames instead of eigenmaps for representing the reduced coordinates in chapter 5. Currently, a graph is constructed from the data with a user defined input such as k-nearest neighbors or radius. However, with many types of data, there is no intuitive or natural way to choose number of nearest neighbors or the radius. Example 7.1 demonstrates that constructing an inaccurate graph for the data can lead to erroneous reduced dimension results. The algorithm described in section 7.1 for graph construction relies on the geometry of the data instead of user defined input.

For many kernel eigenmaps methods, the final step is to diagonalize the kernel matrix and take the eigenvectors, called *eigenmaps*, as the new coordinates for the data. In example 11, I give a motivating example suggesting that eigenmaps may not be ideal. If the new coordinates are allowed to be nonorthogonal elements of a frame, then they may outperform eigenmaps. A new algorithm in section 5.3 is described for constructing frame coordinates using the frame potential and subspace frames.

An analysis of several representation systems for LIDAR data is given in chapter 8. Special emphasis placed on directional filter banks. The main conclusion of this work is that contourlets perform better than other representation systems on

urban LIDAR data and that the Terrain Structural Simiilarity Index is a better metric for evaluating image quality than other currently used metrics.

An algorithm that combines ceptstral methods with the Radon tranform for estimating motion blur is given in chapter 9. This method outperforms directional filters and purely cepstral methods for estimating the point spread function that causes blurring.

Chapter 2

Hyperspectral data

2.1 Data collection

The technology for hyperspectral sensors was first developed in the 1970s. NASA's Jet Propulsion Laboratory demonstrated one of the first examples of an airborne hyperspectral sensor in 1982 [28]. One of the main applications of hyperspectral sensors is to remotely identify materials from their spectral signatures. Many scientific, commercial, and military groups are interested in hyperspectral data and have created their own tools for collecting and analyzing the data.

A normal color image records the intensity of light in three spectral bands or channels, red, green, and blue. A hyperspectral image is similar except that many bands are collected across a wider range of the light spectrum. The spectral range for a hyperspectral sensor usually starts at a wavelength of 400 nanometers and can extend beyond the limits of human perception, 700 nanometers, all way up to the near-infrared region with wavelength of $2,500$ nanometers. Figure 2.1 demonstrates the colors and wavelengths for visible light. It should be noted that the intensity of light is not uniform; the intensity is at a maximum in the green bands and smoothly decreases away from the maximum.

Other important parameters for a hyperspectral sensor include: the altitude of the sensor, the number of bands, the ground pixel size, and the ground swath. The

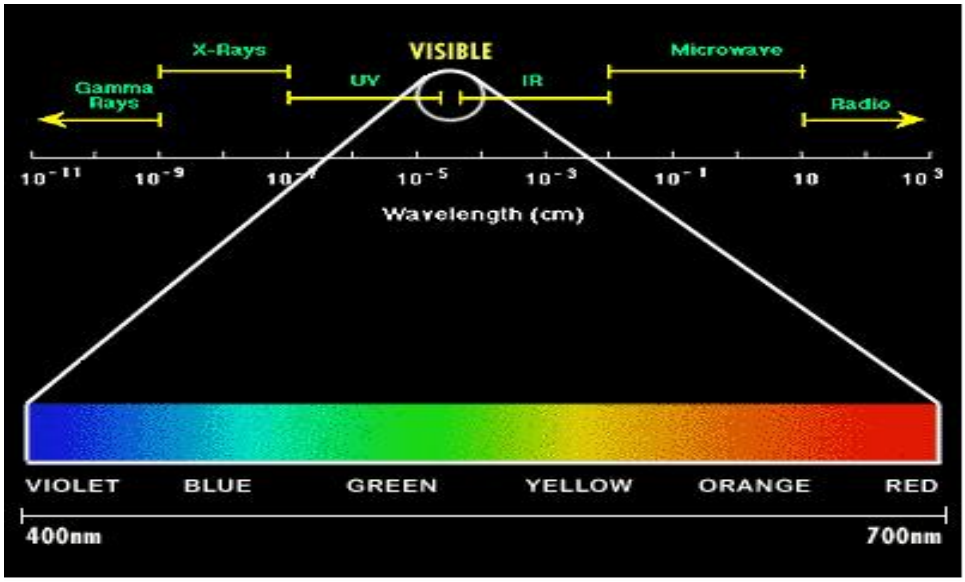


Figure 2.1: The visible region of the light spectrum.

ground pixel size gives the resolution of the sensor and the ground swath describes the typical size of the region being sensed perpendicular to direction of motion. Table 2.1 gives some examples of hyperspectral sensors and their corresponding parameters.

Table 2.1: Examples of hyperspectral sensors

Sensor	Altitude	Spectral Range (wavelength)	Number of bands	Ground pixel size	Ground swath
AIS	4 km	1,200-2,400 nm	128	8 m	0.3 km
HYDICE	6 km	400-2,500 nm	210	3 m	1 km
Hyperion	705 km	400-2,500 nm	200	30 m	7.5 km

One can see from the resolution size of the data that different materials probably do exist within the same pixel of the image. This gives rise to pixels having a mixed spectral signature, i.e., the collected spectral signature is mixture of the "pure" spectral signatures on the ground. This will be discussed in more detail

later. There are other problems associated with data collection such as upwelling and downwelling. The light that is collected at the sensor may have been reflected from the atmosphere and never reached the ground. Or the light that is collected at the sensor may have been reflected several times from objects that may be outside the resolution area.

There are two type of hyperspectral data, radiance and reflectance. The type of data that has just been described is radiance data. For reflectance data, the effects of the atmosphere are removed. The atmospheric effects are modelled as a filter that is convolved with the spectral signatures. The goal then becomes to deconvolve the radiance data and approximate the "true" reflected spectral signatures.

2.2 Working with hyperspectral data

Sometimes when hyperspectral data is collected certain bands are unusable. The bands may be unusable because water in the atmosphere absorbed too much radiation at a certain wavelength or for various other reasons. One of the first steps in working with hyperspectral data is to either remove or fill-in values for these "bad" bands. In our work, we have removed the bad bands. Also, to apply kernel methods it is necessary to reshape the datacube, $n \times m \times D$, where the first two coordinates are the pixel location and the last coordinate is the spectral response. Reshaping is done by collapsing each column, 2nd coordinate, below the previous column. Thus, a $n \times m$ matrix becomes a $nm \times 1$ vector. Only in our case, each entry in the vector is the spectral signature of a pixel, thus we have a $nm \times D$

matrix. Many people like to think of the spectral signature of a pixel as a column vector, so the transpose of this matrix is taken creating a matrix of size $D \times N$ where $N = nm$.

2.2.1 Endmember dectection as a type dimensionality reduction

There are various notions of what it means to be an endmember. One notion is that an endmember is the spectral signature of a "pure" material, i.e., an object that does not consist of materials giving different spectral signatures. Using this notion, an endmember is often measured under laboratory conditions. This is not what we mean by endmember. For us, an endmember of a datacube is the spectral signature of a pixel that is used to build other spectral signatures in the datacube, i.e., each pixel in the datacube is represented as a linear combination of endmembers. In reality, most spectral signatures in a datacube cannot be reconstructed perfectly as a linear combination of endmembers because of noise. There is a relationship between the amount of noise and how well endmembers can be used to reconstruct spectral signatures. The data, X, a $D \times N$ matrix, is factored approximately as the product of two matrices, $X \approx EA$ [37]. Here E is a $D \times k$ endmember matrix where each column is the spectral signature of a pixel taken from the image, k is the number of endmembers. A is a $k \times N$ abundance matrix. Each pixel in the datacube, i.e., column of X, is represented a linear combination of endmembers

with some physical constraints,

$$X_j \approx EA_j$$
$$\sum_{i=1}^{k} A_{ij} = 1 \ \forall j = 1 \dots N,$$
$$A_{ij} \geq 0 \ \forall i = 1 \dots k, \ j = 1 \dots N.$$

The idea here is that no endmember can contribute negatively to a spectral signature; this makes physical sense. The percentage that the i^{th} endmember contributes to the j^{th} pixel's signature is A_{ij}, a number between zero and one. This is why A is referred to as the abundance map; it gives the abundance of the materials at each pixel. In this formulation, we would like endmembers to correspond to a pure, identifiable materials. Also, in order for this model to be of use in dimentionality reduction, the total number of endmembers, k should be less than D. The new coordinates for the data is the abundance matrix A, $X_j \longmapsto A_j$ for all $j = 1 \dots N$.

The endmember approach above has the added benefit of using distortion as a metric. The distortion between the datacube, X, and its approximation, EA, is $\|X - EA\|_2$. Letting the number of endmembers vary allows a distortion curve to be generated. Using this curve, it is possible to quantitatively measure how well the datacube is approximated. The next section demonstrates why it is difficult to determine how well dimensionality reduction algorithms perform on hyperspectral data. Since dimensionality reduction algorithms do not reconstruct the data, it does not make sense to use disortion as a metric.

Figure 2.2: Color image of the urban data set (Courtesy Robert Pazak, U.S. Army Corps of Engineers.)

2.3 Our data, urban HYDICE sensor imagery

Our datacube comes from Hyperspectral Digital Imagery Collection Experiment (HYDICE). The urban data set is 16 bit, reflectance data of an urban terrain with 307×307 pixels and 210 bands. The urban data set is one of the standard nonartificial data sets used for testing dimensionality reduction algorithms. A color image of the datacube is show in Figure 2.2.

As can be seen from the color image of the data there are classes in the datacube, such as trees, grass, asphalt, soil and roof tops. In Figure 2.3 the spectral signature of a pixel from each of these classes is shown.

The gaps in the curves correspond to unusable bands. One interesting aspect of these spectral curves is that most of the variation occurs beyond 700 nm, in the infrared region beyond human perception.

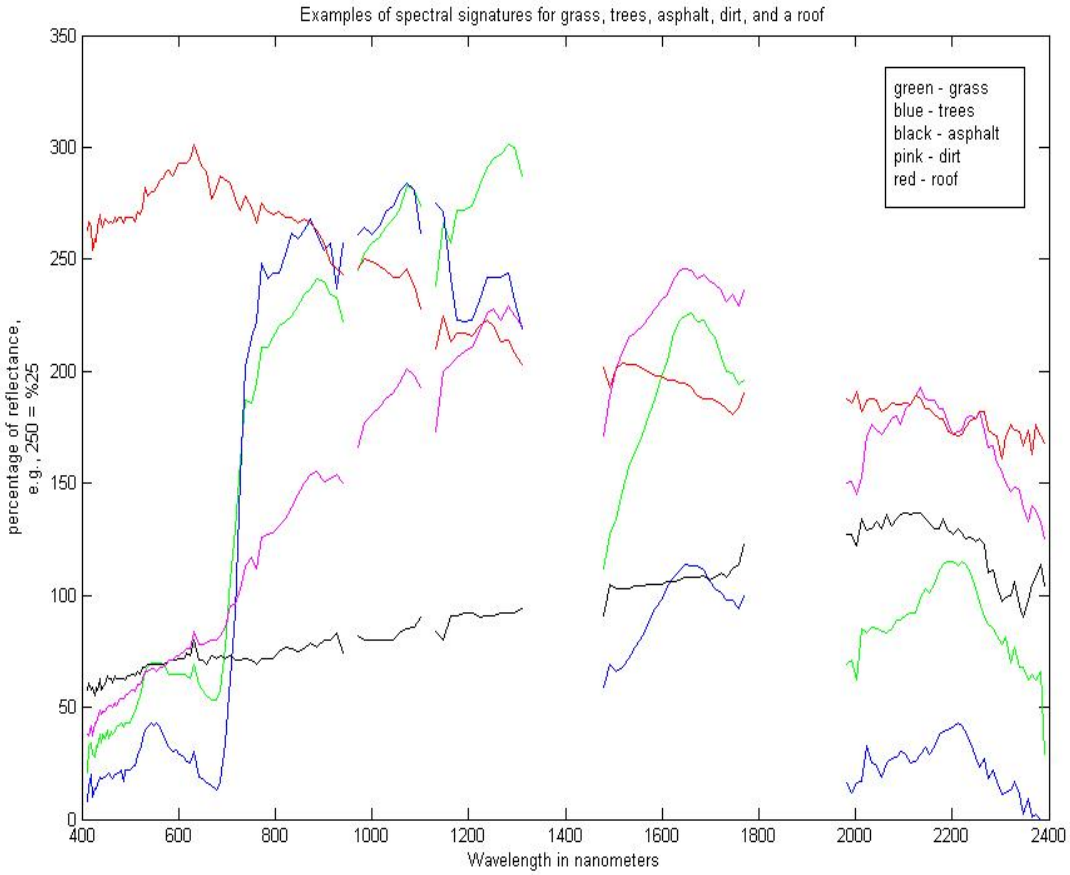


Figure 2.3: Examples of spectral signatures for different materials

My team at the NWC has worked with National Geospatial-Intelligence Agency (NGA) on testing various algorithms using the urban terrain data set. Image Analysts for the NGA identified twenty-three classes in the datacube. This count did not include outliers. The classes included trees, asphalt, soil, grass vegetation, and a number of roof tops. In Figure 2.4 the soil and grass vegetation class are displayed. The sharp rise in the spectral curve for grass vegetation is characteristic of vegetation. I have been told that this comes from chlorophyll absorbtion.

One of the goals of dimensionality reduction for hyperspectral data is to pre-

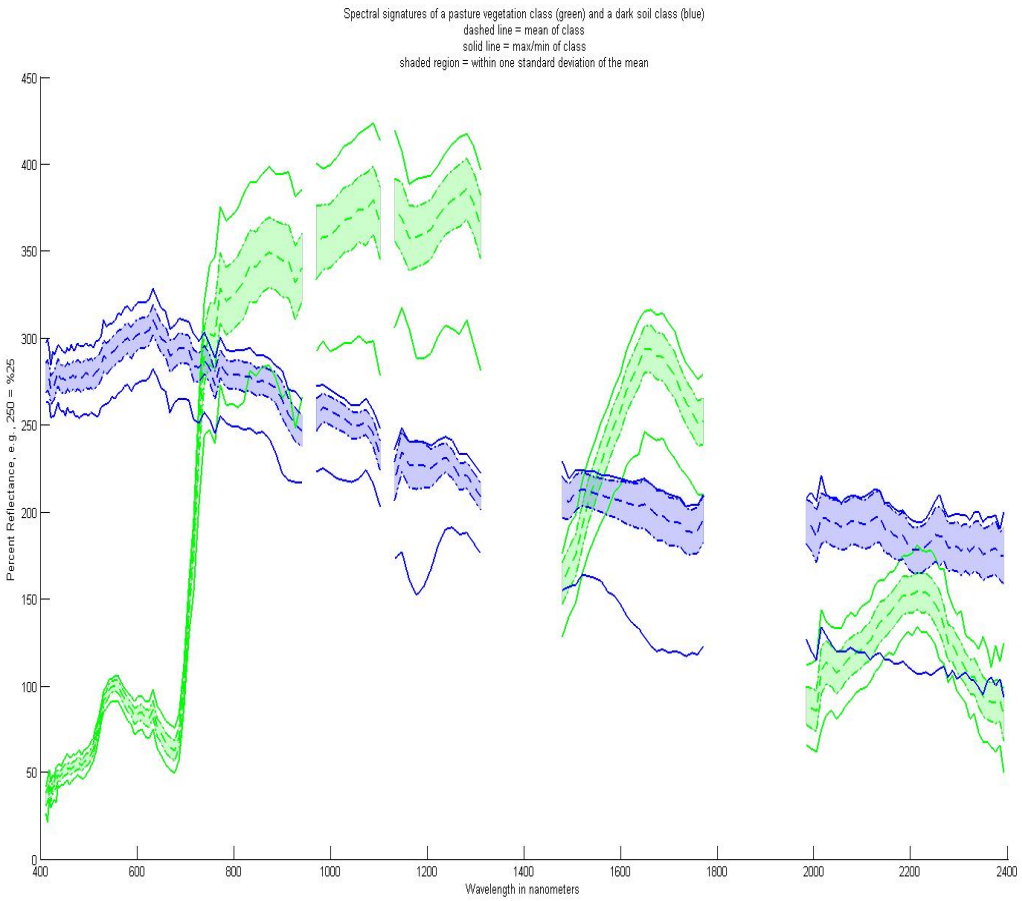


Figure 2.4: Two example classes of data (Courtesy Jesse Sugar-Moore)

serve pixels that are in the same class while separating them from pixels of a differing class. This means that pixels x_i and x_j are in class C if and only if y_i and y_j are in class C. Accomplishing this class preservation and separation can be difficult. As can be seen from Figure 2.5 and Figure 2.6 many classes have considerable spectral overlap and for some classes the variance in the spectral signature is large.

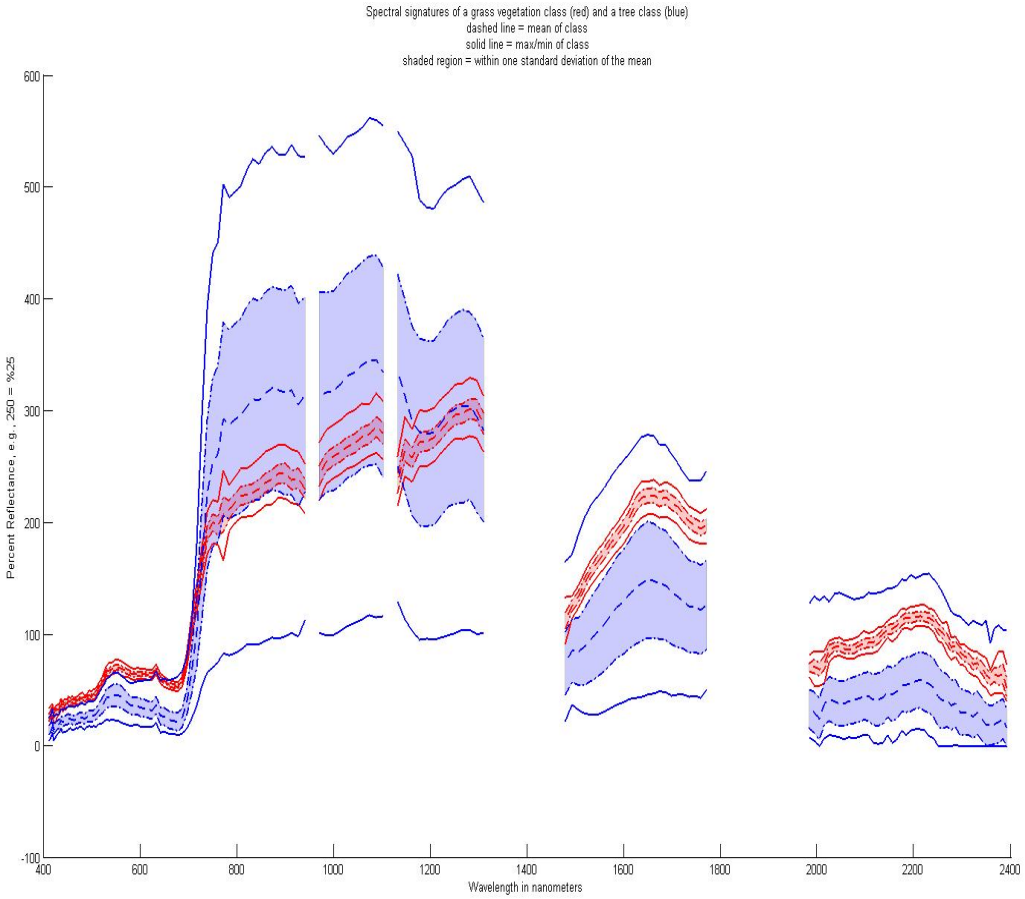


Figure 2.5: The grass vegetation class and tree class (Courtesy Jesse Sugar-Moore)

2.4 Classification

There are many techniques that attempt to determine the classes within a datacube and whether two pixels are in the same class or not. Vector angle and mean distance are two of the simplest classifiers. The idea behind vector angle is as follows. Fix a pixel, p, in the datacube and an angle, θ. S_p stands for the spectral signature of p. A pixel, q, in the datacube is in the same class as p if the angle between their spectral signatures is less than θ. Equation 2.1 gives the vector angle

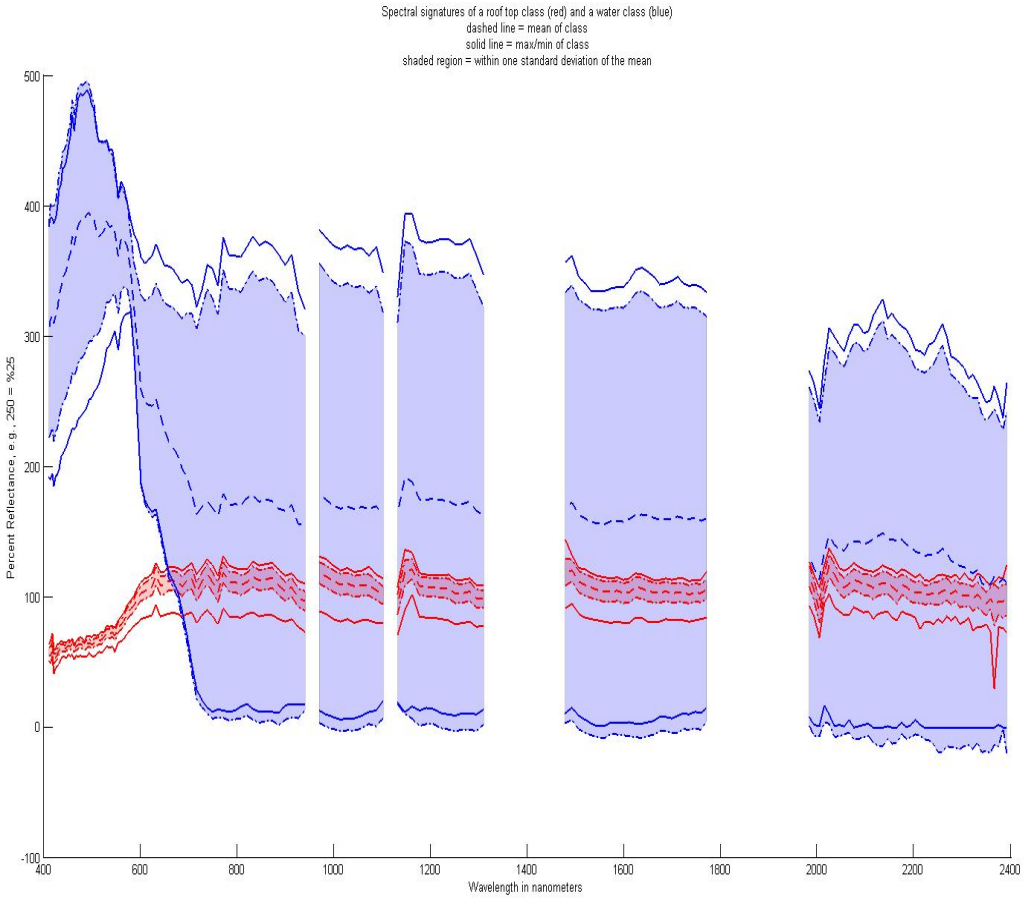


Figure 2.6: A class with small variance and a class with large variance (Courtesy Jesse Sugar-Moore)

between p and q.

$$\varphi = arccos\left(\frac{S_p \cdot S_q}{\|S_p\|_2 \|S_q\|_2}\right) \tag{2.1}$$

The mean distance classifier is computed by fixing p and a radius r. A pixel q in the datacube is in the same class a p if $\|S_p - S_q\|_2 < r$.

As can be seen in the figures, many classes have similar spectral signatures and some classes have a large variance of spectral signatures. This makes using vector angle and mean distance classifiers problematic. Both of these methods depend heavily on the initial pixel chosen and the angle or radius being used. Often there

are false negatives and false positives because of the parameters and the nature of the class.

The problem of dimensionality reduction for hyperspectral data is as follows. Given a datcube of size $n \times m \times D$ with k classes. The goal is to transform the datacube, $X \mapsto T(X)$, into a datacube of size $n \times m \times d$ where d is as smallest possible spectral dimension that preserves the class structure. By preserving the class structure, one means that if a pixel, p, in the original datacube is determined, by whichever metric used, to be a mixture of classes $C_{p_1}, \ldots, C_{p_m}$ then its corresponding pixel in the transformed datacube, $T(p)$, should have the same mixture for the transformed classes. In practice, it is difficult to implement this notion to determine whether the data was sucessfully reduced. There are infintely many metrics that can be used to determine classes, many of the classes themselves are dependent on the initial pixel chosen to represent the class. In theory, one would have to test the data using all metrics and classes generated by pixels in the datacube. This is not feasible. The numerical results contained in chapter 6, explain our approach at measuring the success of each dimensionality reduction algorithm.

Chapter 3

Operators on graphs

In order to cover the basics of kernel eigenmap methods, it is first necessary to understand a few facts about graphs and operators on graphs. Most of what is presented here is comes from Fan Chung's book on "Spectral Graph Theory " [15].

3.1 Graphs

Definition 1 (graph)**.** *A graph G is an ordered pair of sets, G = (V,E), where V is the set of vertices or nodes and E is a set of pairs of vertices called edges.*

If x and y are nodes in the graph, G, then an edge from x to y is denoted by (x, y). If there is an edge between x and y, then x and y are said to be neighbors or adjacent. We use the notation $x \in V(G)$ to denote a vertex in G and $(x, y) \in E(G)$ to denote an edge in G. We consider only finite graphs, i.e., $G = (V, E)$ such that $|V|$ is finite.

Definition 2 (undirected edge)**.** *An edge, $(x, y) \in E(G)$ is undirected if (y, x) is also an edge in G.*

Definition 3 (undirected graph)**.** *A graph G is undirected if every edge in G is undirected.*

Definition 4 (weighted graph)**.** *A graph G is a weighted graph if for every edge there is a positve number, i.e., weight, assigned to the edge.*

Definition 5 (degree)**.** *The degree of $x \in V(G)$, where G is an undirected graph, is $d_x = \sum_{(x,y)\in E(G)} W_{xy}$ where W_{xy} is the weight of edge (x, y). If G is unweighted, then d_x is just the number of edges connecting to x.*

Definition 6 (k-regular graph)**.** *A graph G is a k-regular graph if every node has k neighbors.*

Definition 7 (complete graph)**.** *A graph G is complete if there is an edge between every two nodes.*

3.2 Laplacian

The Laplacian of an unweighted graph without loops and multiple edges[1] is a matrix, L, such that

$$L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } \exists \text{ edge } (i, j) \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The Laplacian is symmetric for undirected graphs.

Definition 8 (adjacency matrix)**.** *An adjacency matrix for an undirected graph G is a matrix, A, such that $A_{ij} = 1$ if $(i, j) \in E(G)$ and $A_{ij} = 0$ otherwise.*

Let D be a diagonal matrix for a graph G such that $D_{ii} = d_i$. Then the Laplacian of G is $L = D - A$.

[1] A loop is an edge that starts and ends at the same node. We say there is a multiple edge from x to y when there is more than one edge from x to y.

The normalized Laplacian, $\mathcal{L}$, is defined as

$$\mathcal{L}_{ij} = \begin{cases} 1 & \text{if } i = j \\ \frac{-1}{\sqrt{d_i d_j}} & \text{if } i \neq j \text{ and } \exists \text{ edge } (i,j) \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

$D^{-1/2}$ is the diagonal matrix defined such that $D^{-1/2}_{ii} = \frac{1}{\sqrt{d_i}}$ if $d_i \neq 0$ and $D^{-1/2}_{ii} = 0$ otherwise. With this definition, it can be shown that $\mathcal{L} = D^{-1/2} L D^{-1/2}$. Since $L = D - A$, this implies $\mathcal{L} = I - D^{-1/2} A D^{-1/2}$.

The Laplacian is thought of as an operator that acts on functions defined on the graph, G. If $g : V(G) \to \mathbb{R}$, then

$$\mathcal{L}g(i) = \sum_{(i,j)\in E(G)} \mathcal{L}_{ij} g(j). \tag{3.3}$$

Since $\mathcal{L}$ is symmetric and positive semi-definite, all the eigenvalues of $\mathcal{L}$ are real and nonnegative. In fact, the Rayleigh quotient, $inf_{g\neq 0} \frac{\langle g, \mathcal{L}g\rangle}{\langle g,g\rangle}$, shows that 0 is always an eigenvalue of $\mathcal{L}$ with corresponding eigenvector $D^{1/2}\vec{\mathbf{1}}$, where $\vec{\mathbf{1}}$ is the vector of all ones.

$$\begin{aligned}
inf_{g\neq 0} \frac{\langle g, \mathcal{L}g\rangle}{\langle g,g\rangle} &= inf_{g\neq 0} \frac{\left\langle D^{-1/2}g, LD^{-1/2}g\right\rangle}{\langle g,g\rangle} \\
&= inf_{D^{1/2}f\neq 0} \frac{\langle f, Lf\rangle}{\left\langle D^{1/2}f, D^{1/2}f\right\rangle} \\
&= inf_{D^{1/2}f\neq 0} \frac{\sum_{u\in V(G)} f(u) \sum_{(u,v)\in E(G)} (f(u) - f(v))}{\sum_{u\in V(G)} f(u)^2 d_u} \\
&= inf_{D^{1/2}f\neq 0} \frac{\sum_{(u,v)\in E(G)} (f(u) - f(v))^2}{\sum_{u\in V(G)} f(u)^2 d_u}
\end{aligned}$$

Here, the last summation is over all unordered edges in $E(G)$. The number of connected components of a graph is the multiplicity of the zero eigenvalue.

3.2.1 Example: Laplacian operator on the circle

Consider the nodes of our graph, G, as the samples of a circle that is uniformly sampled n times. G is an undirected, 2-regular graph. The adjacency matrix for G is A with elelments A_{ij} such that

$$A_{ij} = \begin{cases} 1 & \text{if } j = (i+1) mod(n) \text{ or } i = (j-1) mod(n) \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

The matrix D is $D = 2I_{n\times n}$. The Laplacian is $L = D - A$ and the normalized Laplacian is $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$. The eigenvalues of $\mathcal{L}$ are $\lambda_k = 1 - cos(\frac{2\pi k}{n})$ for $k = 0, \ldots, n-1$. The eigenvectors are sampled sinusoids in which the frequency increases with increasing λ.

3.3 Diffusion

The diffusion equation is $\partial_t u - \Delta u = 0$. The kernel for the diffusion equation is $K = e^{t\Delta}$. The operator Δ is a negative semi-definite operator. Our Laplcian above, $\mathcal{L}$, is positive semi-definite therefore substituting $\mathcal{L}$ for $-\Delta$ and using series notation gives

$$e^{-t\mathcal{L}} = I - t\mathcal{L} + \frac{t^2\mathcal{L}^2}{2!} - \frac{t^3\mathcal{L}^3}{3!} + \ldots. \tag{3.5}$$

Setting the time step $t = 1$ and using a first order approximation of $e^{-t\mathcal{L}}$ yields

$$\begin{aligned} K = \quad & e^{-\mathcal{L}} \\ \approx \quad & I - \mathcal{L} \\ = \quad & I - (I - D^{-1/2}AD^{-1/2}) \\ = \quad & D^{-1/2}AD^{-1/2}. \end{aligned}$$

$K = D^{-1/2}AD^{-1/2}$ is an approximation to the diffusion operator on the graph. If the graph is k-regular, then it is easy to see that $K = D^{-1}A$.

This definition of the diffusion operator on the graph is not standard. In [16, 5], the diffusion operator is defined in terms of a transition matrix. Given a graph, $G = (V, E)$, and a symmetric, nonnegative mapping, $W : V(G) \times V(G) \longrightarrow \mathbb{R}$, we can assign weights to the edges of the graph. If $(x, y) \in E(G)$, then $W(x, y)$ can be thought of as a measure of how similar node x is to node y. For each $x \in V(G)$ define $d(x) = \sum_{(x,y)\in E(G)} W(x, y)$. The diffusion operator on the graph is then defined as

$$K(x, y) = \frac{W(x, y)}{d(x)} \tag{3.6}$$

All the entries of K are nonnegative and the rows of K sum to one. K is a transition matrix. $K(x, y)$ can be thought of as the probability of transitioning in one time step from node x to node y. $K^t(x, y) := (K^t)(x, y)$ is then the probability of transitioning from node x to node y in exactly t time steps.

The main difference between these diffusion operators are the diagonal elements. Since the adjacency matrix is zero along the diagonal, the first defintion of

a diffusion operator has zeroes along its diagonal. The transition matrix definition of the diffusion operator does not necessarily have zeroes along the diagonal. In practice the diagonal element tends to be the largest entry in the row.

3.4 Weighted graphs

For a weighted graph without loops, $G = (V, E, W)$, the Laplacian becomes $L = D - W$ where D is the diagonal matrix such that $D_{xx} = d_x$ and W is the matrix of edge weights. The normalized Laplacian is then given by $\mathcal{L} = I - D^{-1/2} W D^{-1/2}$. Thus, $\mathcal{L}(i, j)$ is given by:

$$\mathcal{L}_{ij} = \begin{cases} 1 & \text{if } i = j \\ \frac{-W_{ij}}{\sqrt{d_i d_j}} & \text{if } i \neq j \text{ and } \exists \text{ edge } (i, j) \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

The eigenvalues of the weighted graph can be computed using the Rayleigh quotient. If $f = D^{-1/2} g$, then computations similar to those above give:

$$inf_{g \neq 0} \frac{\langle g, \mathcal{L} g \rangle}{\langle g, g \rangle} = \; inf_{D^{1/2} f \neq 0} \frac{\sum_{(u,v) \in E(G)} (f(u) - f(v))^2 W_{uv}}{\sum_{u \in V(G)} f(u)^2 d_u}.$$

Chapter 4

Kernel eigenmap methods

One of the oldest techniques for dimensionality reduction is principal component analysis (PCA). Since kernel eigenmap methods come from PCA, we briefly review PCA [43] and discuss similarities between the two techniques.

4.1 Principal component analysis

Suppose our data is $X \in \mathbb{R}^{D\times n}$ where D is the spectral dimension and n is the number of pixels. We think of an experiment measuring variables $d_i,\ i = 1 \dots D$ being ran n times. The mean of the data is given by the vector m such that $m_i = \frac{1}{n}\sum_{j=1}^{n} X_{ij}$. To mean center the data points $\{x_i\}_{i=1}^{n}$, i.e., the columns of X, m is subtracted from each data point. Thus, the new data points are $\{x_i - m\}_{i=1}^{n}$. Assume X is mean centered otherwise since PCA is a linear transformation we can translate the data to the origin, perform the transformation, and then translate the data back. The first step of PCA involves computing the covariance matrix[1]

$$C = \frac{1}{n-1} XX^{\star} \tag{4.1}$$

C is a symmetric, positive semi-definte matrix that measures the similarities or correlations amongst the variables. More formally, C_{ij} measures the linear relationship between the variables d_i and d_j. Since C is symmetric and positive

[1]The normalization $\frac{1}{n-1}$ gives a less biased estimate than $\frac{1}{n}$ for small sample sizes.

semi-definite the eigenvalues of C are real and nonnegative and the eigenvectors of C are orthonormal. Let V be the matrix of eigenvectors of C. The k^{th} column of V is the eigenvector corresponding to the k^{th} largest eigenvalue. These eigenvectors become the new, principal axes for the data. The new coordinates for the data are the projections of X onto V, i.e., $V^{\star}X$. Figure 4.1 demonstrates data points and the principal axes that come from PCA.

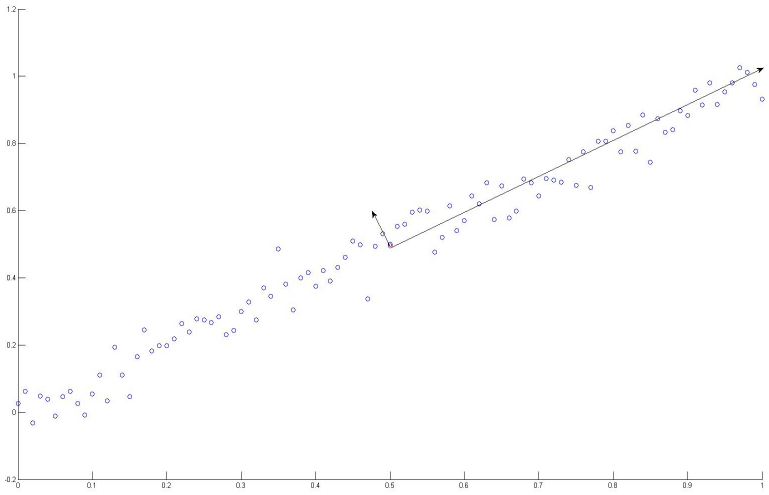


Figure 4.1: Data points with principals axes.

As can be seen from the figure, PCA transforms the data so that if the data is projected along any axis, the greatest variance lies on the first eigenvector, i.e., principal component. This is the line of best fit for the data. The next principal component gives the second greatest variance of all vectors that are orthogonal to the first principal component. The first two principal components together give the plane of best fit for the data. Continuing in this fashion, all D principal components are determined.

The Karhunen-Loeve theorem proves that the principal components are an optimal basis for the data. More precisely, suppose P is a projection operator onto a k-dimensional subspace of $\mathbb{R}^D$. The distortion is then given by

$$Distortion(P) = \sum_{j=1}^{N} \|x_j - Px_j\|_{\ell_2}^2 .$$

If $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$ are the eigenvalues corresponding to the correlation matrix[2] for the data and $\{v_i\}_{i=1}^D$ are our principal components. The minimizer of the distortion is the projection operator projecting onto the subspace spanned by $\{v_i\}_{i=1}^k$, and the distortion is given by

$$P_{\{v_i\}_{i=1}^k} = \arg\min Distortion(P).$$

Here the $\arg\min$ is taken over all rank k projections. The distortion of the $\arg\min$ is

$$Distortion(P_{\{v_i\}_{i=1}^k}) = \sum_{j=k+1}^{D} \lambda_j.$$

Often it is the case that only the first d eigenvalues are significant. Most of the variance is captured by projecting onto the first d eigenvectors. In this sense, the data is said to be *intrinsically* d dimensional. The dimensionality reduction follows by allowing the new coordinates for the data to be the projection onto the subspace of the first d principal components,

$$X \longmapsto P_{\{v_i\}_{i=1}^d} X.$$

[2]The correlation matrix is the same as the covariance matrix except each variable x_i is divided by its standard deviation $\frac{x_i}{SD(x_i)}$

PCA can also be thought of as an application of the singular value decomposition (SVD). The SVD states that if $X^\star$ is an $n \times D$ matrix, then $X^\star$ can be factored as the product

$$X^\star = U\Sigma V^\star. \tag{4.2}$$

Here, Σ is a $n \times m$ diagonal matrix and U and V are $n \times n$ and $D \times D$ unitary matrices, respectively. The columns of V are precisely the eigenvectors of the covariance matrix of X. Thus, the new coordinates for the data are given by $X^\star V$ or more efficiently by $U\Sigma$.

PCA does not work well for all types of data. If the data does not have a Gaussian distribution then the principal components may not do a very good job of capturing the underlying basis for the data, Figure 4.2.

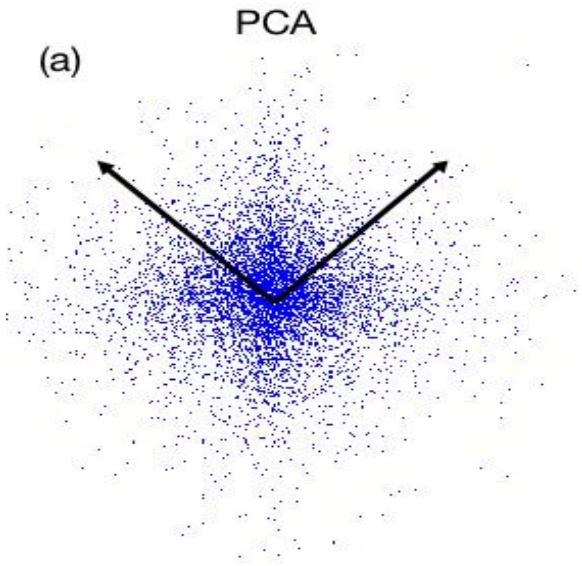


Figure 4.2: PCA fails to capture the underlying basis. (Courtesy Jonathon Schlens)

Also, if the data is nonlinear then PCA may not capture the underlying basis well. In Figure 4.3 we consider a circle; here a natural basis is the angle θ coming

from polar coordinates. When PCA is performed the data is projected along a line through the center of the circle. Even though the circle is one dimensional, projecting the data onto its first principal component does not preserve the neighborhood structure of the circle.

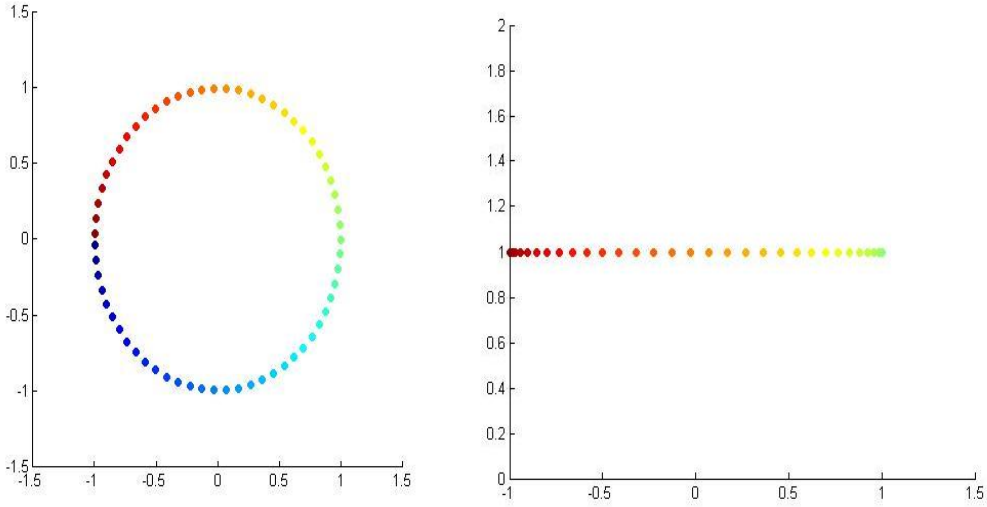


Figure 4.3: PCA fails to capture angular nature and neighborhood structure of the circle.

We would like to be able to perform dimensionality reduction on non-Gaussian and nonlinear data. This is a major reason behind the creation of kernel eigenmap methods.

4.2 Kernel Eigenmap Methods

Kernel methods were first introduced by Aizerman et al. in 1964 [1]. Most kernel eigenmap methods consist of the following two steps:

1. Construction of a $n \times n$ symmetric, positive semi-definite kernel K.

2. Diagonalizing K and selecting the d most significant eigenvectors $v_1, \ldots, v_d \in \mathbb{R}^n$. These eigenvectors are also called eigenmaps.

The dimensionality reduction comes from mapping the each data point $x_i \in \mathbb{R}^D$ to $y_i \in \mathbb{R}^d$, where $d \leq D$ and

$$x_i \mapsto \begin{pmatrix} v_{m_1}(i) \\ \vdots \\ v_{m_d}(i) \end{pmatrix}$$

$\{y_i\}_{i=1}^n$ are the new coordinates for the data.

The idea behind kernel methods is to express correlations or similarities between vectors in the data space X in terms of a symmetric, positive semi-definite kernel function $K : X \times X \mapsto \mathbb{R}$. Since K is symmetric and positive semi-definite, Mercer's theorem proves that $K(x, y)$ can be expressed as an inner product in a high, possibly infinite, dimensional space,

$$K(x, y) = \Phi(x) \cdot \Phi(y), \forall x, y \in X. \tag{4.3}$$

Here $\Phi : X \mapsto \mathbb{H}$ where $\mathbb{H}$ is the high dimensional inner product space also known as the feature space. The importance of this comes from the fact that many algorithms for data analysis only use the dot product between data points. The data is mapped to a high dimensional space and then the inner products, $\langle \Phi(x), \Phi(y) \rangle$, are computed. Equation 4.3 allows for the algorithm to run without actually mapping the data into $\mathbb{H}$ and computing $\langle \Phi(x), \Phi(y) \rangle$.

Example 9 (Feature space for the Gaussian)**.** *One of the standard kernels to use*

is the Gaussian,

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

Here $\sigma \in \mathbb{R}$, is not necessarily related to the standard deviation of the data; σ is a parameter that the designer of the kernel can tune. The feature map Φ for the Gaussian can be derived as follows,

$$\begin{aligned}
K(x,y) &= e^{-\frac{\|x-y\|^2}{\sigma}} \\
&= e^{-\frac{\langle x-y, x-y\rangle}{\sigma}} \\
&= e^{-\frac{\langle x,x\rangle}{\sigma}} e^{-\frac{\langle y,y\rangle}{\sigma}} e^{2\frac{\langle x,y\rangle}{\sigma}} \\
&= e^{-\frac{\|x\|^2}{\sigma}} e^{-\frac{\|y\|^2}{\sigma}} \left(1 + \frac{2\langle x,y\rangle}{\sigma} + \frac{(2\langle x,y\rangle)^2}{2!\sigma^2} + \dots\right) \\
&= e^{-\frac{\|x\|^2}{\sigma}} e^{-\frac{\|y\|^2}{\sigma}} \left(1, \sqrt{\frac{2\langle x,y\rangle}{\sigma}}, \sqrt{\left(\frac{2\langle x,y\rangle}{2!\sigma}\right)^2}, \dots\right) \cdot \left(1, \sqrt{\frac{2\langle x,y\rangle}{\sigma}}, \sqrt{\left(\frac{2\langle x,y\rangle}{2!\sigma}\right)^2}, \dots\right)
\end{aligned}$$

for the sake of clarity and to avoid introducing multi-index notation, we do this next step in one dimension.

$$\begin{aligned}
&= e^{-\frac{\|x\|^2}{\sigma}} \left(1, \sqrt{\frac{2}{\sigma}}x, \sqrt{\frac{2}{2!\sigma}}x^2, \dots\right) \cdot e^{-\frac{\|y\|^2}{\sigma}} \left(1, \sqrt{\frac{2}{\sigma}}y, \sqrt{\frac{2}{2!\sigma}}y^2, \dots\right) \\
&= \Phi(x) \cdot \Phi(y).
\end{aligned}$$

The above shows that the feature space for this kernel is infinite dimensional.

One motivation for mapping data into a higher dimensional space is that the kernel allows distinct classes to be separated and keeps elements within the same class together. An easy example of this can be seen by considering two distinct classes of points in $\mathbb{R}^2$. The points in the first class form a disk surrounded by the

points of the second class, i.e. the points in the second class form the boundary of the disk. It is not possible to separate these classes using vector angle or Euclidean distance. The popular technique from support vector machines of separating the classes with a hyperplane does not work here either. However, if we take a function, Φ, that maps the interior of the disk into $\mathbb{R}^3$ and keeps the boundary of the disk fixed to the $\mathbb{R}^2$ plane embedded in $\mathbb{R}^3$, then these classes will have been separated. Vector angle and the hyperlane will then be able to classify this data.

Another reason for using kernels is that they allow for more complex, non-linear relationships between the data. The Gaussian kernel demonstrates that it can preserve the neighborhood structure of the circle. First, some kernel eigenmap methods are reviewed.

4.2.1 Kernel PCA

Kernel PCA was first used in 1995 for support vector machines; [42] gives a brief introduction to kernel PCA and its applications.

Suppose our data is $X = \{x_i\}_{i=1}^n \subseteq \mathbb{R}^D$ and there is a mapping $\Phi : X \mapsto \mathbb{H}$. Kernel PCA is PCA performed in the feature space. The covariance in the feature space is given by

$$C_\Phi = \frac{1}{n-1} \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\star. \tag{4.5}$$

Assume here that the data is centered in the feature space. Just as with PCA, the covariance is diagonalized. Let the columns $V \in \mathbb{R}^{n \times n}$ be the eigenvectors of C_Φ. If v is an eigenvector of C_Φ, then the following calculations show v is in the span

$\{\Phi(x_1), \ldots, \Phi(x_n)\}$

$$\begin{aligned} \lambda v &= C_\Phi v \\ &= \frac{1}{n-1}\sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\star v \\ &= \frac{1}{n-1}\sum_{i=1}^{n} a_i \Phi(x_i). \end{aligned}$$

Here $a_i = \Phi(x_i)^\star v$ for all $i = 1, \ldots, n$. Multiplying the eigenvector equation, $\lambda v = Cv$, by $\Phi(x_j)$ allows for the construction of the kernel, K.

$$\Phi(x_j) \cdot \lambda v = \Phi(x_j) \cdot C_\Phi v \tag{4.6}$$

$$= \Phi(x_j)^\star C_\Phi \sum_{i=1}^{n} \tilde{a}_i \Phi(x_i) \tag{4.7}$$

$$= \Phi(x_j)^\star \frac{1}{n-1}\sum_{l=1}^{n} \Phi(x_l)\Phi(x_l)^\star \sum_{i=1}^{n} \tilde{a}_i \Phi(x_i) \tag{4.8}$$

Define the kernel, K,

$$K_{li} = \Phi(x_l) \cdot \Phi(x_i) = \Phi(x_l)^\star \Phi(x_i). \tag{4.9}$$

Thus, Equation 4.8 equals

$$\begin{aligned} \Phi(x_j)^\star \frac{1}{n-1}\sum_{l=1}^{n} \Phi(x_l)\Phi(x_l)^\star \sum_{i=1}^{n} \tilde{a}_i \Phi(x_i) &= \frac{1}{n-1}\sum_{i,l=1}^{n} K_{jl}K_{li}\tilde{a}_i \\ &= \frac{1}{n-1}\sum_{i=1}^{n} (K^2)_{ji}\tilde{a}_i \\ &= \frac{1}{n-1}K^2\tilde{a}. \end{aligned}$$

Similarly,

$$\begin{aligned}
\Phi(x_j) \cdot \lambda v &= \lambda \Phi(x_j)^\star \sum_{i=1}^{n} \tilde{a}_i \Phi(x_i) \\
&= \lambda \sum_{i=1}^{n} \tilde{a}_i \Phi(x_j)^\star \Phi(x_i) \\
&= \lambda K_{ji} \tilde{a}_i \\
&= \lambda K \tilde{a}.
\end{aligned}$$

Therefore, combining these two results gives

$$(n-1)\lambda K \tilde{a} = K^2 \tilde{a}. \tag{4.10}$$

Assuming K is one-to-one 4.10 can be rewritten as

$$(n-1)\lambda \tilde{a} = K \tilde{a}. \tag{4.11}$$

It is easy to see that K is positive semi-definite,

$$\begin{aligned}
\langle z, Kz \rangle &= \sum_{i=1}^{n} z_i \sum_{j=1}^{n} K_{ij} z_j \\
&= \sum_{i=1}^{n} z_i \sum_{j=1}^{n} \Phi(x_i)^\star \Phi(x_j) z_j \\
&= \left\langle \sum_{i=1}^{n} \Phi(x_i) z_i, \sum_{j=1}^{n} \Phi(x_j) z_j \right\rangle \\
&\geq 0.
\end{aligned}$$

To arrive at the new coordinates for the feature data, $\{\Phi(x_1), \ldots, \Phi(x_n)\}$, we must project this data onto the eigenvectors of C_Φ. The new coordinates of $\Phi(x_i)$ are $V^\star \Phi(x_i)$ for all $i = 1 \ldots n$.

In terms of dimensionality reduction, it should be noted that if $n \geq D$ then the new coordinates for the data may have *more* dimensions than the original data. Here

it is important to represent the feature data in terms of the chosen $d \leq D$ significant eigenvectors. All the results for PCA and X can be translated into analagous results for kernel PCA and $\Phi(X)$. Also, like PCA, if the mapping Φ is invertible then its corresponding Kernel PCA is invertible.

4.2.2 Locally linear embedding

Locally linear embedding (LLE) [41] was developed by Saul and Roweis in 2000. LLE was one of original kernel methods. The basic idea behind LLE is as follows.

1. Construct a graph. Consider the data points x_i for $i = 1, ..., n$ as the nodes of our graph. The edges and weights for each edge are created in step 2.

2. Express each data point x_i as a linear combination of its neighbors. There are different notions of what constitutes being a neighbor. The two most common methods are either

   (a) fix an integer k and choose the k-closest points to x_i or

   (b) fix an $\epsilon > 0$ and consider all the points within ϵ of x_i as neighbors.

   Regardless of how we choose the defintion of neighbor, not every data point can be represented as a linear combination of its neighbors. This is why it is necessary to minimize a functional, F, that projects each data point onto the space spanned by its neighbors.

$$\arg\min_{W} F(\{x_i\}_{i=1}^{n}) = \arg\min_{W} \sum_{i=1}^{n} |x_i - \sum_{j \in N(i)} w_{ij} x_j|^2 \tag{4.12}$$

$$N(i) = \{x_j : j \in \{1, 2, ...j-1, j+1, ...n\} \textit{ and } x_j \textit{ is a neighbor of } x_i\} \tag{4.13}$$

This creates a hyperplane through each node in the graph.

3. The functional in step 2 is invariant to rotations and rescaling of the data, X. If we add the condition that

$$\sum_j w_{ij} = 1 \text{ for } i = 1, ..., n \tag{4.14}$$

then this functional becomes invariant to translations. This invariance means that any linear transformation that maps a hyperplane to a lower dimensional space must preserve the weight matrix created in step 2. LLE preserves the local geometry of the manifold. We want to find new coordinates, Y, that maintain the local geometry, W. This is done by minimizing the following functional:

$$\arg\min_Y \Gamma(Y) = \arg\min_Y \sum_{i=1}^{n} |y_i - \sum_{j \in N(i)} w_{ij} y_j|^2. \tag{4.15}$$

Here each vector $y_i \in \mathbb{R}^N$ and is a column of Y. $\Gamma(Y)$ can be written as a quadratic form,

$$\Gamma(Y) = \sum_{ij} K_{ij}(y_i \cdot y_j), \tag{4.16}$$

where $K = (I - W)^*(I - W)$. This assertion is verified by expanding the square terms to give

$$\sum_i \Big[(y_i - \sum_j w_{ij} y_j)(y_i - \sum_j w_{ij} y_j)^*\Big]. \tag{4.17}$$

Note that the first factor, $I - W$, is analogous to the Laplacian operator on the graph, only now a "weighted" adjancey matrix is used. Expanding 4.17

gives

$$K_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki} w_{kj}. \tag{4.18}$$

The optimal embedding - up to a global rotation of the embedding space - is found by computing the bottom $d+1$ eigenvectors of the matrix, K. The eigenvector with corresponding eigenvalue 0 is discarded, giving a d-dimensional representation.

The kernel for LLE is $K = (I - W)^*(I - W)$. This is the square of the Laplacian on the graph constructed from the data points and the chosen neighborhood structure. The new coordinates for the data come the d eigenmaps corresponding to the d smallest nonzero eigenvalues. The smallest are chosen because the functional Γ is being minimized.

4.2.3 Laplacian eigenmaps

In their paper Laplacian Eigenmaps for Dimensionality Reduction and Data representation [2], Belkin and Niyogi construct a Laplacian on the data points considered as a graph. Briefly, the motivation for doing this is as follows. The data points are viewed as samples of a d-dimensional manifold, M, that is embedded in a higher dimensional space. The goal is to look at functions, $f : M \mapsto \mathbb{R}$, such that points close to each other on the manifold get mapped to points close to each other in $\mathbb{R}$. Functions , $f : M \mapsto \mathbb{R}$, that satisfy the Mean Value Theorem

$$|f(x_i) - f(x_j)| \leq \|x_i - x_j\| \, \|\nabla f\| + o\left(\|x_i - x_j\|\right),$$

are considered[3]. This leads us to consider minimizers of the functional $\int_M |\nabla f|^2$, i.e., we want to solve the following minimization problem

$$f_i = \underset{\|f\|_{L^2(M)}=1}{\arg\min} \int_M |\nabla f|^2. \tag{4.19}$$

Here f_i is orthogonal to f_j for $j < i$. Assuming the manifold is without boundary and applying integration by parts demonstrates that harmonic functions on the manifold are minimizers of this functional. Thus we solve for f, such that

$$f_i = \underset{\|f\|_{L^2(M)}=1}{\arg\min} \int_M f\Delta f. \tag{4.20}$$

The first d eigenfunctions become our new coordinate system for the data.

The algorithm first considers the data as a graph $G = (V, E)$ and assigns weights to the edges. Each data point, x_i is vertex i of the graph. If x_i is connected to x_j then W_{ij} is the weight of the edge from vertex i to vertex j, it is given by

$$W_{ij} = e^{-\frac{\left\|x_i - x_j\right\|^2}{\sigma}}. \tag{4.21}$$

Here σ is in $\mathbb{R}$ and the matrix, W, is symmetric and easy to compute. If If x_i and x_j are not connected then $W_{ij} = 0$. The final step requires solving a generalized eigenvector problem.

$$Lf = \lambda Df$$

[3]Technically, some justification should be given explaining why it acceptable to consider the Euclidean distance in the high dimensional space as opposed to the Riemannian metric on the manifold. I omit the details here, see [2] for a detailed explanation.

Here, $L = D - W$ is the Laplacian and D is a diagonal matrix such that $D_{ii} = \sum_{j=1}^{n} W_{ij}$. The first d eigenmaps become our new coordinate system for the data.

It is interesting to note that although LLE and Laplacian Eigenmaps have similar results, i.e., they consider the Laplacian of the graph, they are derived from very different frameworks. Belkin and Niyogi use the geometry of the manifold and the eigenfunctions for the Laplacian operator on the manifold to motivate computing eigenvectors. Instead of using geometry, Saul and Roweis create a functional that minimizes the ℓ_2-norm while keeping W invariant to rotations, translations and dilations of the data.

Example 10 (Laplacian eigenmaps for pure data)**.** *It is instructive to see how dimensionality reduction using Laplacian eigenmaps works for "pure" data. Pure data can be explained as follows. Suppose our hypercube, X, contains pixels $\{x_i\}_{i=1}^{n} \in \mathbb{R}^D$ and m classes, $\{C_1, \ldots, C_m\} \in R^D$. We say the data is pure if for each x_i there exists a C_j such that $x_i = C_j$.*

The kernel is $K : X \times X \mapsto \mathbb{R}$ such that

$$K(x, y) = e^{-\frac{\|x-y\|^2}{\sigma}} \ \forall x, y \in X. \tag{4.22}$$

The matrix form of the kernel is $K[i, j]$ such that $K[i, j] = K(x_i, x_j)$. If one considers the kernel acting on the pure data, X above, then since there are at most $\frac{m(m-1)}{2} + 1$ unique entries in the matrix K, this matrix has a very special form.

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1m} \\ \vdots & K_{22} & \dots & K_{2m} \\ & \vdots & \ddots & \vdots \\ & & & K_{mm} \end{bmatrix}$$

Here each K_{ij} is a submatrix of size $|C_i| \times |C_j|$ with all of the entries being the same nonzero constant. The constant for the diagonal submatrices, K_{ii} is 1. If one assumes that each submatrix has a unique constant then the kernel, K, is of rank m. More importantly, if the number of classes is greater than two then only one eigenvector is necessary to separate the classes. This says that the transpose of the first eigenvector, i.e., corresponding to the largest eigenvalue, of K will be of the form

$$\left[\underbrace{v_1, v_1, \dots v_1,}_{|C_1| \ times} \ \underbrace{v_2, v_2, \dots v_2,}_{|C_2| \ times} \ \cdots \ \underbrace{v_m, v_m \dots v_m}_{|C_m| \ times} \right],$$

with each of v_i being unique. This eigenvector gives perfect classification. All the elements of the i^{th} class, and only elements of the i^{th} class, were mapped to v_i for all $i = 1, \dots, m$. The kernel separated and preserved the classes.

It is interesting to note that if there are only two classes, then above reasoning fails because the first eigenvector is $\vec{1}$. It is then necessary to look at the second eigenvector to separate classes. Also, the above is only true for pure data. Once the data has mixed pixels then determining the target dimension, d, becomes much more complex.

4.2.4 Diffusion maps

The Diffusion Maps algorithm models diffusion as a transition operator on the graph. First a graph for the data is chosen using k-nearest neighbors or a radius. The weights are assigned to the edges using

$$W_{ij} = e^{-\frac{\left\|x_i - x_j\right\|_2^2}{\sigma}}. \tag{4.23}$$

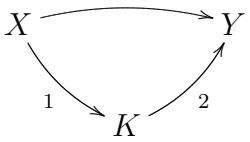
Note, although there are no loops in the graph, $W_{ii} = 1$, for all $i = 1 \ldots n$ and W is symmetrized by defning x_j to be a neighbor of x_i if x_i is a neighbor of x_j. The diffusion operator, T, is created by normalizing W to be doubly stochastic, i.e., that is the rows and columns sum to one. Thus, $T = D^{-1/2} W D^{-1/2}$, where $D_{ii} = \sum_{j=1}^{n} W_{ij}$. All the entries of T are nonnegative and the rows and columns of T sum to one. In practice, it is necessary to iterate this normalization several times so that the column and row sums converge to one. T is a transition matrix. $T(x, y)$ can be thought of as the probability of transitioning in one time step from node x to node y. $T^t(x, y) = (T^t)(x, y)$ is then the probability of transitioning from node x to node y in exactly t time steps.

Once the diffusion operator, T, is constructed it is diagonalized and the eigenvectors corresponding to the d largest eigenvalues are taken as the new coordinates.

Chapter 5

Frame based approach

Kernel eigenmaps methods represent the data in terms of the eigenvectors of the kernel. It's a two step process.



The steps consist of the following processes:

1. Create an $N \times N$ positive semi-definite, symmetric kernel K.

2. Diagonalize K and select the d most significant eigenvectors $v_1, \ldots, v_d \in \mathbb{R}^N$.

   Our transformed data points are then $y_j = \begin{pmatrix} v_1(i) \\ \vdots \\ v_d(i) \end{pmatrix} \in \mathbb{R}^d$.

There is empirical evidence to suggest that eigenvectors might not be ideal for representing the data. The basic reasoning is as follows [30]. Classes of data in the hyperspectral image are not orthogonal to each other and they do not necessarily have norm one. Ideally, a kernel would transform the data to a higher dimensional feature space in which the classes are orthogonal. This would allow for the data to be separated using a metric such as vector angle. However, constructing a kernel to orthogonalize the classes is difficult. As we have seen in the hyperspectral chapter,

the classes themselves often overlap and have similar profiles.

An alternative approach to this dilemma is to relax the orthonormality condition. Frames allow us to do this. The data can be represented by vectors that are not necessarily orthogonal or norm one. The hope here is that each class will be mapped to its own dimension, i.e., element in the frame. Here is an example motivating the above discussion.

Example 11. *The following example comes from Matt Hirn [26]. Figure 5.1 is a tile with three classes of data, grass, asphalt, and trees. The data is not pure in the sense that the pixels may be mixed and there is also some soil in the image too. This tile is 32 × 32 pixels in the Urban data set discussed above.*

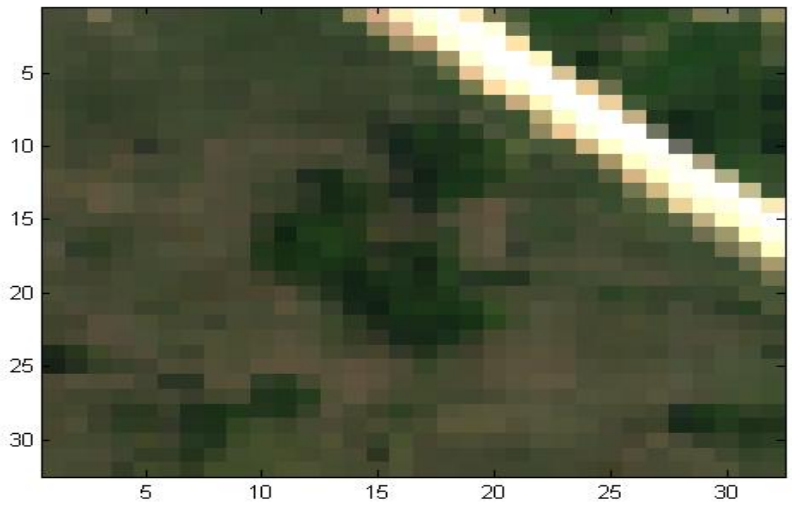
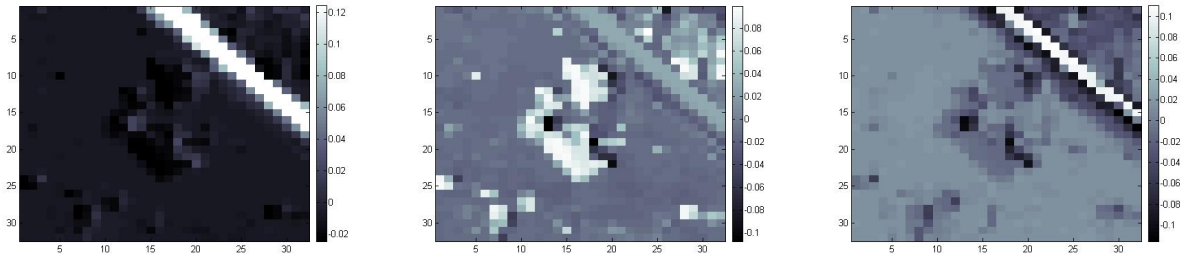


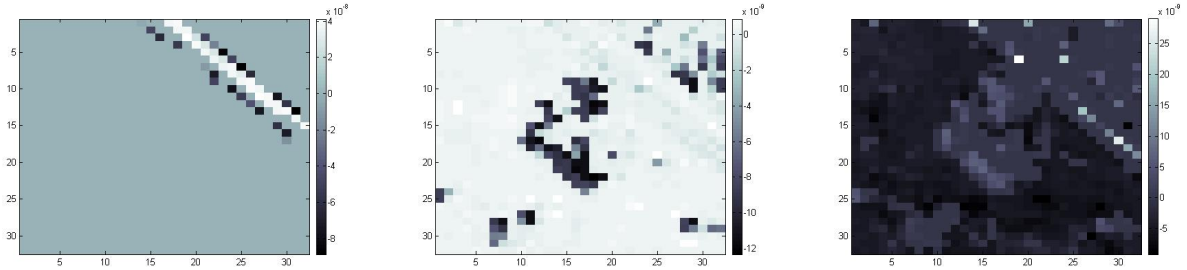
Figure 5.1: The tile has grass vegetation class, asphalt class, and tree class.

Ideally, a successful dimensionality reduction would map the data in such a way that the classes can be separated. In practice, this is difficult to achieve. The LLE algorithm creates eigenmaps for the data that do not correspond to classes.



(a) (b) (c)

Figure 5.2: LLE eigenmaps on the tile (a) first eigenmap (b) second eigenmap (c) third eigenmap



(a) (b) (c)

Figure 5.3: Frame bands for the tile, (a) band 1 (road) (b) band 2 (trees) (c) band 3 (grass vegetation)

This does not allow for successful pixel classification using vector angle or Euclidean distance. Figure 5.2 shows the first three bands, i.e., eigenmaps, of LLE being ran on the tile.

In Figure 5.2, one can see that the eigenmaps are not separating distinct classes well. Frames do a better at separating classes in this tile. In Figure 5.3, the frame spans the same space as the three eigenmaps above. This frame was chosen because its coefficients had mimimun ℓ^1-norm in the representation formula. This will be explained in more detail below.

5.1 Introduction to frames

Let $\Phi = \{\varphi_j\}_{j=1}^s \subset \mathbb{C}^N$, where $s \geq N$. Φ is a *finite frame* for $\mathbb{C}^N$ if there exist constants $A, B > 0$ such that

$$A\|f\|^2 \leq \sum_{j=1}^{s} |\langle f, \varphi_j \rangle|^2 \leq B\|f\|^2, \qquad \forall\ f \in \mathbb{C}^N. \tag{5.1}$$

The numbers A, B are called the *frame bounds*. It is a well known fact that any spanning set is a frame for $\mathbb{C}^N$, while every frame is indeed a spanning set. A frame is *tight* if one can choose $A = B$ in the definition, i.e., if

$$\sum_{j=1}^{s} |\langle f, \varphi_j \rangle|^2 = A\|f\|^2, \qquad \forall\ f \in \mathbb{C}^N. \tag{5.2}$$

Finally, a frame is *unit norm* if

$$\|\varphi_j\| = 1, \qquad \forall\ j = 1, \ldots, s. \tag{5.3}$$

If Φ satisfies (5.1), (5.2), and (5.3), then we say Φ is a *finite unit norm tight frame* (FUNTF) for $\mathbb{C}^N$. In this case, the frame bounds satisfy $A = B = s/N$. In particular, if Φ is a FUNTF with frame bounds $A = B = 1$, then Φ is an orthonormal basis.

Assume now that $\Phi = \{\varphi_j\}_{j=1}^s$ is a frame for $\mathbb{C}^N$. The *analysis operator* of Φ is defined as follows:

$$L : \mathbb{C}^N \to \mathbb{C}^s, \quad Lf := \{\langle f, \varphi_j \rangle\}_{j=1}^s.$$

The *synthesis operator* is given by:

$$L^\star : \mathbb{C}^s \to \mathbb{C}^N, \quad L^\star\{c_j\}_{j=1}^s = \sum_{j=1}^{s} c_j \varphi_j.$$

One obtains the *frame operator* by composing $L^\star$ with L:

$$S : \mathbb{C}^N \to \mathbb{C}^N, \quad Sf = L^\star Lf = \sum_{j=1}^{s} \langle f, \varphi_j \rangle \varphi_j$$

Some important properties of S are the following:

(i) S is invertible and self-adjoint.

(ii) Every $f \in \mathbb{C}^N$ can be represented as

$$f = \sum_{j=1}^{s} \langle f, S^{-1}\varphi_j \rangle \varphi_j = \sum_{j=1}^{s} \langle f, \varphi_j \rangle S^{-1}\varphi_j. \tag{5.4}$$

(iii) Φ is a tight frame if and only if $S = AI$.

Based on equation (5.4), one defines the *dual frame* of Φ as $\tilde{\Phi} = \{\tilde{\varphi}_j\}_{j=1}^{s} := \{S^{-1}\varphi_j\}_{j=1}^{s}$; the frame operator of $\tilde{\Phi}$ is S^{-1}. If Φ is a tight frame for $\mathbb{C}^N$, then $S^{-1} = \frac{1}{A}I$, and the representation formula is simple:

$$f = \frac{1}{A} \sum_{j=1}^{s} \langle f, \varphi_j \rangle \varphi_j = \frac{1}{A} \sum_{j=1}^{s} \langle f, \varphi_j \rangle \varphi_j.$$

5.2 Frame potential

Define the frame potential of a finite unit norm frame $\Phi = \{\varphi_j\}_{j=1}^{s}$ for $\mathbb{C}^N$ as:

$$\mathrm{FP}(\Phi) := \sum_{j=1}^{s} \sum_{k=1}^{s} |\langle \varphi_j, \varphi_k \rangle|^2.$$

In [3] a characterization of FUNTFs is given in terms of the frame potential:

Theorem 12 (Benedetto and Fickus 2002)**.** *For a given N and s, let S^{N-1} denote the unit sphere in $\mathbb{C}^N$ and consider:*

$$\mathrm{FP} : \underbrace{S^{N-1} \times \cdots \times S^{N-1}}_{s \text{ times}} \to [0, \infty).$$

Then:

1. *Every local minimizer of the frame potential is also a global minimizer.*

2. *If $s \leq N$, the minimum value of the frame potential is s, and the minimizers are precisely the orthonormal sequences in $\mathbb{C}^N$.*

3. *If $s \geq N$, the minimum value of the frame potential is s^2/N, and the minimizer are precisely the FUNTFs for $\mathbb{C}^N$.*

5.2.1 Subspace frames

Let $\Phi = \{\varphi_j\}_{j=1}^s \subset \mathbb{C}^N$ and let W be a subspace of $\mathbb{C}^N$ of dimension $r < N$. We say Φ is a *finite subspace frame* for W if $\text{span}(\Phi) = W$. It is clear from this definition that there exist constants $A, B > 0$ such that

$$A\|f\|^2 \leq \sum_{j=1}^{s} |\langle f, \varphi_j \rangle|^2 \leq B\|f\|^2, \qquad \forall\ f \in W. \tag{5.5}$$

We note that if we had instead used (5.5) as our definition, then it would not necessarily imply that $\text{span}(\Phi) = W$ but rather that $\text{span}(\Phi) \supseteq W$. The unit norm property as well as the notion of a tight frame remain similar in this setting. More specifically, if we can take $A = B$ in (5.5) then we call Φ a *tight subspace frame*. Finally, if Φ is a finite unit norm tight subspace frame, then we say Φ is a *subspace FUNTF*.

We define L, $L^\star$, and S exactly the same as in section 1, however we note that the properties of these maps change for subspace frames. In particular, we see:

(a) $L : \mathbb{C}^N \to \mathbb{C}^s$ is no longer injective, but rather $\ker(L) = (\mathbb{C}^N \setminus W) \cup \{0\}$.

(b) $L^\star : \mathbb{C}^s \to \mathbb{C}^N$ is no longer surjective, but rather image$(L^\star) = W$.

(c) Based on (a) and (b), we see that $S : \mathbb{C}^N \to \mathbb{C}^N$ is no longer invertible.

Because of (c), none of properties (i) - (iii) from section 1 hold for subspace frames. The question then becomes: in what sense do subspace frames satisfy properties (i) - (iii) above? Theorems below show that subspace frames satisfy natural modifications of the above properties.

Let W_{on} be a set of r orthonormal vectors such that span$(W_{on}) = W$. We will also consider W_{on} as an $N \times r$ matrix where the columns of this matrix are the vectors in the set W_{on}. We define Φ_W to be the $r \times s$ matrix whose columns are the coordinates of Φ in W_{on}; that is:

$$\Phi_W := W_{on}^\star \Phi, \tag{5.6}$$

where we have implicitly used the matrix form of Φ, that is the $N \times s$ matrix whose columns are the elements of Φ. The j^{th} column of Φ_W is the projected W-subspace coordinates of Φ.

Proposition 13. *The set Φ_W consisting of the columns of the matrix Φ_W is a frame for $\mathbb{C}^r$.*

Proof. Since span$(W_{on}) = W$, we have $\ker(W_{on}^\star) \cap W = \{0\}$. Therefore, since span$(\Phi) = W$ as well, we see that $W_{on}^\star \Phi$ has rank r. □

We denote the analysis, synthesis, and frame operators of Φ_W by L_W, $L_W^\star$, and S_W, respectively. In terms of the analysis operator, L, for Φ, $L_W = LW_{on}$. By

proposition 13 we see that S_W will satisfy (i) - (iii).

Theorem 14. *Φ is a subspace FUNTF for W with frame bound A if and only if Φ_W is a FUNTF for $\mathbb{C}^r$ with frame bound A.*

Proof. We do the forward direction first: let $g \in \mathbb{C}^r$, then:

$$\begin{aligned}
\langle S_W g, g\rangle &= \langle L_W g, L_W g\rangle \\
&= \langle \Phi^\star W_{on} g, \Phi^\star W_{on} g\rangle \\
&= \sum_{j=1}^{s} |\langle W_{on} g, \varphi_j\rangle|^2 \\
&= A\|W_{on} g\|^2 \\
&= A\langle W_{on} g, W_{on} g\rangle
\end{aligned}$$

Therefore we have:

$$\begin{aligned}
\langle S_W g, g\rangle - A\langle W_{on} g, W_{on} g\rangle &= 0 \quad \Longrightarrow \\
\langle S_W g, g\rangle - A\langle W_{on}^\star W_{on} g, g\rangle &= 0 \quad \Longrightarrow \\
\langle g, (S_W - AI) g\rangle &= 0 \quad \Longrightarrow \\
S_W &= AI
\end{aligned}$$

For the reverse direction, let $f \in W$. There exists $g \in \mathbb{C}^r$ such that $W_{on} g = f$.

Therefore,

$$\begin{aligned} A\|f\|^2 &= A\langle f, f\rangle \\ &= A\langle W_{on}g, W_{on}g\rangle \\ &= \langle Ag, g\rangle \\ &= \langle S_W g, g\rangle \\ &= \langle W_{on}^{\star} L^{\star} L W_{on} g, g\rangle \\ &= \langle LW_{on}g, LW_{on}g\rangle \\ &= \langle Lf, Lf\rangle \\ &= \sum_{j=1}^{s} |\langle f, \varphi_j\rangle|^2 \end{aligned}$$

□

We define the dual frame of Φ_W in the usual way, that is $\tilde{\Phi}_W = S_W^{-1}\Phi_W$. We now define the *dual subspace frame* of Φ as follows:

$$\tilde{\Phi} := W_{on}\tilde{\Phi}_W = W_{on}S_W^{-1}W_{on}^{\star}\Phi. \tag{5.7}$$

As the name implies, the set $\tilde{\Phi} = \{\tilde{\varphi}_j\}_{j=1}^{s} = \{W_{on}S_W^{-1}W_{on}^{\star}\varphi_j\}_{j=1}^{s}$ will have the following properties:

Proposition 15. *$\tilde{\Phi}$ is a subspace frame for W.*

Proof. This follows from proposition 13. □

Theorem 16. *Every $f \in \mathrm{W}$ can be represented as*

$$f = \sum_{j=1}^{s}\langle f, \tilde{\varphi}_j\rangle\varphi_j = \sum_{j=1}^{s}\langle f, \varphi_j\rangle\tilde{\varphi}_j.$$

Proof. The first representation formula is $\Phi\tilde{\Phi}^\star f = f$ for all $f \in W$. Letting $f = W_{on}g$ for some $g \in \mathbb{C}^r$, we have:

$$\begin{aligned}
\Phi\tilde{\Phi}^\star f &= \Phi(W_{on}S_W^{-1}W_{on}^\star\Phi)^\star f \\
&= \Phi\Phi^\star W_{on}(S_W^{-1})^\star W_{on}^\star(W_{on}g) \\
&= SW_{on}S_W^{-1}g \\
&= SW_{on}(W_{on}^\star SW_{on})^{-1}g \qquad (5.8)
\end{aligned}$$

Since $W_{on}W_{on}^\star$ is the identity on W,

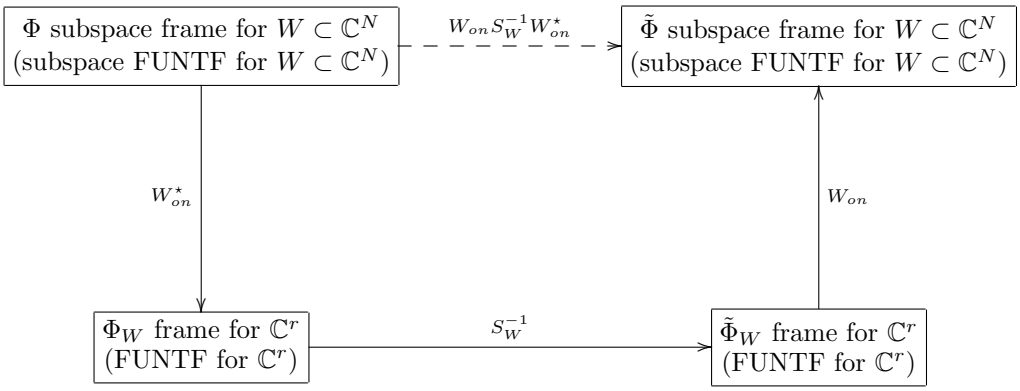
$$\begin{aligned}
(5.8) &= W_{on}W_{on}^\star SW_{on}(W_{on}^\star SW_{on})^{-1}g \\
&= W_{on}g \\
&= f
\end{aligned}$$

The second representation formula is $\tilde{\Phi}\Phi^\star f = f$ for all $f \in W$.

$$\begin{aligned}
\tilde{\Phi}\Phi^\star f &= (W_{on}S_W^{-1}W_{on}^\star\Phi)\Phi^\star f \\
&= W_{on}(W_{on}^\star SW_{on})^{-1}W_{on}^\star SW_{on}g \\
&= W_{on}g \\
&= f
\end{aligned}$$

□

The following commutative diagram illustrates the above ideas:



The following theorem is a trivial generalization of theorem 12:

Theorem 17. *For a given N and s, let W be a subspace of $\mathbb{C}^N$ of dimension $r < N$ and consider the resctricted frame potential:*

$$\mathrm{FP}|_W : \underbrace{(S^{N-1} \times \cdots \times S^{N-1})}_{s \text{ times}} \to [0, \infty).$$

Then:

1. *Every local minimizer of the restricted frame potential is also a global minimizer.*

2. *If $s \leq r$, the minimum value of the restricted frame potential is s, and the minimizers are precisely the orthonormal sequences in W.*

3. *If $s \geq r$, the minimum value of the restricted frame potential is s^2/r, and the minimizer are precisely the subspace FUNTFs for W.*

Theorem 17 shows that the minimum value of the frame potential depends on the dimension of the subspace W.

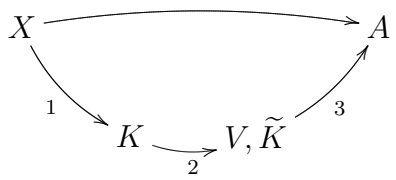
Proof. Let W_{on} be a set of r orthonormal vectors such that $\text{span}(W_{on}) = W$ and consider it as an $N \times r$ matrix. If $\Phi = \{\varphi_j\}_{j=1}^{s}$ is a finite unit norm set of vectors in W, then the coordinates of Φ in W_{on} are given by the $r \times s$ matrix $\Phi_W = W_{on}^{\star}\Phi$. In [3] it is shown that $\text{FP}(\Phi) = \text{Tr}(S^2)$, where S is the frame operator of Φ. Using the previous two statements we then have:

$$\begin{aligned}
\text{FP}|_W(\Phi) &= \text{Tr}(S^2) \\
&= \text{Tr}([(W_{on}\Phi_W)(W_{on}\Phi_W)^{\star}]^2) \\
&= \text{Tr}([\Phi_W\Phi_W^{\star}]^2) \\
&= \text{Tr}(S_W^2) \\
&= \text{FP}(\Phi_W)
\end{aligned}$$

Since Φ_W is a finite unit norm set of vectors in $\mathbb{C}^r$, we can apply theorem 12 to get (1) and (2). Combining theorem 12 along with theorem 14 gives (3). □

5.3 Dimensionality reduction with frames

We combine frames with dimensionality reduction by projecting the columns of the kernel onto a subspace V. Here $V = \{v_1, \ldots, v_r\}$, are the r most significant eigenvectors of K. This gives new r-dimensional coordinates for the data, $\widetilde{K} = V^{\star}K$.



A FUNTF, $\Phi = \{\varphi_j\}_{j=1}^{s}$, for $\mathbb{R}^r$ is found using the frame potential. Note that the

user chooses the number of frame elements, s. Solve the following minimization problem for each column of $\widetilde{K}$:

$$\text{argmin}\|A_i\|_1 \quad \text{subject to} \quad \Phi A_i = \widetilde{K}_i, \quad (P_1)$$

for all $i = 1, \ldots, N$. $\widetilde{K} = \Phi A$, where the i^{th} column of A is A_i above. The new coordinates for an original data point x_i are A_i. In order to achieve dimensionality reduction, we select the d most significant rows of A. Thus, if rows $\{m_1, \ldots m_d\}$ of A are chosen then then data points of X are transformed to rows of A, $x_i \mapsto (A_{m_1}, \ldots, A_{m_d})$.

Note, it is not necessary to use the eigensubspace of K. We are currently investigating other methods for selecting subspaces. The ℓ^1-minimization step comes from recent compressed sensing results. Ideally, we would like to solve the ℓ^0-minimization problem:

$$\text{argmin}\|A_i\|_0 \quad \text{subject to} \quad \Phi A_i = \widetilde{K}_i, \quad (P_0)$$

for all $i = 1, \ldots, N$. The ℓ^0-norm of a vector is the number of nonzero entries in the vector; it gives the sparsest representation of the vector with respect to the frame.

The motivation for finding a sparse representation is as follows. The number of classes in the data is typically much smaller than the number of data points. Each data point corresponds to an area depending on the resolution of the sensor. Within this area, there are most likely a small number of classes, e.g., soil, asphalt. The spectral signature of the pixel is then a combination of the classes within the given area. Since there are small number of classes in this area, the spectral signature should have a sparse representation with respect to "class representation". If each

class can be represented by a frame element, then we can apply known compressed sensing results to determine a sparse representation for the spectral signature of the pixel. One result of compressed sensing is that under certain conditions, the solutions to (P_0) and (P_1) are equal. Ideally, we would like to solve (P_0); however, this is computationally infeasible. We solve the ℓ^1-minimization instead and assume our data satisfies the necessary conditions for (P_0) and (P_1) equivalence. Namely, each data point, i, must satisfy $\|A_i\|_0 \leq C\frac{log(r)}{r}$ where C is a known constant and r is as above.

In the last step of the algorithm each row of A is reshaped into an image of size $n \times m$. The rows that perform the best at separating and preserving classes are selected. This selection is done by visual inspection. We are currently investigating methods for automating the visual inspection process.

Chapter 6

Numerical examples

Our data is one of the standard urban data sets used for hyperspectral imaging. The data is provided by the U.S. Army Corps of Engineers; it is available to the public on their website. The data was collected using HYDICE sensor imagery. The data are $307 \times 307 \times 210$, in which $D = 210$ are the spectral bands representing reflectance data stored as unsigned 16 bit integers. Because of memory and computational constraints, the data is divided into 32×32 tiles Figure 6.1. After disabling unusable bands, the algorithms run on a tile consisting of 32×32 spatial dimensions and 162 spectral dimensions.

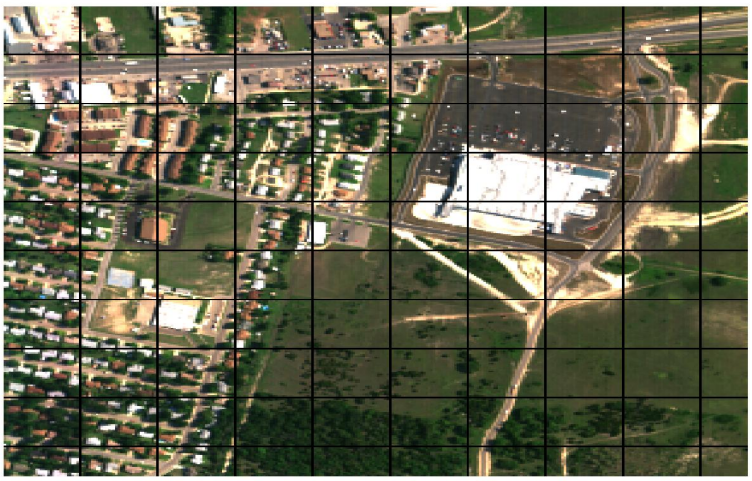
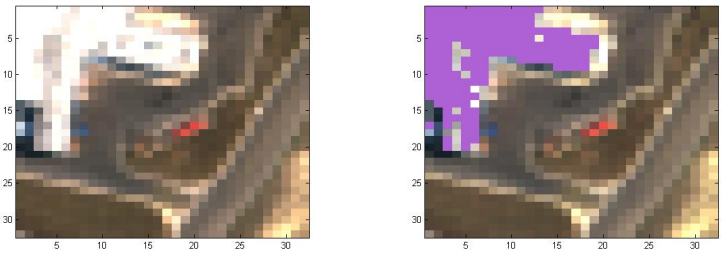


Figure 6.1: Color representation of the gridded urban data.



(a) Tile five from the top, three from the right

(b) Pixels classified as building

Figure 6.2: Example tile

The chosen tile for this example is three tiles from the right edge and five tiles down from the top, Figure 6.2 (a). Pixel (5,2) was chosen to classify the building using vector angle with a tolerance of 0.15. The resulting classification is shown in Figure 6.2 (b). There are 175 pixels classifed as building pixels when the classification is done using all 162 spectral bands.

6.1 Methodology

In this section, I compare LLE, Laplacian Eigenmaps, Diffusion Maps, and the frame based approach previously outlined. The first three methods are run on the above tile with varying numbers of neighbors and with a target dimension of four. Four was chosen as the target dimension because this gave the best classification results. Once the data is reduced, vector angle is again performed on the reduced data with the same pixel, (5,2), being used. A count of the number of false positives and false negatives is then kept. A pixel is a false positive if it is classified as a

building pixel in the reduced data set but not in the original data set. Similary, a pixel is a false negative if it is classified as a building pixel in the original data set but not in the reduced data set.

There are several problems with performing this type of analysis. First and foremost amongst these is the fact that the original classification may be incorrect. Since there is no ground truth data, it is necessary to take classification performed on the nonreduced data set as ground truth. Using classification performed on the original tile as a reference, it is then possible to evaluate classification on the reduced data set. Second, the reference pixel chosen, (5,2), may not be the ideal representative of the building class. In the hyperspectral section above, it was demonstrated that many classes have a large variance. If the chosen pixel is not close to the mean of its class, then vector angle may perform poorly when using that pixel as the class representative. The results below should take these two factors into consideration.

6.2 Results

LLE was run on the above tile with a neighbor setting varying from 2 neighbors to 98 neighbors incrementing each time by 4. The target dimension was four. The following figures are examples of classification that was performed for various neighbor settings; they can be compared with original classification above. For LLE the best classification was performed with 46 neighbors giving 27 false positives and 11 false negatives. This is out of 1,024 pixels in which 175 of the pixels are classified

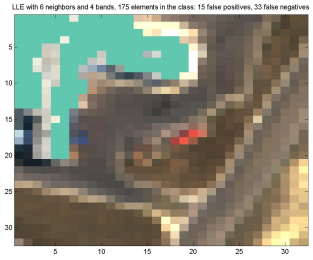
as building.

Laplacian Eigenmaps was run on the above tile with a neighbor setting varying from 2 neighbors to 98 neighbors incrementing by 4 each time. The target dimension was four. The following figures are examples of classification that was performed for various neighbor settings; they can be compared with original classification above. For Laplacian Eigenmaps the best classification was performed with 34 neighbors giving 16 false positives and 12 false negatives.

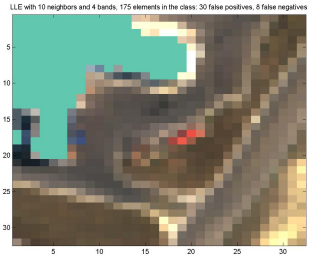
Diffusion Maps was run on the above tile with a neighbor setting varying from 2 neighbors to 98 neighbors incrementing by 4 each time. The target dimension was four. The following figures are examples of classification that was performed for various neighbor settings; they can be compared with original classification above. For Diffusion Maps the best classification was performed with 46 neighbors giving 27 false positives and 11 false negatives.

It is interesting to note that the relationship between the number of neighbors chosen and the number of false positive and negatives is complex. Figures 6.6, 6.7, 6.8 plot the number of false positives and false negatives per number of neighbors. This result shows that one cannot interpolate when choosing the number of neighbors. If there are f_1 false positives when n_1 neighbors are chosen and f_3 false positives when n_3 neighbors are chosen, $n_1 < n_3$, then one cannot conclude the number of false positives is between f_1 and f_3 when n_2 neighbors are chosen, $n_1 < n_2 < n_3$. Also, it is interesting to note how Diffusion Maps and Laplacian Eigenmaps are more stable than LLE.

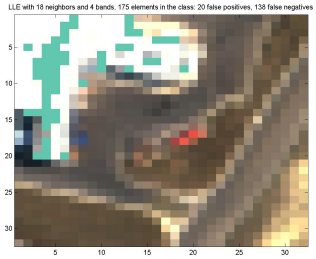
The frame approach to dimensionality reduction, outlined above, was run on



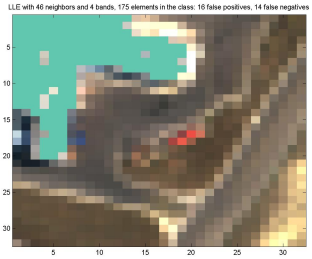
(a) 6 neighbors



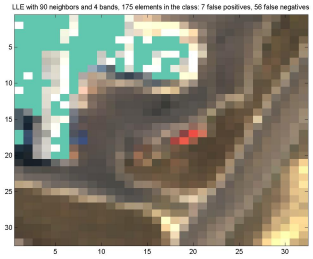
(b) 10 neighbors



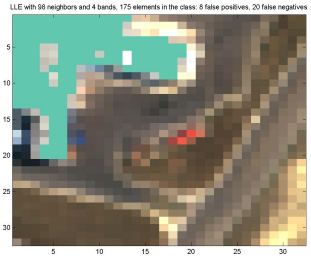
(c) 18 neighbors



(d) 46 neighbors

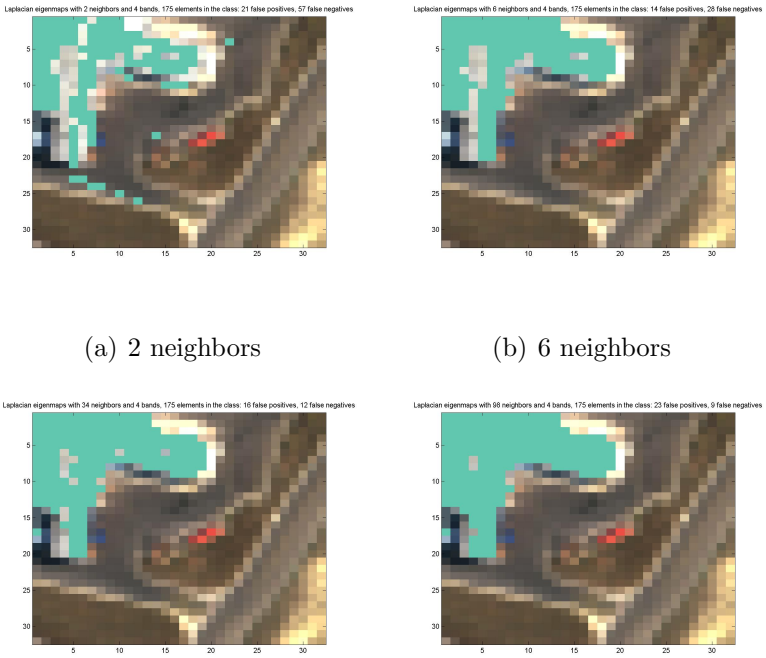


(e) 90 neighbors



(f) 98 neighbors

Figure 6.3: LLE classification with various neighbors



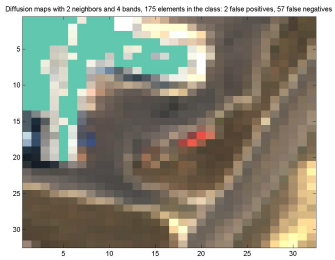
(a) 2 neighbors

(b) 6 neighbors

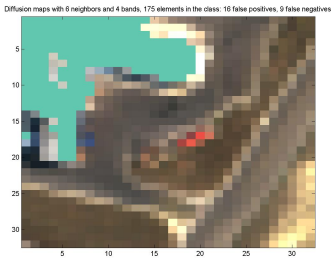
(c) 34 neighbors

(d) 98 neighbors

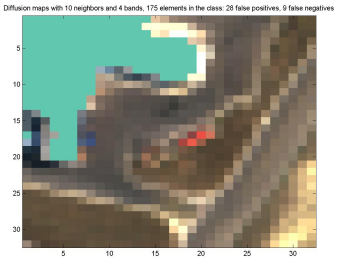
Figure 6.4: Laplacian Eigenmaps classification with various neighbors



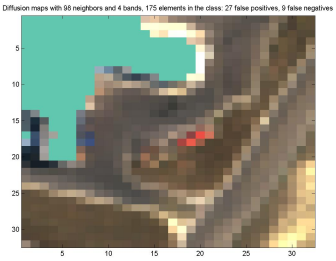
(a) 2 neighbors



(b) 6 neighbors



(c) 10 neighbors



(d) 98 neighbors

Figure 6.5: Diffusion Maps classification with various neighbors

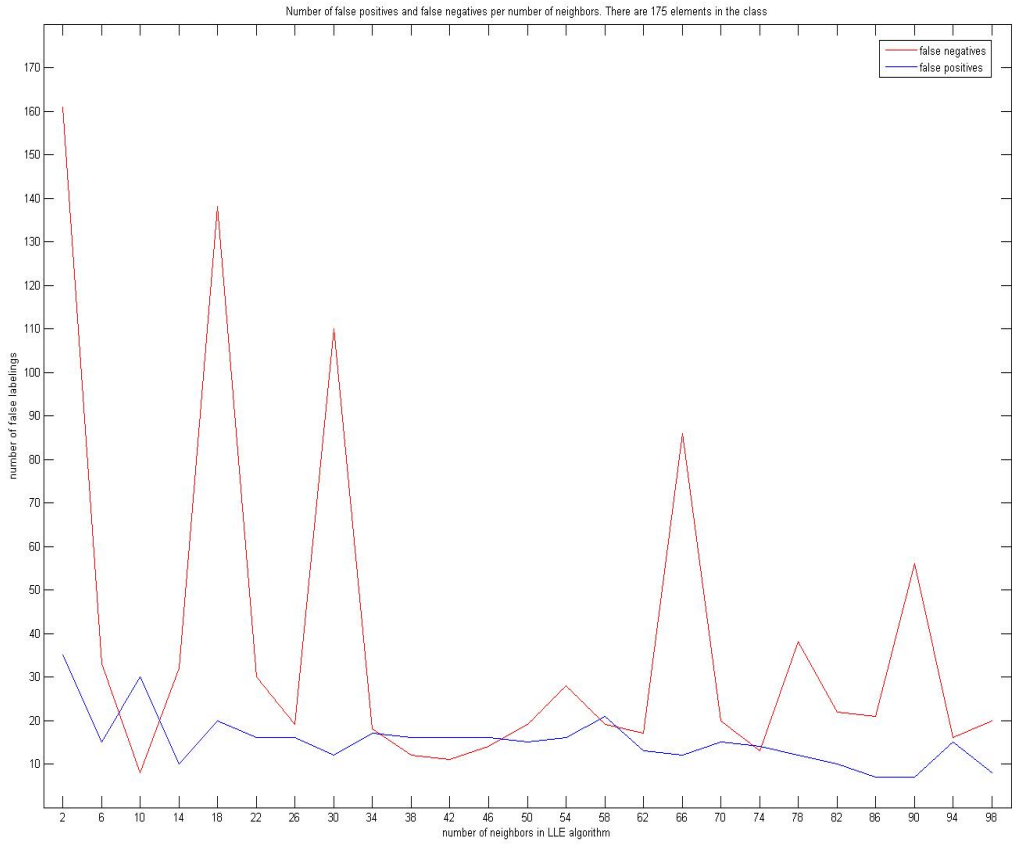


Figure 6.6: LLE false positive and negatives per number of neighbors

the above tile using LLE for the kernel construction with a neighbor setting of 10. The target dimension was four. Figures 6.9 (a), (b), (c), and (d) display the frame bands. The frame approach gives 12 false positives and 13 false negatives, Figure 6.9 (e). This is a slightly lower classification error rate than the other dimensionality reduction techniques.

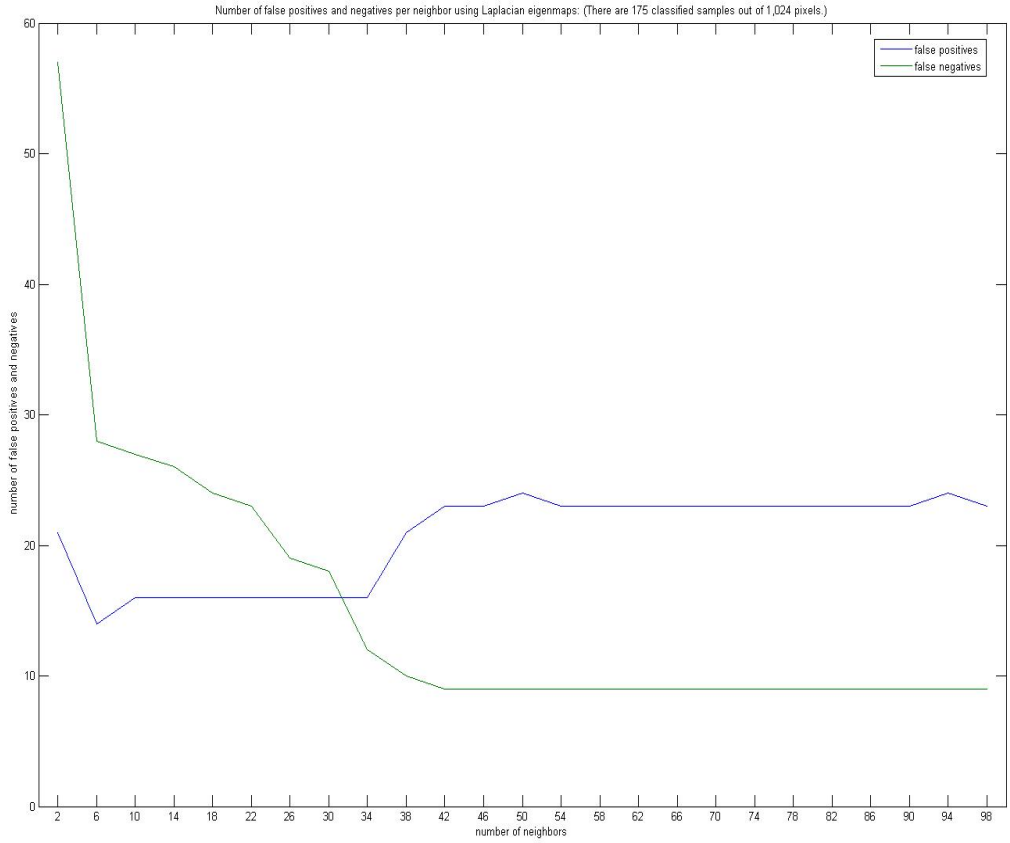


Figure 6.7: Laplacian eigenmaps false positives and negatives per number of neighbors

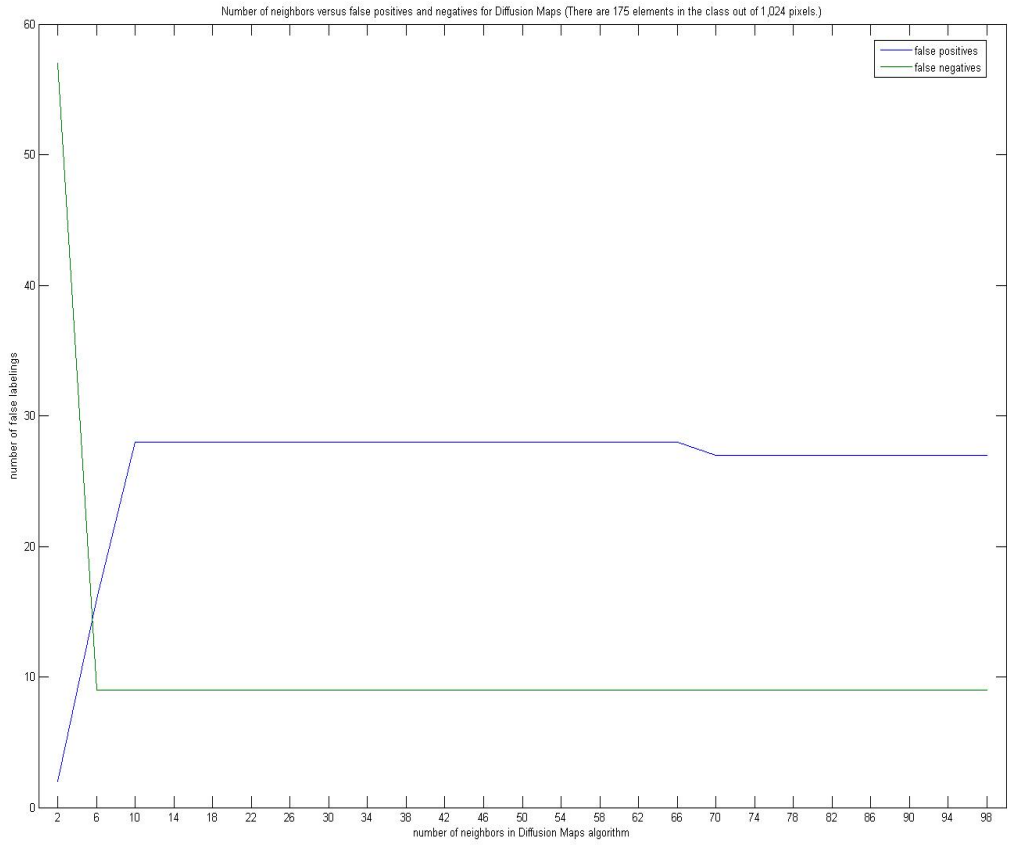
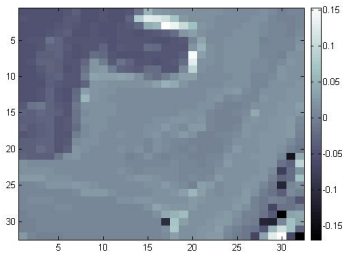
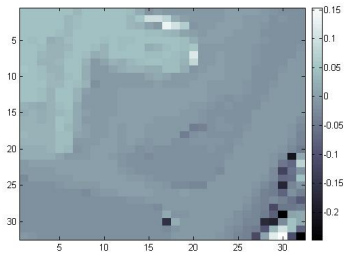


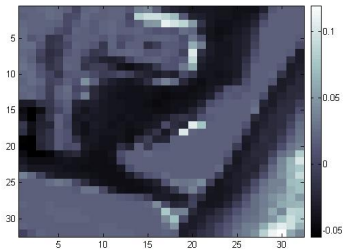
Figure 6.8: Diffusion Maps false positives and negatives per number of neighbors



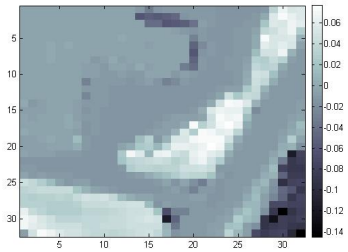
(a) frame band 1



(b) frame band 2



(c) frame band 3



(d) frame band 4

(e) frame classification

Figure 6.9: Four chosen Frame bands for example tile

Chapter 7

Geometric neighborhood analysis

Manifold kernel eigenmap methods such as Locally Linear Embedding (LLE) [41], Hessian LLE [22], Laplacian Eigenmaps [2], and Diffusion Wavelets [17] are popular techniques for performing dimensionality reduction on high dimensional data. Each of these techniques requires that a neighborhood parameter be set in order to construct the neighborhood for each data point. In particular, if the given data are $\{x_j\}_{j=1}^n$, there are the two standard approaches for determining the neighborhood of a data point, $N(x_i)$:

1. (k-nearest neighbors) Fix $k \in \mathbb{N}$, define $N(x_i)$ to be the set of k-closest data points to x_i, not including x_i. With this definition, neighborhoods are not necessarily symmetric, i.e., x_j could be a neighbor to x_i but x_i may not be a neighbor of x_j. It is possible to symmetrize the neighborhoods by simply defining x_j to be a neighbor of x_i if x_i is a neighbor of x_j. This is done after the k-closest neighbors are chosen, hence a neighborhood may have more than k elements.

2. (radius r) Set the radius, $r > 0$, define $N(x_i) = \{x_m : d(x_m, x_i) < r\} \setminus \{x_i\}$. Here d is the metric on the data points.

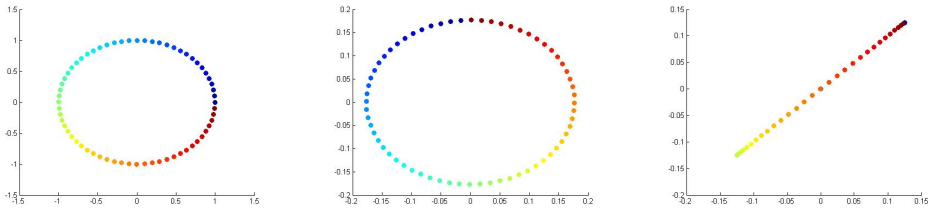
There are problems associated with both of these methods. In many cases, the data are not understood well enough to knowledgeably set the k-nearest neighbors

or radius parameters. For the k-nearest neighbors method, if k is chosen to be too large, then the neighborhood of x_i will contain points that should not be in there. Conversely, if k is chosen to be too small, then the neighborhood of x_i will not contain all of its neighbors. These same problems persist if a radius parameter is set. Choosing the correct, k, for the k-nearest neighbors parameter or radius parameter, r, is important for constructing the correct graph around x_i.

One of the main ideas behind manifold kernel methods is that if the correct graph structure for the data is chosen and an operator on this graph is constructed, e.g., Laplacian, then the eigenmaps of this operator give a good representation for the data. Dimensionality reduction follows by choosing a subset of the eigenmaps to represent the data. However, if the neighborhood parameter is set incorrectly, then the wrong graph will be constructed and the eigenmaps for operator on the graph will not be a good representation of the data. Consider a circle with sample 64 times. If the correct number of neighbors is chosen, $k = 2$, then the reconstruction using the first two eigenmaps give almost perfect reconstruction, Figure 7.1 (a),(b). The reconstruction rotates and rescales the data. However, when the number of neighbors is chosen to be 3, the eigenmaps fail to reconstruct the circle, 7.1 (c).

In what follows, I outline a geometric technique for constructing the neighborhood for a data point. The motivation behind this technique is that locally manifolds are flat like Euclidean space and the assumption that the data is sampled from a manifold. The manifold, M, is d-dimensional and embedded in the higher dimensional space $\mathbb{R}^D$ [1]. The neighbors of a data point, x_i, should be the points,

[1]Technically, the dimension of the manifold can vary from neighborhood to neighborhood.



(a) Original circle sampled 64 times

(b) Reconstructed circle using 2 neighbors

(c) Reconstructed circle using 3 neighbors

Figure 7.1: Choosing neighbors for the circle

$\{x_j\}_{j=1}^{k}$, such that the geodesic distance, $d_M(x_i, x_j)$ is approximately the same as the Euclidean distance, $d_E(x_i, x_j)$. Since we are dealing with data points sampled from a manifold, the geodesic distance between two points is approximated by the shortest path connecting the points. A path connecting points is constructed using a notion of what it means for a data point to be between two data points in higher dimensions.

Definition 18 (*between*)**.** *Let the distance between two points, x and y be denoted by $\rho(x, y)$. A point y is* between*(x, z) if $\rho(x, y) \leq \rho(x, z)$ and $\rho(y, z) \leq \rho(x, z)$.*

Note that this definition is symmetric in the sense that if y is *between*(x, z) then y is *between*(z, x). Figure 7.2 displays a two-dimensional region of potential between points. Also, this defintion is equivalent to the definition of *between* in 1 dimension, $x \leq y \leq z$.

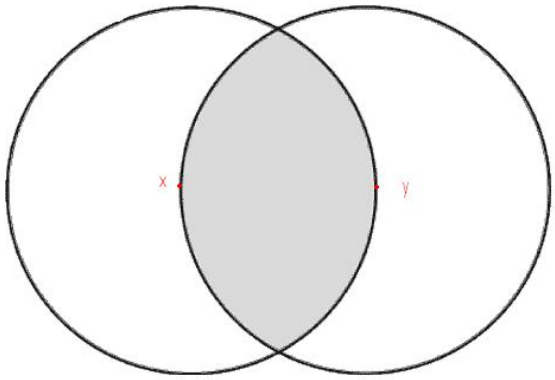


Figure 7.2: The shaded region is two-dimensional representation of points *between* x and y.

7.1 Algorithm

This is the concept behind computing the neighborhood structure of a data point. The actual implementation of the algorithm is different because of optimization.

Algorithm 19 (Flat neighborhood graph algorithm)**.** *Suppose data points* $\{x_i\}_{i=1}^n$ *and a metric,* ρ*, are given. Let the matrix* $A(x_j)$ *represent the local adjacency matrix around* x_j*. If the neighborhood graph algorithm determines that there should be an edge between* x_j *and* x_n*, then* $A(x_j)(j,n) = \rho(x_j, x_n)$*.* $A(x_j)$ *is constructed as follows:*

1. *Compute and sort in ascending order the distances between the other data points and* x_j*. Let* $\rho(x_{j_k}, x_j)$ *be the distance between* x_j *and* x_j*'s* k^{th}*-closest data point. Here,* $k = 1, \ldots, n-1$*.*

2. *Define* $A(x_j)(j, j_1) = A(x_j)(j_1, j) = \rho(x_j, x_{j_1})$*.*

Define $N(x_j) = \{x_{j_1}\}$.

3. *(For loop) For* $k = 2, \ldots, n-1$

   (3.1) Set $B = \{x_m \in N(x_j) : x_m$ *is* between$(x_j, x_{j_k})\}$.

   (3.2) If B *is empty then*

   $$A(x_j)(j, j_k) = A(x_j)(j_k, j) = \rho(x_j, x_{j_k})$$

   $$N(x_j) = N(x_j) \cup \{x_{j_k}\}.$$

   (3.3) $k = k + 1$

4. *Estimate the dimension of manifold,* d*, around* x_j*. This is done by performing principal components analysis on a set of points in a* D*-dimensional ball centered at* x_j*. The radius of the ball is distance from the last element in* N *to* x_j*.*

5. *Construct* d *linearly independent tangent vectors,* $\{v_1, \ldots v_d\}$*, at* x_j*. Let* $V =$ *span* $\{v_1, \ldots v_d\}$.

6. *Let* $\{x_{N_k}\}_{k=1}^{|N(x_j)|} = N(x_j)$*. For each* X_{N_k} *that is not approximately in* V*, remove the data point from* $N(x_j)$ *and set* $A(x_j)(j, N_k) = A(x_j)(N_k, j) = 0$.

Here are several points regarding the algorithm:

1. Steps 1-3 may construct too many neighbors for the data point. Data points that are not close to the tangent plane at x_j can be selected as neighbors of x_j. This is why steps 4-6 are used to remove these points.

2. There are many methods for estimating the dimension: Hausdorff, box counting, packing numbers. Principal component analysis with a setting that captures 97.5 percent of the variance has performed the best for my data [30].

3. Technically, step 6 should remove a neighbor, x_i, whose vector, v_i, is not in the tangent plane. However, the constraint of removing vectors that are orthogonal to the tangent plane, has performed better.

7.2 Results

Below are data sets on which the flat neighborhood graph algorithm has been run. The pixel x_j above is red and the neighboring pixels are solid blue. It is interesting to note that the k-nearest neighbors and the radius methods for neighborhood construction would give different results.

Figure 7.4 demonstrate the results of algorithm steps 1-3 being run on the Swiss roll. Note how there is an incorrect neighbor. Factoring in the tangent plance in steps 4-6 of the algorithm remove the incorrect neighbor.

The algorithm was applied to the hyperspectral data tile show in Figure 6.2 (a). The resulting graph differs significantly from that which would come from k-nearest neigbors or the radius method. The results demonstrate that different data points may have a different number of neighbors and that there is no relationship between class and the number of neighbors. Figure 7.5 (a) gives a histogram of the number of neighbors for the data points. Figure 7.5 (b) can be compared with 6.2 (a) to demonstrate the lack of relationship between class and number of neighbors.

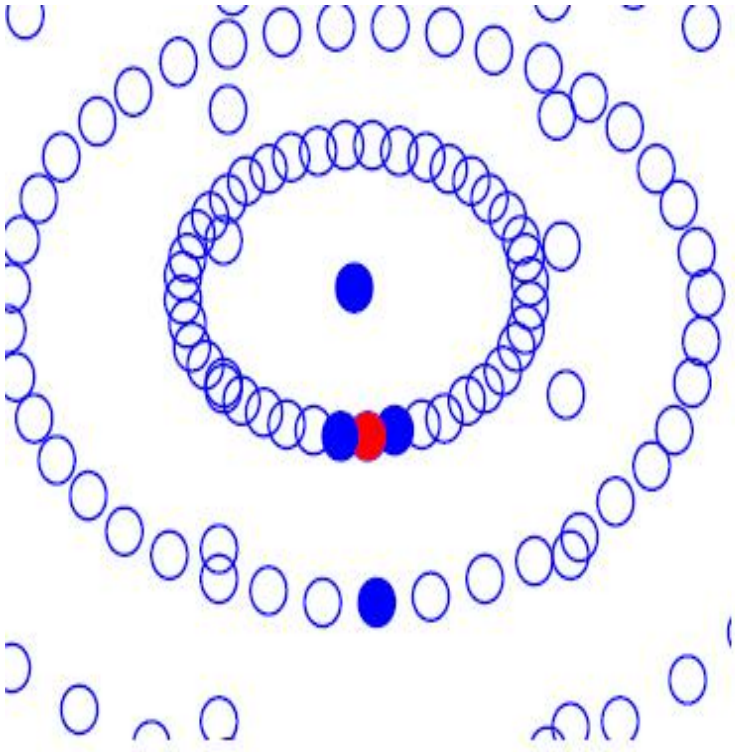


Figure 7.3: Neighborhood for a point on the sphere.

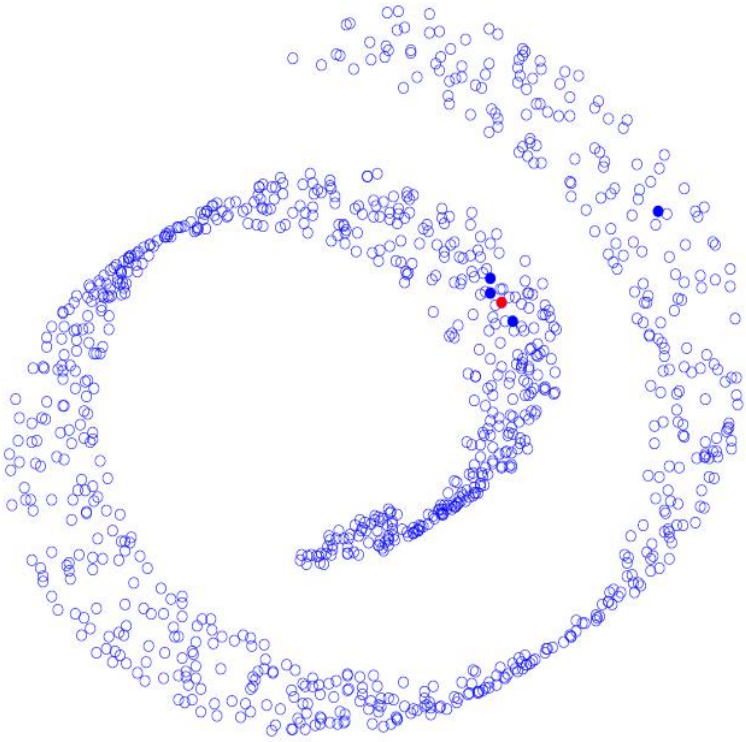
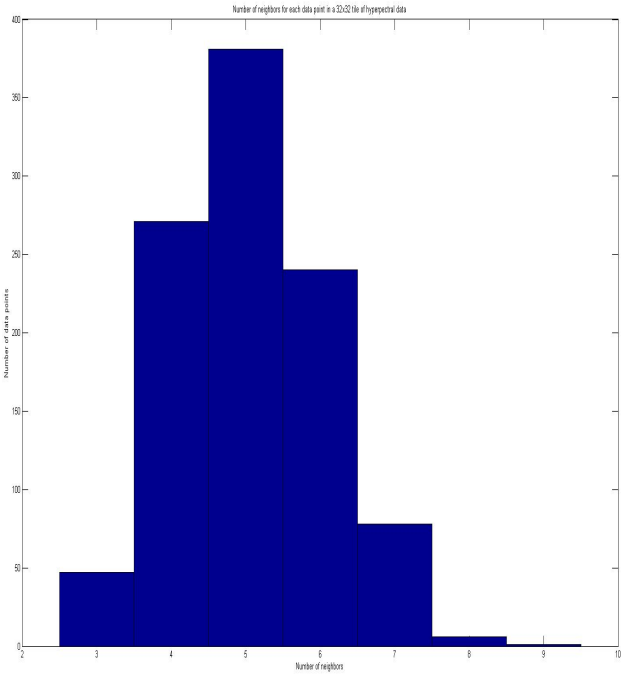
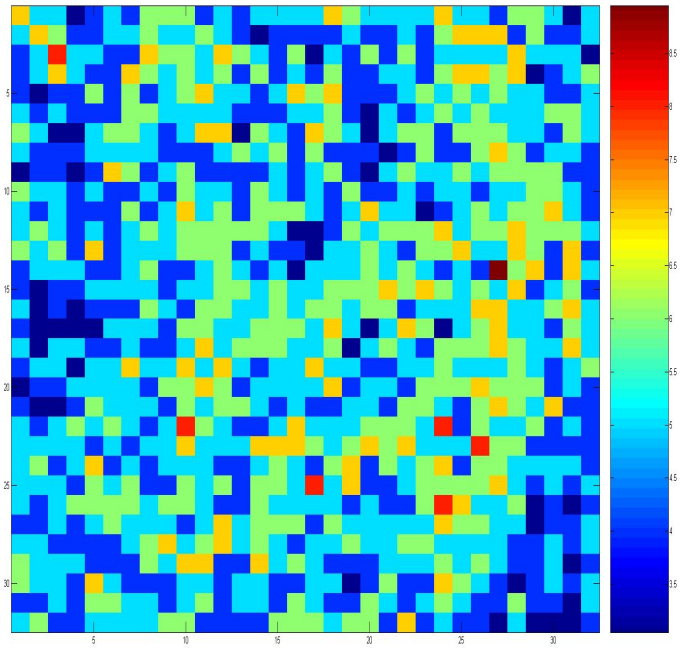


Figure 7.4: Incorrect neighborhood for a point on the Swiss roll.



(a) Histogram of the number of neighbors



(b) Number of neighbors per data point

Figure 7.5: Hyperspectral neighbors

Chapter 8

Representation systems for urban terrain elevation data

Throughout the year of 2006 and until June of 2007, I worked as a research assistant for the Norbert Wiener Center (NWC) on a project for the United States Army Corps of Engineers' Topographic Engineering Center. Below is a summary of work conducted jointly with John Benedetto, Chris Flake, Ioannis Konstantinidis, Diego Maldonado, and Jeff Sieracki.

8.1 Introduction

Light detection and ranging (LIDAR) data is collected by using a laser pulse and a sensor onboard an airplane. The airplane passes over an area of interest and measures the distance to the first object on the laser pulse's path. The collected data is a vector with three components, (x, y, z), where (x, y) is the location on the ground and z is the height of the object above the ground[1]. Figure 8.1 demonstrates the swaths of data collected by an airplane passing over Fort Belvoir.

The LIDAR data, which is also known as point cloud data, is gridded. The output of this gridding is a data elevation matrix (DEM), M, in which the (x, y) position in the matrix is the ground coordinate and $M(x, y)$ is the height of the

[1]Technically, LIDAR measures the distance from the airplane to the object. However, the difference of the height of the airplane above ground level and the measured distance of the LIDAR pulse is the z value.

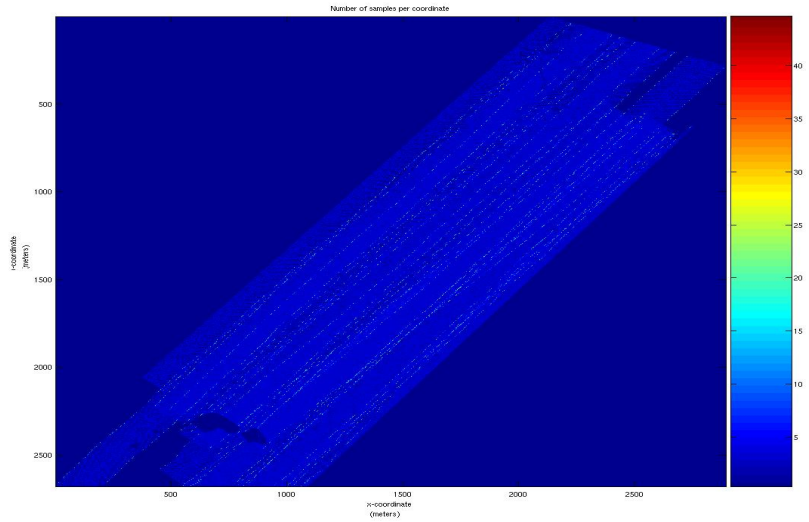


Figure 8.1: Airplane swaths and LIDAR coordinates for data collection over Fort Belvoir (Courtesy U.S. Army Corps of Engineers)

object at (x, y). Figure 8.2 demonstrates the point cloud data overlayed with the gridded data.

There are many applications of LIDAR data such as cartography, 3-dimensional cartography, and feature discrimination. Sometimes LIDAR data is coupled with hyperspectral data to help identify the material on the ground. An example of feature discrimination is line of sight. Line of sight allows a person on the ground to know from which buildings they can be seen and targeted. Although these applications are nice, there are some problems associated with LIDAR data. One of the main problems is file size. Currently, the data sets are so large that they cannot be transmitted in a reasonable amount of time. As a consequence of this, research is being done to determine optimal representation systems for compressing LIDAR data sets.

The goal of this project was to determine how well some representations sys-

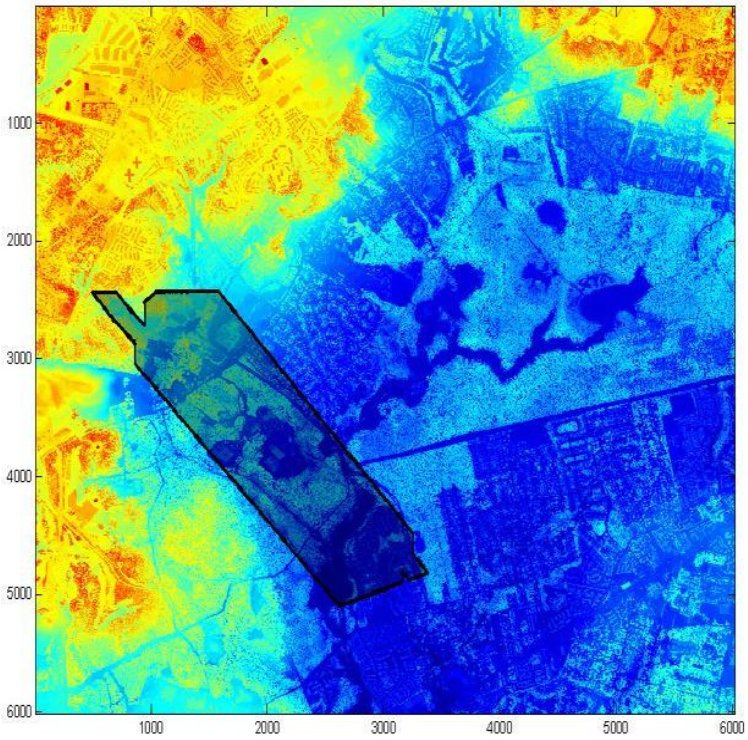


Figure 8.2: Area of point cloud data overlayed on the DEM (Courtesy U.S. Army Corps of Engineers)

tems, that are well-known for images, perform on LIDAR data. Standard representations such as JPEG and JPEG2000 have been used along with lesser known filter banks that have a directional component such as curvelets, contourlets, and ridgelets. Directional filter banks capture the edges of buildings, roads, and other features that are important in an urban terrain battle environment. Another goal of the project was to determine metrics for measuring the success of compression results. Because in many applications the enduser is a person, it is important to have metrics that are comparable to the human eye.

8.1.1 Contributions and results

My contributions and results are briefly summarized here.

1. LIDAR data sets are not images. Even though the DEM that comes from LIDAR data looks like an image, it behaves differently under standard imaging techniques. This may motivate someone to develop a representation system that is "tailored" for LIDAR data.

2. A modified version of the SSIM index for LIDAR data corresponds better to the human eye than other commonly used metrics. This result is not surprising, there is an analagous result for images. This means that SSIM is one of the best quantitative measures of how well a compressed reconstruction looks to the human eye. This allows for varying the rate of compression based on the result of the SSIM index.

3. Contourlets with a 9/7 filter bank compressed the data the best with respect to the Terrain structural similarity index (TSSIM). However, the directional aspect of contourlets was not used. Depending on the DEM, either a one-level or two-level contourlet transform gave the optimal results.

8.2 The data

We consider two LIDAR data sets: a 5100 $\times$ 5100 meter square tile of terrain elevation data from the city of New Orleans and a 6019 $\times$ 6019 meter square tile of terrain elevation data from Fort Belvoir. Both tiles are 1-Meter DEM and

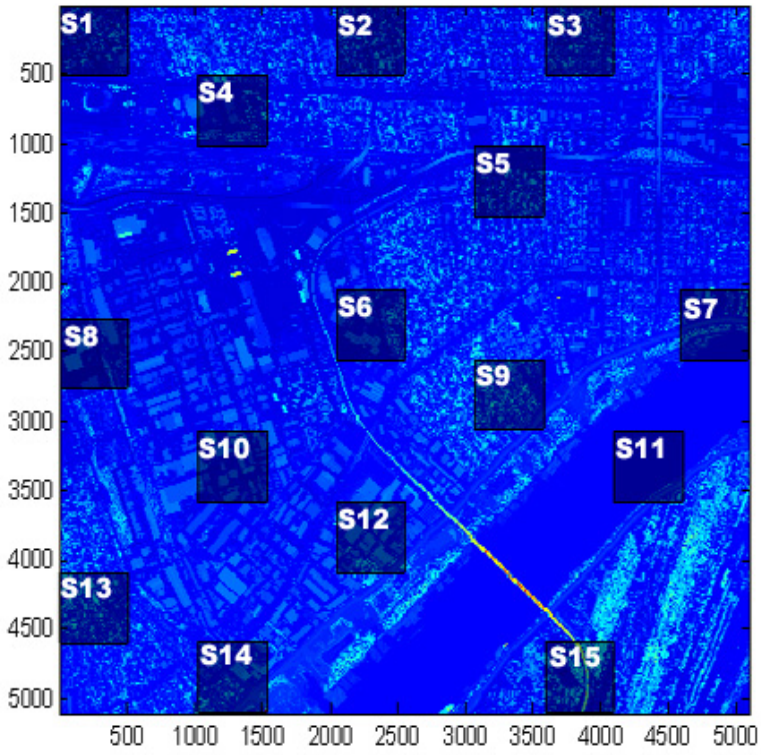


Figure 8.3: Tiling the 5100 × 5100 New Orleans DEM. Fifteen subtiles were chosen, S1-S15. (Courtesy U.S. Army Corps of Engineers)

were obtained from first return LIDAR scanning. For computational optimality and subsequent statistical analysis, these tiles were segmented into subtiles of size 512 × 512. The segmentation was chosen to be representative of different local terrain types such as industrial areas, urban areas, water and suburban residential neighborhoods of varying tree density. See Figure 8.3 and Figure 8.4.

8.3 Representation systems

Five representation systems were studied. These are:

- The Discrete Cosine Transform (DCT)

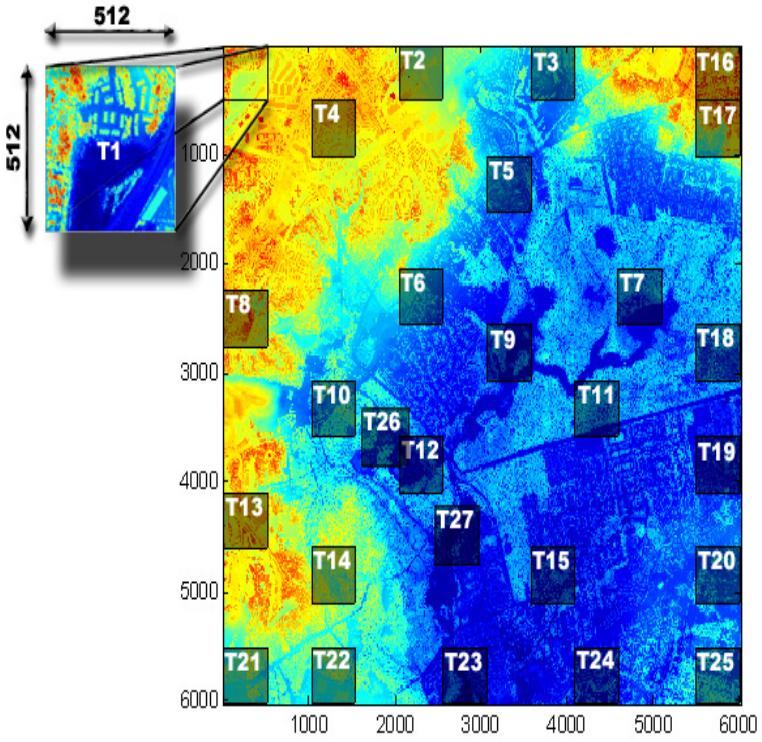


Figure 8.4: Tiling the 6019 × 6019 Fort Belvoir DEM. Twenty-five subtiles were chosen, T1-T25. (Courtesy U.S. Army Corps of Engineers)

- 5/3 and 9/7 Discrete Wavelet Transforms (DWT)
- Curvelets
- Contourlets
- Ridgelets

The DCT is one of the main components of JPEG compression and was selected as a natural comparison to determine how standard image processing techniques would fare when applied to DEMs. As described below, the DCT differs from more modern representations in that it is a time-frequency rather than time-scale decomposition. The 5/3 and 9/7 DWTs have also been extensively applied in image

processing and are associated with the JPEG2000 standard. With regard to JPEG and JPEG2000, we note that our analysis is with respect to the underlying mathematical foundation of these methods and does not consider the quantization and coding that are used in these standards.

Curvelets, contourlets, and ridgelets comprise new, generalized transforms that can be combined with any of the filter banks used in the standard DWT. These methods are not in common usage nor are they part of any engineering standards. These methods were explored with different parmeters and configurations; some of which are discussed below.

8.3.1 Discrete Cosine Transform

For our DCT representation we used the standard MATLAB 2-dimensional Discrete Cosine Transform. This transformation can be described as follows: Given a natural number n, define the $n \times n$ matrix T by

$$T_{ij} = \begin{cases} \frac{1}{\sqrt{n}} & \text{if } i = 0, \\ \\ \sqrt{\frac{2}{n}} \cos\left(\frac{(2j+1)i\pi}{2n}\right) & \text{if } i > 0, \end{cases}$$

where $i, j = 0, \ldots, n-1$. Let A be an $n \times n$ matrix, i.e, $A = \{a_{ij}\}_{i,j=0}^{n-1}$. Then the Discrete Cosine Transform of A is the matrix $D = T^t A T$ (here T^t is the transpose of T). The DCT is the real-valued version of the Discrete Fourier Transform (DFT). As such, D codes the frequency behavior of the elements of A. The upper-left quarter of D codes the Low-Low frequency (coarser scale) aspects of A, while the lower-right quarter of D codes the High-High frequency (fine scale) aspects of A.

The fundamental fact behind the applications of these types of transformations is that the naked eye is more sensitive to variations of the low frequency coefficients than those of the high frequency coefficients. Thus, the physiology of the eye allows for ad-hoc manipulations in the elements of the matrix D. If $\tilde{D}$ represents D after these manipulations, the Inverse DCT is performed to obtain a matrix $\tilde{A}$ close (in an appropriate sense) to the original matrix A. Since T is an orthogonal matrix, i.e., $TT^t = Id$, the Inverse DCT is also easily implemented. The DCT lies in the core of the JPEG compression standard, which makes it a strong reference point.

8.3.2 Discrete Wavelet Transform

As shown above, the DCT is based on the Cosine function. The periodicity of this function implies its lack of localization. In other words, the Cosine funtion is "all over the place". As a consequence, if a matrix A is perturbed or modified in a few of its entries, then *all* of its DCT coefficients will be notably affected by such perturbation. Thus, a *local* perturbation in A generates a *global* response in its DCT. This is an inherent drawback of the DCT.

In order to circumvent the local-global perturbation scheme, classes of *well-localized* functions were considered as bases of the spaces of interest (mostly L^2). These classes are obtained by translations and dilations of a fixed profile function called a *wavelet*.

The DWT is implemented by means of filter banks. The mathematical equivalence between the DWT, the Multi-resolution Approximations (MRA), and the

Filter Bank theory has been one of the major accomplishments in Wavelet Theory [33]. This theory has been developed during the last two decades and it has seen an enormous range of applications.

A *scaling function* φ, also called the father wavelet, is a function whose dilations and translations form an orthonormal basis of an approximating space $V_j \subseteq L^2(\mathbb{R})$.

$$V_j = span\{2^{-j/2}\varphi(2^{-j}(t-n))\}_{n\in\mathbb{Z}}$$

A wavelet ψ is a function of zero mean, that can be derived from φ, such that the family

$$\{2^{-j/2}\psi(2^{-j}(t-n))\}_{(j,n)\in\mathbb{Z}^2}$$

is an orthonormal basis of $L^2(\mathbb{R})$. These approximating spaces, V_j, are the core of the multi-resolution approximations. Given a signal $x \in L^2(\mathbb{R})$, its approximation at scale j is given by

$$A_j(x)(t) = \sum_{n\in\mathbb{Z}} a^{(j)}[n]2^{-j/2}\varphi(2^{-j}(t-n)),$$

where, given the integers j and n, the n-th approximating coefficient of x at scale j is

$$a^{(j)}[n] = \int x(t)2^{-j/2}\varphi(2^{-j}t-n)\,dt.$$

Assuming that J is the finest scale, we can approximate $a^{(J)}[n]$ by the samples $x[n]$. Next, x will be decomposed into an approximating signal and a detail signal. This is done by means of filter banks. A filter is just a finite sequence $\{s[n]\}_{n=-N}^{N}$ that will act as a convolution factor. Wavelets have naturally associated a couple of filters

$\{h[n]\}_{n=-N}^{N}$ and $\{g[n]\}_{n=-N}^{N}$. The filter g is related to φ and is called a low-pass filter, while h is related to ψ and it is called the high-pass filter. The coefficients for the approximating and detail signal are given, respectively, by

$$a^{(J+1)}[n] = \sum_{m=-N}^{N} g[m-2n]a^{(J)}[m]$$

and

$$d^{(J+1)}[n] = \sum_{m=-N}^{N} h[m-2n]a^{(J)}[m].$$

In order to reconstruct $x = A_J(x)$ we retrieve the coefficients $a^{(j)}[n]$ by computing

$$\begin{aligned} a^{(J)}[n] &= a^{(J+1)}[n] + d^{(J+1)}[n] \\ &= \sum_{m=-N}^{N} g[n-2m]a^{(J+1)}[m] + \sum_{m=-N}^{N} h[n-2m]a^{(J+1)}[m]. \end{aligned}$$

The key aspect of this approach is that it can be iterated by using the approximation signal as a base signal. Moreover, the subsequent wavelet coefficients are obtained recursively. Namely, when k goes from $J+1$ up to some limit value M (that depends on the size N), we compute

$$a^{(k+1)}[n] = \sum_{m=-N}^{N} g[m-2n]a^{(k)}[m]$$

and

$$d^{(k+1)}[n] = \sum_{m=-N}^{N} h[m-2n]a^{(k)}[m].$$

The coefficients $a^{(k)}[n]$ represent the approximation the signal x at scale k, while the coefficients $d^{(k)}[n]$ represent the details of the signal at scale k.

The DWT is a decomposition method that replaces the time-frequency analysis (as in the DCT) by a time-scale analysis. The signal x can be retrieved from the

coarsest approximation and the detail coefficients in a similar computational way, i.e., by means of filters and up-sampling.

Although this description deals with a one-dimensional signal x, in the case of image processing (two-dimensional signals) all of the above is iteratively applied to the (one-dimensional) rows and columns of the images to analyze.

We paid special attention to two classes of filters: the 9/7 and 5/3 filters. In the case of the 9/7 filter protocol we have

$$\begin{aligned} g = & [.037828455506995, -.023849465019380, -.11062440441842, \\ & .37740285561265, .85269867900940, .37740285561265, \\ & -.11062440441842, -.023849465019380, .037828455506995] \end{aligned}$$

and

$$\begin{aligned} h = & [-.064538882628938, -.040689417609558, .41809227322221, \\ & .78848561640566, .41809227322221, -.040689417609558, \\ & -.064538882628938] \end{aligned}$$

In the case of the 5/3 filter protocol we have

$$g = \frac{1}{4\sqrt{2}}[-1, 2, 6, 2, -1]$$

and

$$h = \frac{1}{2\sqrt{2}}[1, 2, 1]$$

The Discrete Wavelet Transform using the 9/7 filter is the main component of the lossy JPEG2000 compression standard, while the 5/3 filter is the main com-

ponent of the lossless JPEG2000 compression standard. This makes 9/7 and 5/3 filters a state-of-the-art reference point.

For the 5/3 and 9/7 wavelet filter bank decompositions we used the software implementation from the DIPUM toolbox. Available at http://www.imageprocessingplace.com DIPUM stands for Digital Image Processing Using MATLAB.

8.3.3 Discrete curvelet transform

The Discrete Curvelet transform is a recently developed two-dimensional multi-scale transform that generalizes the Wavelet transform. It has the advantage of representing images at different scales and different angles. The Discrete Curvelet Transform has strong directional character (angle sensitivity) in which elements are highly anisotropic at fine scales, with effective support shaped according to a certain parabolic scaling. It was introduced by Emmanuel Candes and David Donoho [8, 12, 11, 10, 9].

The one-dimensional Wavelet transform is very sensitive to point singularities of one-dimensional signals. In the two-dimensional Wavelet transform the frequency plane is decomposed into dyadic rectangles whose axis are parallel to the coordinate axis. This makes the two-dimensional Wavelet transform sensitive to singularities along horizontal, vertical, and diagonal lines. Since the singularities of an image are its edges, the two-dimensional Wavelet transform has a good response (sparse representation) when the images considered have horizontal, vertical, or diagonal edges. However, when the edges are more general geometrical curves, the DWT

needs many coefficients to describe those edges.

In the Curvelet transform the frequency plane is decomposed into dyadic annuli, these annuli are in turn split into wedges in such a way that the length of the base of each wedge is the square root of the width of the corresponding annulus (parabolic scaling).

Each wedge w supports the Fourier transform of a smooth function ψ_w. The translations of ψ_w do not form an orthonormal basis of L^2 anymore, but they are still a good generating set. Namely, the family

$$\{\psi_w(t_1 - n_1, t_2 - n_2)\}_{w,(n_1,n_2)\in\mathbb{Z}^2}$$

is a *frame* for L^2. We can decompose now any signal x as

$$x(t_1, t_2) = \sum_{w,n_1,n_2} a[w, n_1, n_2]\psi_w(t_1 - n_1, t_2 - n_2),$$

where the coefficients $a[w, n_1, n_2]$ are easily computed by means of inner products.

The Discrete Curvelet Transform algorithm has been coded by the Curvelet.org team: E. Candes, L. Demanet, D. Donoho, and L. Yingis. Available, after free registration, at

http://www.curvelet.org/software.html.

8.3.4 Discrete contourlet transform

The Discrete Contourlet Transform, introduced by Minh-Do and Martin Vetterli in [18] and [19], shares the same tiling principle as in the Discrete Curvelet Transform. However, in this case pyramidal directional filter banks are shown to

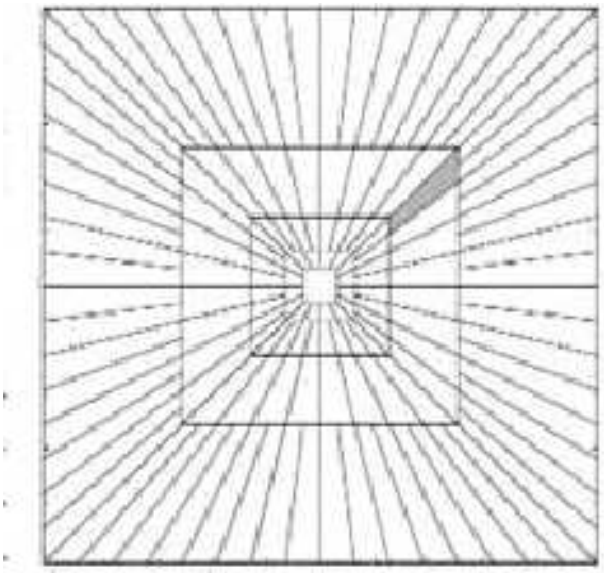


Figure 8.5: Tiling of the frequency plane in the Contourlet Transform

provide an effective method to implement the digital curvelet transform [20]. Also, instead of using annuli, the frequency plane is tiled using sheared rectangles, Figure 8.5.

The Contourlets decomposition algorithm has been coded by Minh-Do and it is available at

http://www.ifp.uiuc.edu/ minhdo/software.

8.3.5 Discrete ridgelet transform

The Ridgelet transform was introduced by Candes and Donoho in 1999 [6, 7] as a new multi-scale representation scheme for images that are smooth away from discontinuities along lines. In this aspect, they improve on the performance of the two-dimensional Wavelet transform by combining wavelets and the Radon transform.

The approximation system called ridgelets in the pioneering work by Candes (1997, 1998) refers to a ridge function

$$\psi_{a,b,\theta}(t_1, t_2) = a^{1/2}\psi(a(t_1\cos(\theta) + t_2\sin(\theta)) - b),$$

where ψ is a wavelet. In order to obtain a method of series representation, Candes constructs 'ridgelet frames', see [6, 7]. This representation is sparse when it describes images with linear edges.

The Finite Ridgelet Transform (FRIT) was created by Minh-Do and Martin Vetterli [21]. The FRIT adapts the ideas of Candes and Donoho to the case of discrete signals. Also, the MATLAB code for the FRIT has been written by Minh-Do and it is available at

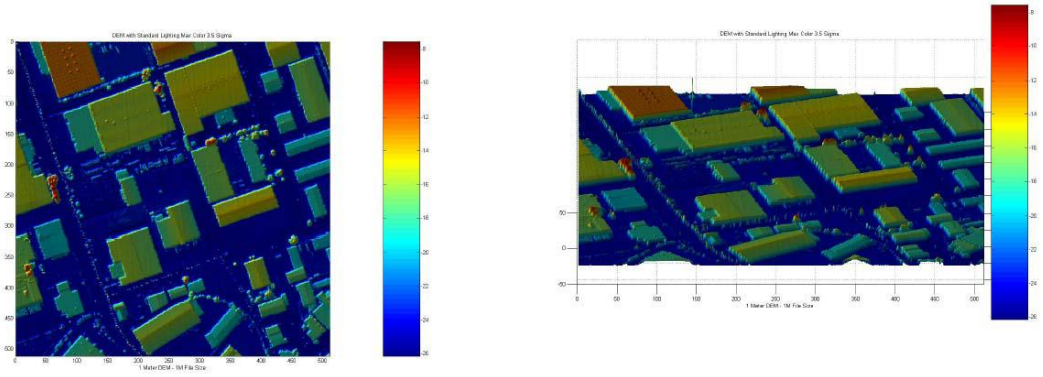
http://www.ifp.uiuc.edu/ minhdo/software/

8.4 Reconstructions

During this project many DEMs were transformed using one of the above transforms, the coefficients were then threshed, and the reconstruction DEMs were created. Many of the transforms have several parameters that can be set and this, coupled with the fact that there are many tiles and different ways of threshing, generated a lot of data. There is too much data to display here; the reconstructions below are just a small sampling of what was generated.

The tile S10 from the New Orleans LIDAR data, see Figure 8.6, was transformed. This tile was chosen for its urban properties.

Below is a series of reconstructions of S10 using twenty-five percent of the orig-



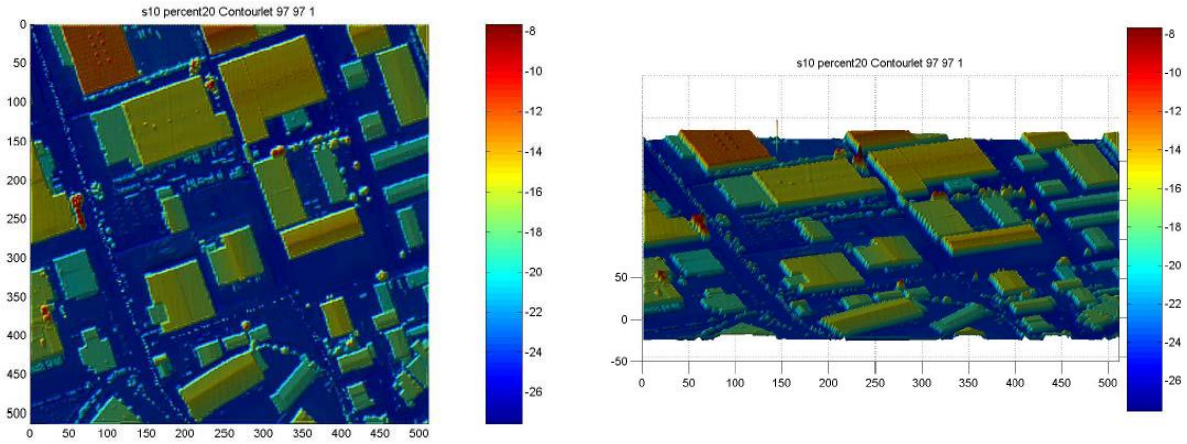
(a) (b)

Figure 8.6: The original S10 tile, (a) top-down view (b) view with 25 degree angle of elevation.

inal coefficients. Several of the transforms are redundant, therefore it is important to use the original number of coeffients when comparing transforms. It is interesting to note the different artifacts given by the various transforms. Also, sometimes a reconstruction will look acceptable from a top-down view; however, when viewed at an elevation artifacts can be seen. Also, please note that one can see in the reconstructions a spike towards the upper, left corner. This is not an artifact; this spike is in the original data as well.

8.5 Metrics and SSIM

Several norms were used in comparing the reconstructed DEMs against the original DEMs. These norms included the ℓ^1-norm, ℓ^2-norm, ℓ^∞-norm, and the



(a) (b)

Figure 8.7: The S10 tile reconstruction using the contourlet transform with a 9/7 filter bank and no directional component, (a) top-down view (b) view with 25 degree angle of elevation.

total variation semi-norm. The SSIM index is not a norm; however, it gave better visual results than the other norms.

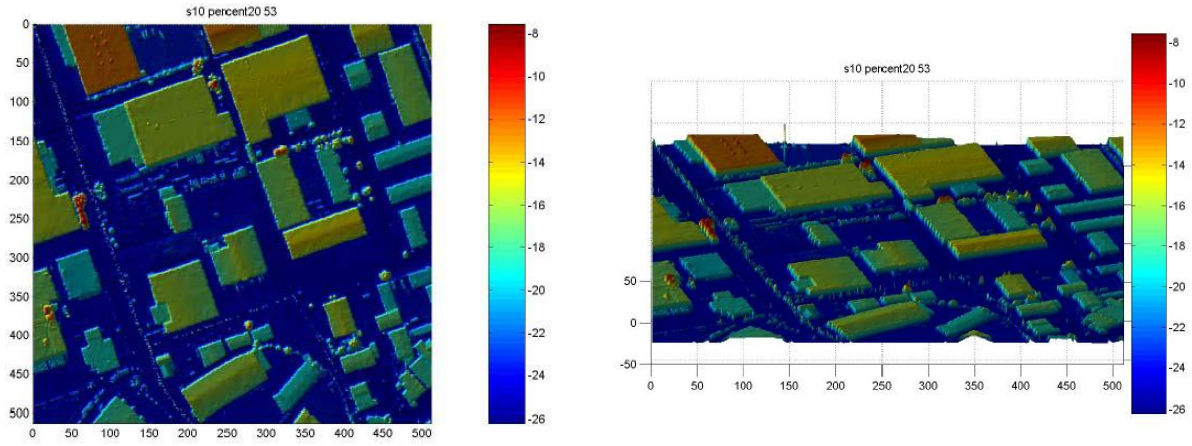
Briefly, here are defintions of the norms and the SSIM index.

The ℓ^p-distance between two DEMs $A = \{a_{ij}\}_{i,j=1}^N$ and $B = \{b_{ij}\}_{i,j=1}^N$ comes from generalizations of the Euclidean distance. Given $p \geq 1$ we define the ℓ^p-distance between A and B is

$$\|A - B\|_{\ell^p} = \sup_{x:\|x\|_{\ell^p}=1} \|(A - B)x\|_{\ell^p}$$

Here $x = \{x_n\}_{n=1}^N$ is a vector and

$$\|x\|_{\ell^p} = \left(\sum_{n=1}^{N} |x_n|^p \right)^{1/p}.$$



(a) (b)

Figure 8.8: The S10 tile reconstruction using the DWT with the 5/3 filter bank, (a) top-down view (b) view with 25 degree angle of elevation.

If $p = \infty$ we write

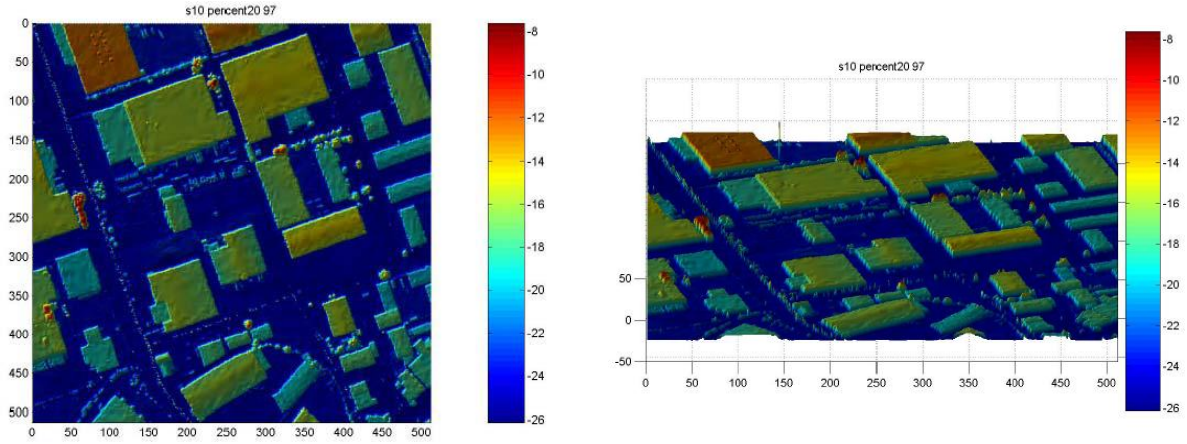
$$\|x\|_{\ell^\infty} = \sup_{n=1}^{N} |x_n|.$$

In the case $p = 2$, this is the usual Euclidean distance. Basic results of numerical analysis allow the ℓ^1-norm to be computed easily by considering the maximum column sum, where the column sum is the sum of the magnitudes of the elements in a given column. An analagous result holds for the ℓ^∞-norm, only with column replaced by row.

The Total Variation (TV) semi-norm finds many applications in image processing. Given a signal $f(t)$, its TV semi-norm is given by

$$\|f\|_{TV} = \int |\nabla f(t)| dt.$$

This is just the ℓ^1-norm of the derivative of f. In practice, for a vector, the



(a) (b)

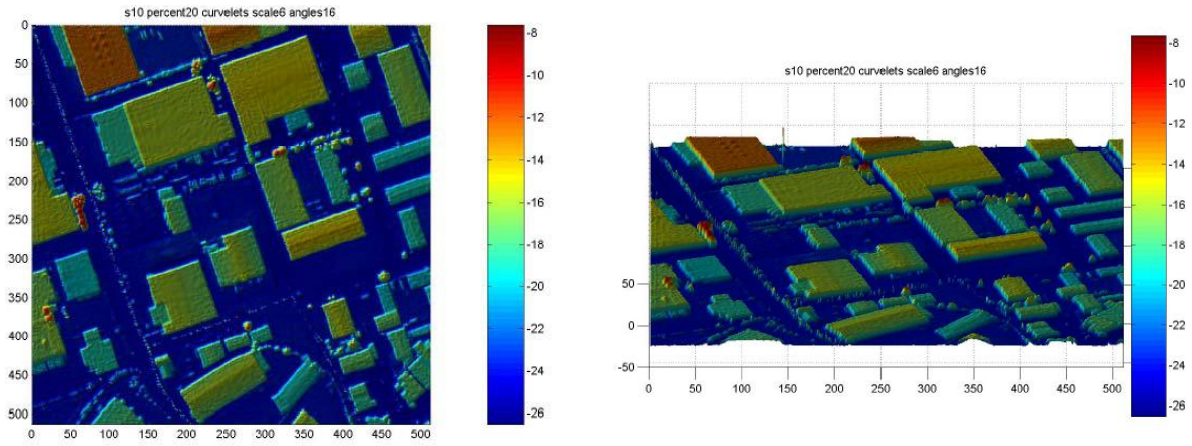
Figure 8.9: The S10 tile reconstruction using the DWT with the 9/7 filter bank, (a) top-down view (b) view with 25 degree angle of elevation.

derivative is quantized giving

$$\|f\|_{TV} = \sum_{i=1}^{n-1} |f(x_{i+1}) - f(x_i)|.$$

For a DEM the total variation is taken over each row and column and then summed.

The SSIM index is slightly more involved. The SSIM index was first introduced in the image processing literature [48, 49, 50, 47] as an alternative to norm-based comparison measures. It appears to better approximate judgments of similarity by the human eye on natural images. It combines three different quality index: luminance l, contrast c, and correlation s that are applied locally over a sliding window and then averaged across all pixels in the image. The SSIM index is a statistical measure. It is computed based on the moments of the pixel value distribution. Assume that X and Y are $n \times n$ images embedded in $\mathbb{R}^{n^2}$. An $M \times M$ window



(a) (b)

Figure 8.10: The S10 tile reconstruction using the curvelet transform with a 9/7 filter bank, (a) top-down view (b) view with 25 degree angle of elevation.

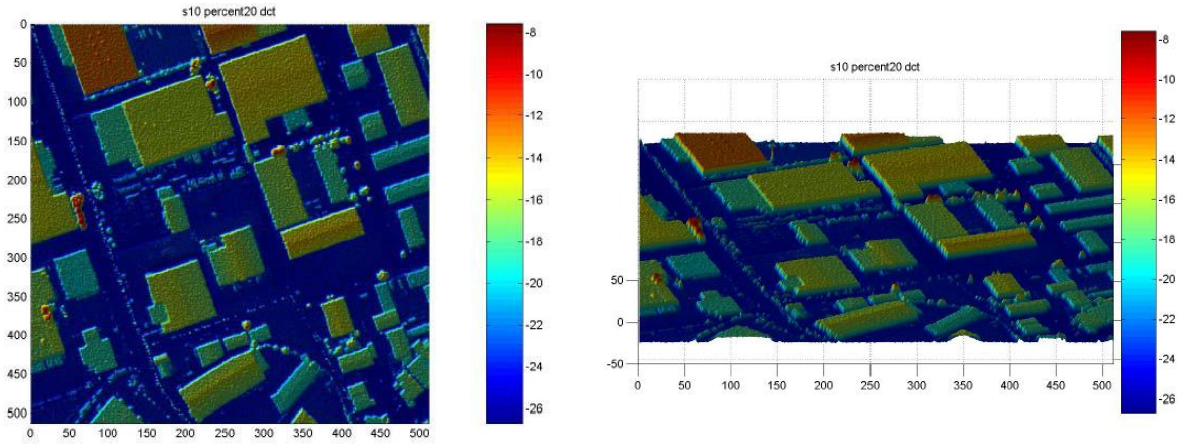
(sub-image) of the image X can be realized as a vector of dimension $N = M^2$. Let $a = \{a_i\}_{i=1}^N$ and $b = \{b_i\}_{i=1}^N$ be the pixels of X and Y corresponding to this window, centered on the target pixels of X and Y, respectively. First, given a weight w, the weighted means $\mu_a = \sum_{i=1}^N w_i a_i$ and $\mu_b = \sum_{i=1}^N w_i b_i$ are computed. The luminance comparison between the two target pixels is defined by

$$l(a,b) = \frac{2\mu_a\mu_b + C_1}{\mu_a^2 + \mu_b^2 + C_1},$$

where C_1 is included for numerical stability. Next the variances $\sigma_a = \sum_{i=1}^N w_i a_i^2 - \mu_a^2$ and $\sigma_b = \sum_{i=1}^N w_i b_i^2 - \mu_b^2$. Contrast is now defined by

$$c(a,b) = \frac{2\sigma_a\sigma_b + C_2}{\sigma_a^2 + \sigma_b^2 + C_2}.$$

The third component, correlation, is based on the cross-correlation $\sigma_{a,b} = \sum_{i=1}^N w_i a_i b_i -$



(a) (b)

Figure 8.11: The S10 tile reconstruction using the DCT, (a) top-down view (b) view with 25 degree angle of elevation.

$\mu_a\mu_b$, and defined by

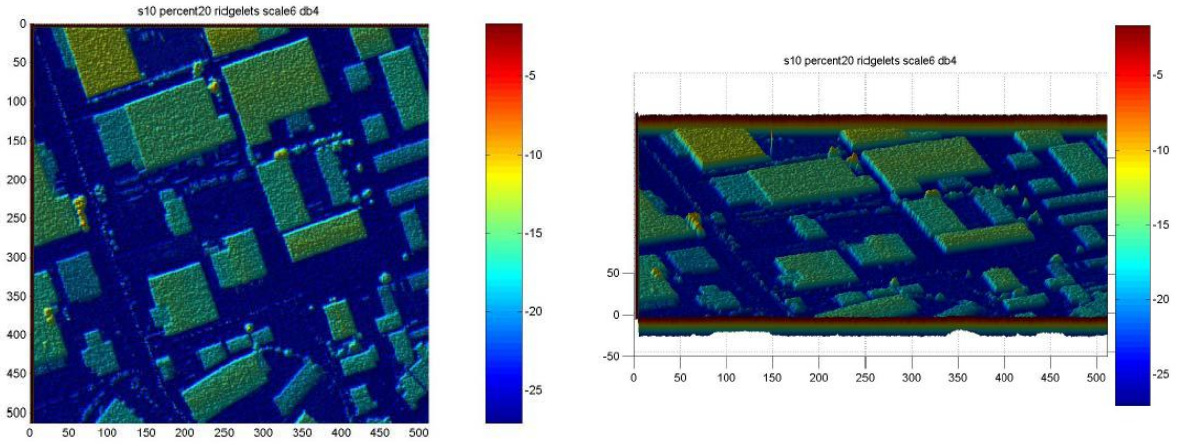
$$s(a,b) = \frac{\sigma_{a,b} + C_3}{\sigma_a\sigma_b + C_3}.$$

Again, C_2 and C_3 avoid small denominators. Now set

$$SSIM(a,b) := l(a,b) \times c(a,b) \times s(a,b)$$

In our experiments we fixed w to be an 11×11 Gaussian weight and used a MATLAB implementation of the SSIM adapted from the one by Z. Wang. This adaptation is based on the fact that we chose to discard the luminance component, considering that absolute mean elevation differences pay little role in evaluating the similarity of urban terrain DEMs. The index thus modified is referred to as TSSIM (*Terrain* SSIM) and, in a given window, it takes the form

$$TSSIM(a,b) = \frac{\sigma_{a,b} + C}{\sigma_a^2 + \sigma_b^2 + C}$$



(a) (b)

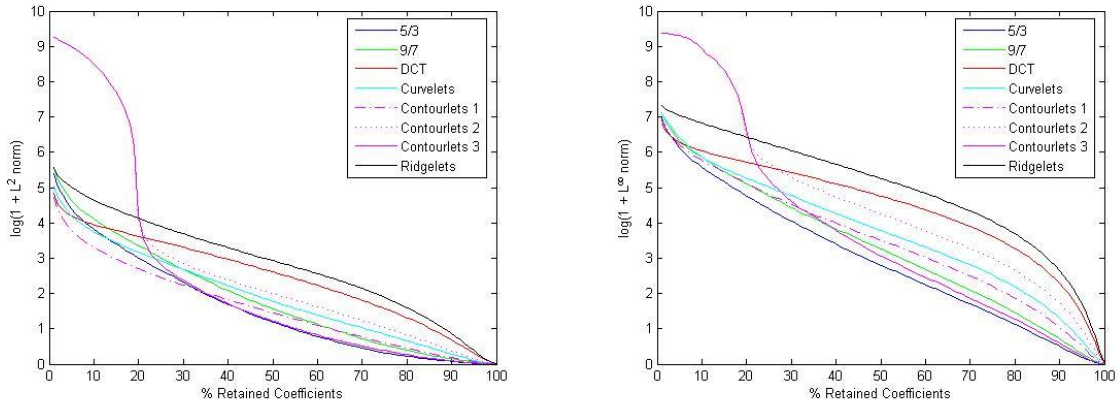
Figure 8.12: The S10 tile reconstruction using the finte ridgelet transform, (a) top-down view (b) view with 25 degree angle of elevation.

where C is adapted to the dynamic range of the images under consideration. Finally, TSSIM(X, Y), the TSSIM index between X and Y, is obtained as the average of the TSSIM$(a, b)'s$ over the sliding window. Clearly, the TSSIM is not a distance. TSSIM is a number between 0 and 1, the closer (*visually*) two images X and Y are from one another, the closer TSSIM(X, Y) is to one.

8.6 Quantitative results

The above transforms were performed on many DEMs. The norms of the difference between the original and reconstructed DEM was taken for many norms, semi-norms, and the TSSIM index. These error estimates were averaged over the DEMs and quantitative results were produced. Below is a small sampling of the quantitative results. Please note, that in the last graph, contourlets with a 9/7

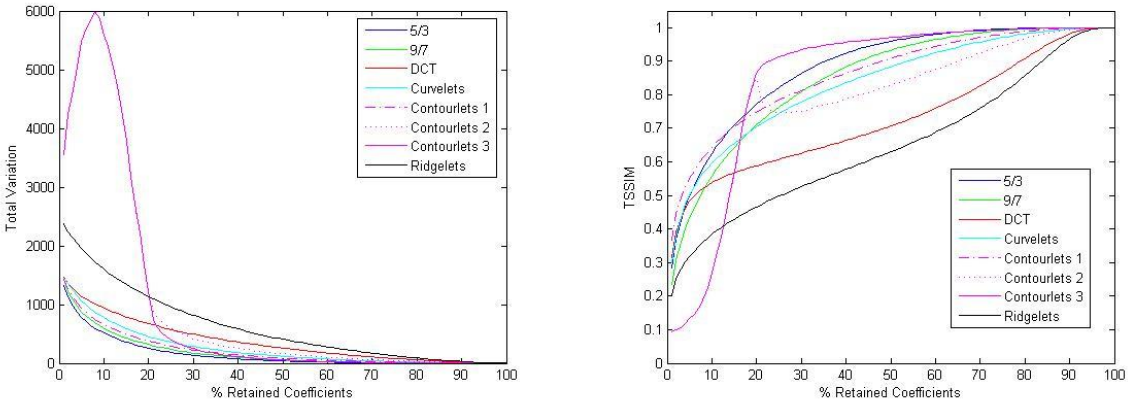
filter bank is giving the best TSSIM results. This is why we believe this method would give the best compression.



(a) (b)

Figure 8.13: The ℓ^2-norm and ℓ^∞-norm versus number of coefficients retained, (a) ℓ^2-norm (b) ℓ^∞-norm .

Also pleae note, if one looks closely at the TSSIM graph a small "knee" appears for one version of contourlets being run. This is not a mistake. This was actually quite an interesting result regarding TSSIM, contourlets with a high directional component, and urban data. Basically, the knee appeared with tiles that contained a large number of structures, e.g., buildings, houses, ... etc. This is a type of feature discrimination that we did not intend to find. When this transform is performed on images of building from airplanes and satellites, not DEMs, the knee is not present. This leads us to believe that image processing for DEMs may require some significant differences than image processing for images.



(a) (b)

Figure 8.14: The total variation semi-norm and the TSSIM index versus number of coefficients retained, (a) total variation (b) TSSIM.

8.7 Epilogue

After the above work was completed, we investigated adaptive techniques for compressing DEMs. Donoho's algorithm [23], wedgelets, is an adaptive scheme that partitions the DEM along edges and uses linear regression to find the optimal coefficient for each wedge of the image. The optimal representation is defined to be the minimizer of a functional that has both, ℓ^p norm and coefficient count. Results have shown that wedgelets gives better compression than other representations. However, the computational time of wedgelets is significantly higher than non-adaptive transformations.

Chapter 9

Motion deblurring

In August of 2006, I spent time at the University of Minnesota's Institute for Mathematics and its Applications. Below is a summary of the work that my team accomplished on image deblurring. This is joint work with Felix Krahmer[1] , Youzuo Lin[2] , Bonnie McAdoo[3] , Katharine Ott[4], Jiakou Wang[5]. Our mentor for the project was Brendt Wohlberg[6].

9.1 Introduction and Problem overview

Motion blur occurs when there is relative motion between the camera and the object being captured. In this section we study motion blur, that is, blur that occurs when the motion has constant speed and a fixed direction. The goal is to identify the angle and length of the blur. Once the angle and length of the blur are determined, a point spread function can be constructed. This point spread function is then used in direct deconvolution methods to help restore the degraded image.

[1]New York University

[2]Arizona State University

[3]Clemson University

[4]University of Virginia

[5]Pennsylvania State University

[6]Los Alamos National Laboratory

The process of blurring can be modeled as the following convolution

$$g(x,y) = f(x,y) * h(x,y) + n(x,y), \tag{9.1}$$

where $f(x,y)$ is the original image, $h(x,y)$ is the blurring point spread function, $n(x,y)$ is white noise and $g(x,y)$ is the degraded image. The point spread function for linear motion blur with a length of L and angle θ is given by

$$h(x,y) = \frac{1}{L}\delta(\overrightarrow{L}), \tag{9.2}$$

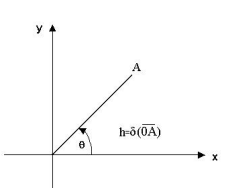
where $\overrightarrow{L}$ is the line segment of length L oriented at an angle of θ degrees from the x-axis.

Taking the Fourier transform of (9.1) we obtain

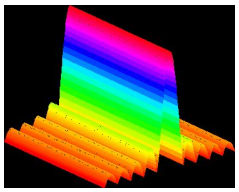
$$G(u,v) = F(u,v)H(u,v) + N(u,v). \tag{9.3}$$

The Fourier transform of the function $h(x,y)$, defined in (9.2), is a sinc function oriented in the direction of the blur.

$$g(x,y)=f(x,y)*h(x,y)+n(x,y) \quad \xrightarrow{\mathcal{F}} \quad G(u,v)=F(u,v)H(u,v)+N(u,v)$$



$\xrightarrow{\mathcal{F}}$



We multiply this sinc function by $F(u,v)$ in the frequency domain, so the ripples of the sinc function are preserved. We wish to identify the ripples in $G(u,v)$ to estimate the blur angle and blur length.

In this report, we describe various algorithms for determining point spread function parameters in the frequency domain. First we examine three methods for estimating blur angle, then two methods for estimating blur length. We compare the accuracy of the algorithms using artificially blurred images with different amounts of noise added. Finally, we use the estimations as parameters in MATLAB's deconvolution tools to deconvolve the images.

9.2 Three methods for angle estimation

9.2.1 The cepstral method

A method for identifying linear motion blur is to compute the two-dimensional *cepstrum* of the blurred image $g(x, y)$. The cepstrum of $g(x, y)$ is given by

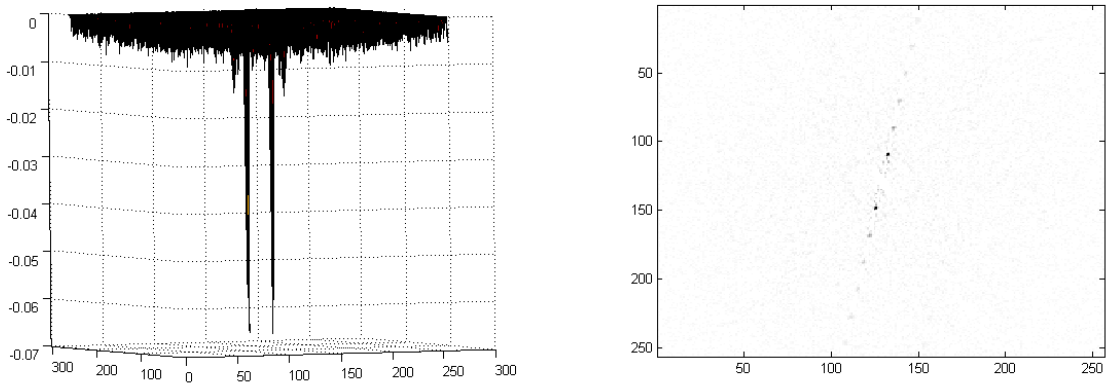
$$\mathcal{C}\left(g(x,y)\right) = \mathcal{F}^{-1}\left(\log\left|\mathcal{F}(g(x,y))\right|\right). \tag{9.4}$$

An important property of the cepstrum is that it is additive under convolution. Thus, ignoring noise, we have

$$\mathcal{C}\left(g(x,y)\right) = \mathcal{C}\left(f(x,y)\right) + \mathcal{C}\left(h(x,y)\right). \tag{9.5}$$

Biemond shows in [4] that $\mathcal{C}\left(h(x,y)\right) = \mathcal{F}^{-1}\left(\log\{|H(x,y)|\}\right)$ has large negative spikes at a distance L from the origin. By the additivity of the cepstrum, this negative peak is preserved in $\mathcal{C}\left(g(x,y)\right)$, also at a distance L from the origin.

If the noise level of the blurred image is not too high, there will be two pronounced peaks in the cepstrum, as show in Figure 9.1. To estimate the angle of



(a) (b)

Figure 9.1: The cepstrum of an image blurred at length 20 and $\theta = 80$. In (a) we see the two prominent negative peaks and in (b) the line through these two peaks appears to have an angle of 80 degrees.

motion blur, draw a straight line from the origin to the first negative peak. The angle of motion blur is approximated by the inverse tangent of the slope of this line.

9.2.2 Steerable filters method

Oriented filters are used to detect the edges in an image. A *steerable filter* is a filter that can be given an arbitrary orientation through a linear combination of a set of basis filters [24]. In this method, we apply a steerable filter to the power spectrum of the blurred image to detect the direction of motion.

The steerable filter we use is a second derivative of the Gaussian function. The radially symmetric Gaussian function in two dimensions is given by

$$G(x, y) = e^{-(x^2+y^2)}. \tag{9.6}$$

It can be used to smooth edges in an image by convolution. The second derivative of $G(x, y)$ will detect edges. By the properties of convolution,

$$d^2(G(x,y) * f(x,y)) = d^2\left(G(x,y)\right) * f(x,y). \tag{9.7}$$

We denote the second derivative of the Gaussian oriented at an angle θ by G_2^θ.

The basis filters for G_2^θ are

$$G_{2a} = 0.921(2x^2 - 1)e^{-(x^2+y^2)} \tag{9.8}$$

$$G_{2b} = 1.843xye^{-(x^2+y^2)} \tag{9.9}$$

$$G_{2c} = 0.921(2y^2 - 1)e^{-(x^2+y^2)}. \tag{9.10}$$

Then the response of the second derivative of the Gaussian at any angle θ, denoted RG_2^θ, is given by

$$RG_2^\theta = k_a(\theta)RG_{2a} + k_b(\theta)RG_{2b} + k_c(\theta)RG_{2c}, \tag{9.11}$$

where

$$k_a(\theta) = \cos^2(\theta) \tag{9.12}$$

$$k_b(\theta) = -2\cos(\theta)\sin(\theta) \tag{9.13}$$

$$k_c(\theta) = \sin^2(\theta). \tag{9.14}$$

To detect the angle of the blur, we look for the θ with the highest response value [39]. That is, we convolve RG_2^θ with the Fourier transform of our blurred image. For each θ, we calculate the L_2 norm of the matrix resulting from the convolution. The θ with the largest L_2 norm is the estimate for the angle of motion blur.

9.2.3 Radon transform method

Given a function $f(x, y)$, or more generally a measure, we define its Radon transform by

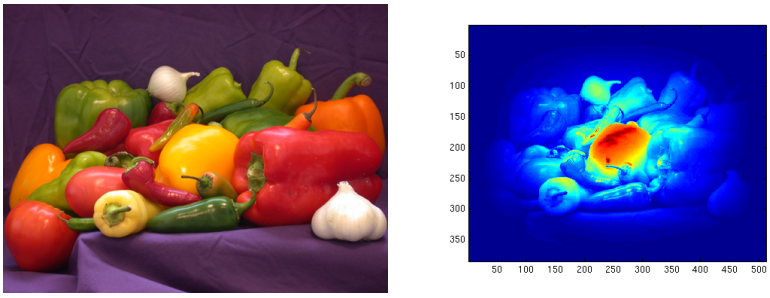
$$R(f)(x, \theta) = \int_{-\infty}^{\infty} f(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)dy. \tag{9.15}$$

This corresponds to integrating f over a line in $\mathbb{R}^2$ of distance x to the origin and at an angle θ to the y-axis.

To implement the Radon transform, we first assume that I is a square image. The content of I is assumed to be of finite support against a black background. Let $g(x, y)$ be the blurred image, and let θ be a vector of t equally spaced values from 0 to $180(1 - 1/t)$. For each $j = 1, \ldots, t$ compute the discrete Radon transform for $\theta(j)$. Call this matrix R. Now determine the angle $\theta(j)$ for which the Radon transform assumes its greatest values. Finally, we find the five largest entries in the j^{th} column of R, for each $j = 1, \ldots, t$, and sum them. The result is a vector v of length t, where each entry corresponds to an angle θ. The maximum entry of v provides the estimate for θ.

This method of angle detection has several shortcomings. Here, we offer three possible obstacles and present modifications to improve the versatility and robustness of the preceding algorithm.

1. If I is an m by n image where $m \neq n$, the axes in the frequency domain will have different lengths in the matrix representation. Calculating the angles in the frequency domain will thus lead to distortion. For example, the diagonal



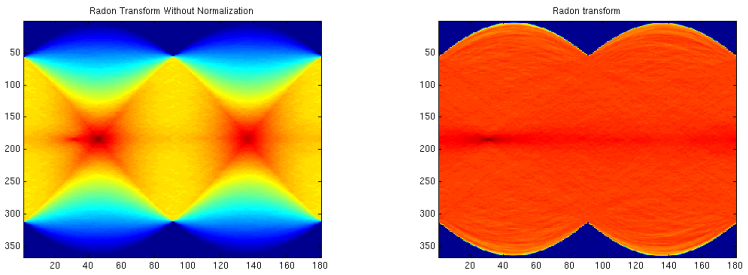
(a) (b)

Figure 9.2: The original image (a) and the image after windowing (b).

will not correspond to an angle of 45 degrees. To correct this, we let $\tilde{\theta} = \tan^{-1}(\frac{n}{m})\tan(\theta)$ and then run the algorithm replacing θ with $\tilde{\theta}$.

2. The preceding algorithm works for an image where the support of the content is finite, and the background is black. When the background is not black, or when there are objects close to the boundary of the image, the sharp edges will cause additional lines in the spectral domain at 0 degrees. The Radon transform will detect these edges. To avoid this effect, we smoothen out the boundaries of the image using a two dimensional Hann window. The values of this windowed image will decay towards the image boundary, as in Figure 9.2, so the edge effects disappear.

3. The Radon transform takes integrals along lines at different angles in a rectangular image. The length of the intersection between these lines and the image depends on the angle. The length is the longest at 45 degrees, so the integral will pick up the largest amount of noise contributions along this line. Thus the algorithm often incorrectly selects the angles of 45 and 135 as the angle



(a) (b)

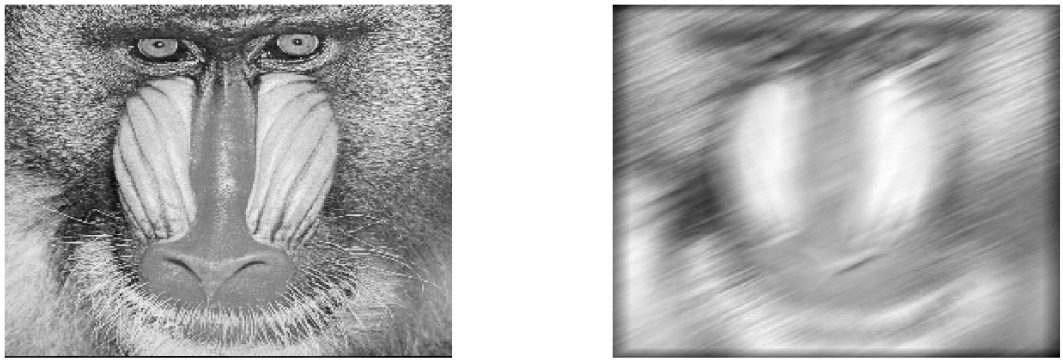
Figure 9.3: The Radon transform of the original image (a) and the normalized Radon transform (b).

estimate. To correct this, we normalize by dividing the image pointwise by the Radon transform of a matrix of 1's of the same dimension as the image.

9.2.4 Results

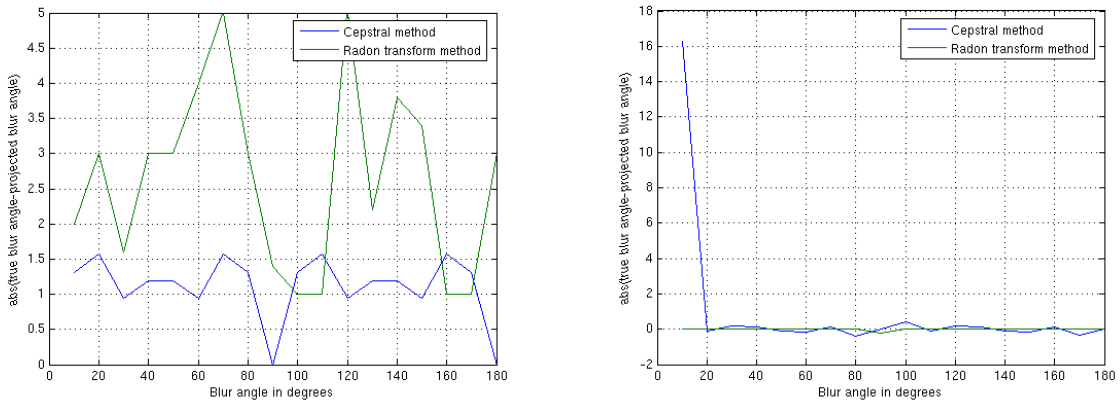
In this section we present results for angle estimation. The tests were run on the mandrill image seen in Figure 9.4. The image was blurred using the MATLAB motion blur tool with angles varying from 0 to 180. The results were recorded for images with both a low level of noise and a high level of noise added. The measurement for noise in an image is the signal-to-noise ratio, or SNR. The SNR measures the relative strength of the signal in a blurred and noisy image to the strength of the signal in a blurred image with no noise. An SNR of 30 dB is a low noise level, while an SNR of 10 dB is a high noise level.

The cepstral method is very accurate at all lengths when there is a low level of noise. The Radon transform also accurately predicts the blur angle, especially at longer lengths. The results are displayed in Figure 9.5. In the presence of noise the



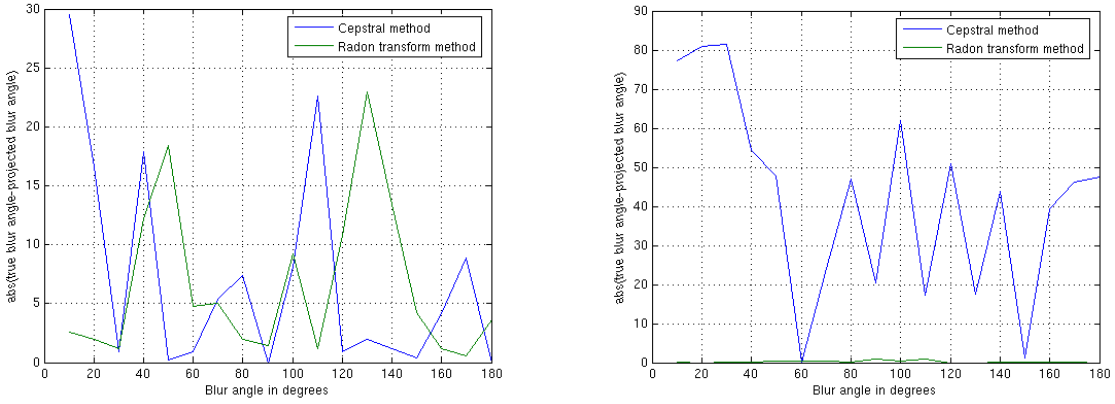
(a) (b)

Figure 9.4: The original mandrill image (a) and an example of a blurred image (b) with no noise. Here the length of the blur is 25 and $\theta = 30$.



(a) (b)

Figure 9.5: The average error in angle estimation for the cepstral and Radon transform methods with SNR = 30 dB. In (a) the length is 10 and in (b) the length is 50.



(a) (b)

Figure 9.6: The average error in angle estimation for the cepstral and Radon transform methods with SNR = 10 dB. In (a) the length is 10 and in (b) the length is 50.

cepstral method breaks down. At an SNR of 10 dB it performs poorly at all lengths. The Radon transform angle estimation, at this same noise level, is not accurate at small lengths but is very accurate at longer lengths, as depicted in Figure 9.6.

The steerable filters had a large amount of error in angle detection even with no noise. When the length was large, between roughly 40 and 70, the algorithm produces moderately accurate results, as in Figure 9.7.

A possible explanation why the Radon transform method fails for small blur lengths is that there is always a discretization necessary and the PSF looks less like a line segment. In fact, for a small length, the lines in the power spectrum of the PSF implemented in MATLAB are not very prominent. We attempted to solve these problems using an alternative approach for modeling the PSF following Choi [14], but this did not lead to better results.

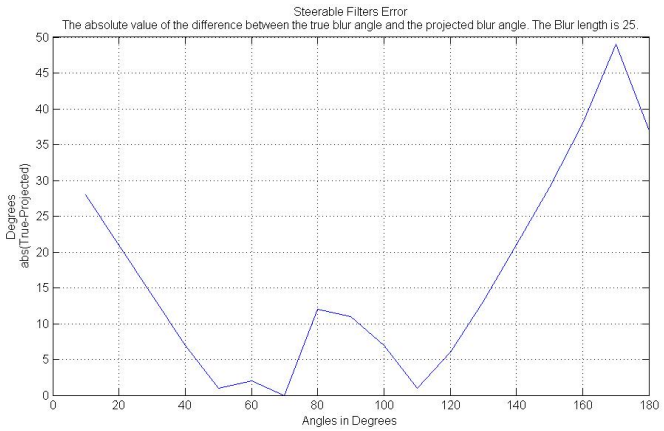


Figure 9.7: The average error in angle estimation for the steerable filter method with no noise.

9.3 Length Estimation

9.3.1 Two Dimensional Cepstral Method

As outlined in Section 9.2.1, the cepstrum of a blurred image shows two significant negative peaks at a distance L from the origin. An estimate for the length of motion blur is this value L. The cepstral method for angle detection is more susceptible to noise than the Radon transform method. Hence, we improve the result for the length detection by first estimating the angle via the Radon transform method in Section 9.2.3.

First, we de-noise the cepstrum of the noisy image using a Gaussian filter. Then we rotate the cepstrum by the angle θ estimated using the Radon transform. Assuming this angle estimate is reasonable, the peaks will lie close to the horizontal axis. Any peaks outside a small strip around the x-axis are caused by noise effects; we only need to look for the peaks inside the strip. Furthermore, the peaks should

appear at opposite positions from the origin. Thus we can amplify them by reflecting the image across the y-axis and then adding it. The result is that the peaks will add up and, in the rest, some noise will cancel.

Once we have made these corrections for noise, the estimated length of the motion blur is the distance between the negative peak and the y-axis, multiplied by an appropriate geometric correction factor.

9.3.2 One Dimensional Cepstral Method

The one dimensional cepstral method for length estimation uses the estimate of θ obtained from Section 9.2.1, 9.2.2 or 9.2.3. The idea is to collapse the log of the two dimensional power spectrum, $\log|\mathcal{F}(g(x,y))|$, onto a line that passes through the origin at an angle θ. Since the spectrum is collapsed orthogonal to the direction of motion, the resulting signal has the approximate shape of a sinc function [39]. Once the power spectrum is collapsed into one dimension, we take the inverse Fourier transform and then locate the first negative peak. We use the x coordinate of this peak to estimate the length.

Recall the definition of the *cepstrum* from (9.4). Note that in this method we essentially take the one dimensional cepstrum of the blurred image.

One algorithm to collapse the two dimensional power spectrum into one dimension is to calculate for each point (x,y) in the spectral domain the value

$$d = x\cos(\theta) + y\sin(\theta). \tag{9.16}$$

In the continuous case, the value $P(x,y)$ would then be projected onto the line

at a distance d from the origin. However, in the discrete case this value d is not necessarily an integer. Thus, we discretize by splitting the value of $P(x, y)$ onto two points, $\lfloor d \rfloor$ and $\lfloor d \rfloor + 1$, and weighting the distribution of $P(x, y)$ according the distance between d and $\lfloor d \rfloor$ as in [40].

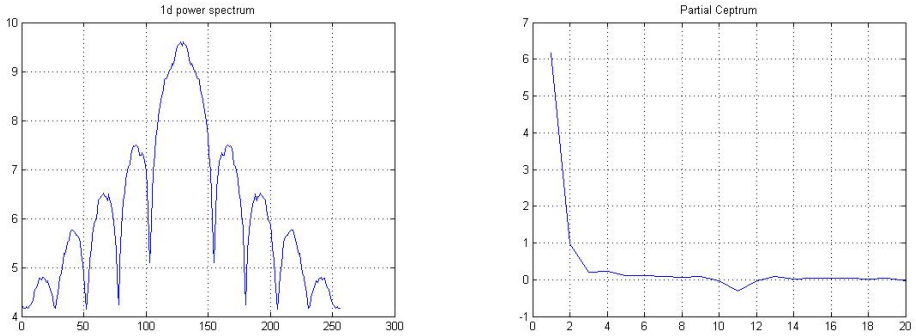
Another weighting method is to use the Radon transform. By taking the Radon transform of a constant matrix of 1's, we find the weights to assign to each point on the line passing through the origin at an angle θ. Take the Radon transform along the line that passes through the origin at an angle θ. This gives us the summation of all values $P(x, y)$ that contribute to each point on the line. Divide pointwise by these weights and now the power spectrum has been collapsed from two dimensions into one.

Following the preceding algorithm, one must compute a coordinate transformation correction factor. Let d_0 be the x-coordinate of the first negative peak in the 1D cepstrum, the length of which is denoted by D. The length d represents the estimated length in the image of size 256×256, and is given by

$$d = 256\frac{d_0}{D}. \tag{9.17}$$

9.3.3 Results

The two dimensional cepstral method for length estimation provides more accurate results than the one dimensional method. In Figure 9.10 we see that for no noise and low levels of noise, SNR of 20 dB and SNR of 30 dB, the two dimensional cepstral method averages less than a pixel in error for lengths between 10 and 60.



(a) (b)

Figure 9.8: Example of a collapsed 1D power spectrum (a) and the 1D cepstrum (b), with $\theta = 0$ and no noise. The actual blur length is 10; the estimated length of the blur by (b) is 11.

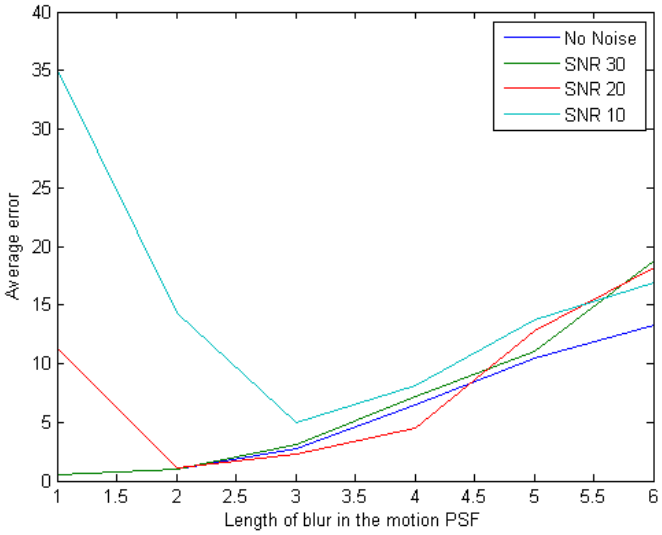
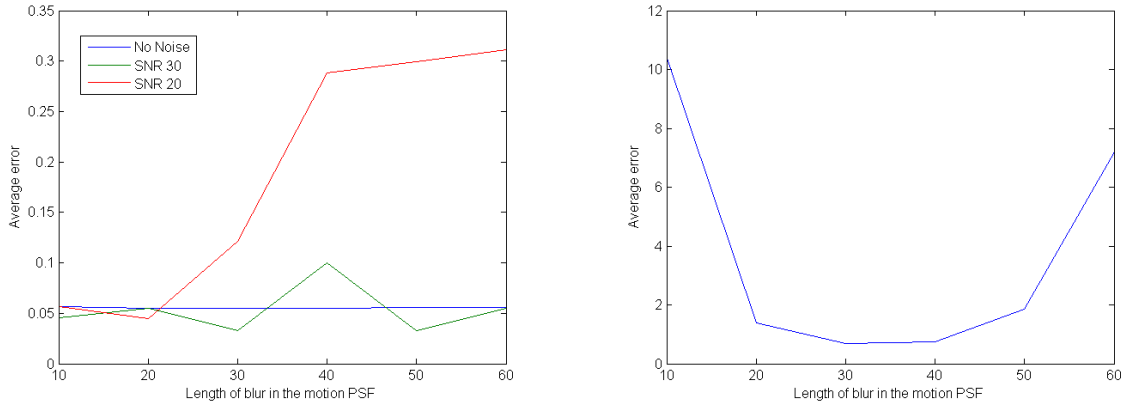


Figure 9.9: Error estimates in length estimation for 1D cepstral method.



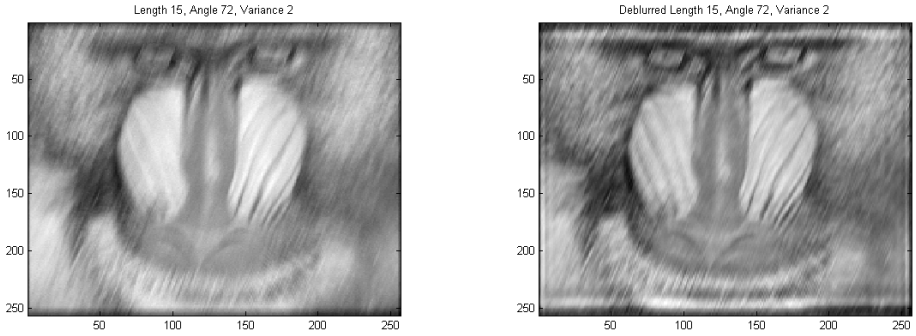
(a) (b)

Figure 9.10: Error estimates in length estimation for 2D cepstral method. In (a) we compare no noise, SNR = 30 dB and SNR = 20 dB. In (b) the error is much higher for an SNR = 10 dB.

At high noise levels, SNR of 10 dB, the method is relatively accurate for lengths between 20 and 50, but breaks down at small and large blur lengths.

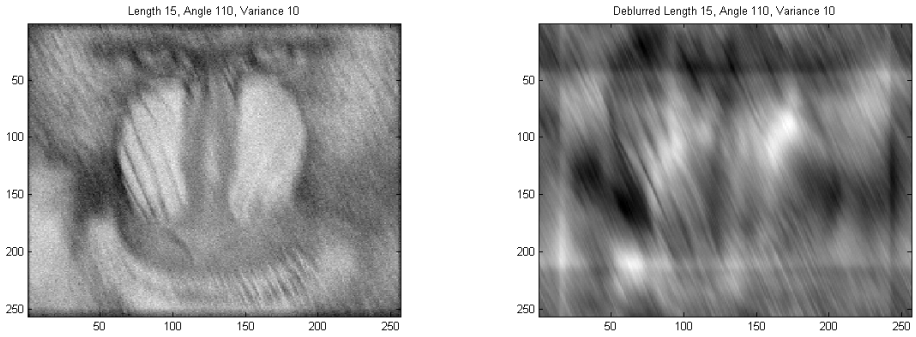
9.4 Deblurring with MATLAB's deconvolution method

In this section we implement restoration tests based on the orientation and length estimates computed in the preceding algorithms. We hope to minimize the effects of the restoration algorithms as much as possible in order to focus on the effects of our algorithms. The most frequently used image restoration algorithms are Wiener filters, Lucy-Richardson and regularized methods. Image restoration is a typical inverse problem and the well-posedness of the problem is critical. After imposing Gaussian noise in the test image, Wiener filters perform poorly because of the ill-conditioning of the problem. The regularized restoration algorithm is



(a) (b)

Figure 9.11: The blurred and noisy image (a) and the restored image (b).



(a) (b)

Figure 9.12: The blurred and noisy image (a) and the restored image (b).

designed to minimize the least squares error between the estimated and true images in the condition of preserving certain image smoothness, which is usually named as *Regularization Technique*. In other words, such a technique could change the ill-conditioned restoration problem to a well-posed problem and provide a reasonable solution. Hence, we decided to use this method to combine it with our estimates.

Our first test image, in Figure 9.11, is blurred and noised by a PSF with length 15, angle 72 degrees and an SNR of 24 dB. The restored images shows edges resulting from missing information (content was moved out of the picture). However,

the image quality has improved, image features are more distinctly visible. The second test image, in Figure 9.12, is blurred and noised by a PSF with length 15, angle 110 degrees and an SNR of 10 dB. In this example, the 2D cepstral method estimated a blur length of 90. Because the estimation for length is so inaccurate, the performance of the restoration algorithm is very poor.

However, this case was one of very few cases where the algorithm broke down for a short blur length. In most cases, the method led to noticeable improvements compared to the original image, even though the angle estimates given by the Radon transform were often slightly off. For a longer blur length, the results were not as satisfactory. However, the angle and length estimates were about right, so at least part of the problem is due to lacking accuracy of the deblurring method for longer blur lengths.

9.5 Conclusions

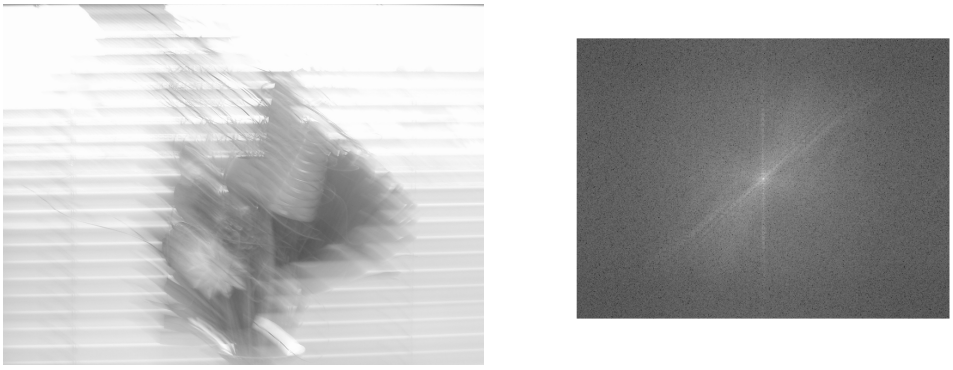
The deblurring results of Section 9.4 showed noticeable improvements in the image quality. Although the SNR of the deblurred image – even when restricted to the central part – still shows great differences from the original image, from the viewer's perspective the blur seems to have been at least partly removed. It is interesting to note that this holds true despite the error of a few degrees that occurs in the Radon transform method for a small blur length. Only when the length estimates go wrong due to increased noise contributions or for longer blur length, the deblurring gives worse results.

For a small blur length, the cepstral method performed better than the Radon transform method for estimating the angle. Further improvements might be achieved by using the Radon transform method to estimate the angle, the two dimensional cepstral method to estimate blur length, and then if the resulting blur length is small, refine the angle estimate using the cepstral method. In any case, starting with a coarse estimate and then refining close to that estimate can allow us to attempt higher precision levels without a great increase in computation time.

The image content may have an impact on the angle that is detected by the algorithms described. To combat this, using the algorithm on various blocks within the image is a possibility. However, performing initial experiments on a block decomposition rather than the whole matrix did not lead to better results. Better improvements might occur for higher noise levels and bigger images.

Another possibility to improve the result is to actually use the functional that is minimized in the regularized deconvolution method to judge the quality of the estimated point spread function. Determine several candidates for the PSF and then compare the minimum values for the associated functionals. If the estimations are far off, the functional will not assume values as small as for the actual PSF.

After optimizing the blur identification using these techniques, the next step will be to try the method on real world images. Looking at the spectral image of a blurred picture in Figure 9.13, one can see faint lines in the blur direction. However, it is not a priori clear if these lines will be picked up as well as for artificially blurred images.



(a) (b)

Figure 9.13: The real blurred image (a) shows faint lines in its power spectrum (b).

Bibliography

[1] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 13:1373–1396, 2003.

[3] J.J. Benedetto and M. Fickus. Finite normalized tight frames. *Adv. Comput. Math.*, 18:357–385, 2003.

[4] J. Biemond. Iterative methods for image deblurring. *Proceedings of the IEEE*, 78(5):856–883, 1990.

[5] J.C. Bremer, R. Coifman, M. Maggioni, and A.D. Szlam. Biorthogonal diffusion wavelets for multiscale representations on manifolds and graphs. *Wavelets XI*, 5914(1):59141, 2005.

[6] E. J. Candes. *Ridgelets: theory and applications*. PhD thesis, Department of Statistics, Stanford University, 1998.

[7] E. J. Candes and D. L. Donoho. Ridgelets: a key to higher-dimensional intermittency? *Phil. Trans. R. Soc. Lond. A.*, pages 2495–2509, 1999.

[8] E. J. Candes and D. L. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise c2 singularities. *Communications on Pure and Applied Mathematics*, LVII, 2004.

[9] E. J. Candes and D. L. Donoho. Continuous curvelet transform. ii. discretization and frames. *Appl. Comput. Harmon. Anal*, 19(2):198–222, 2005.

[10] E. J. Candes and D.L. Donoho. Continuous curvelet transform: I. resolution of the wavefront set. Technical report, 2003.

[11] E.J. Candes and D.L. Donoho. Curvelets, multiresolution representation, and scaling laws.

[12] E.J. Candes and D.L. Donoho. Curvelets: A surprisingly effective nonadaptive representation of objects with edges. Technical report, 1999.

[13] M.M. Chang, A. M. Tekalp, and A.T. Erdem. Blur identification using the bispectrum. *IEEE transactions on signal processing*, 39(3):2323–2325.

[14] J.W. Choi, M.G. Kang, and K.T. Park. An algorithm to extract camera-shaking degree and noise variance in the peak-trace domain. *IEEE transactions on consumer electronics*, 44(3):1159–1168, 1998.

[15] F. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92) (Cbms Regional Conference Series in Mathematics)*. American Mathematical Society, 1997.

[16] 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. part i: Diffusion maps. *Proc. of Nat. Acad. Sci.*, (102):7426–7431, 2005.

[17] R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[18] M. Do and M. Vetterli. Beyond wavelets. J. Stoeckler and G. V. Welland (Eds.).

[19] M. Do and M. Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Trans. on Image Processing*.

[20] M. Do and M. Vetterli. Pyramidal directional filter banks and curvelets. pages 158–161.

[21] M. Do and M. Vetterli. The finite ridgelet transform for image representation. *IEEE Trans. Image Processing*, (1):16–28, 2003.

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

[23] D.L. Donoho. Wedgelets: Nearly minimax estimations of edges. 27(3):859–897, 1999.

[24] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE transactions on pattern analysis and machine intelligence*, 13(9):891–906, September 1991.

[25] J. Ham, D.D. Lee, S. Mika, and B. Scholkopf. A kernel view of the dimensionality reduction of manifolds. *Max Planck Instituet for Biological Cybernetics*, (Technical Report-110), 2003.

[26] M. Hirn. private communication, 2007.

[27] H. Kaufman and A.M. Tekalp. Survey of estimation techniques in image restoration. *IEEE control systems magazine*, 11(1):16–24, January 1991.

[28] J.P. Kerekes and J.R. Schott. Hyperspectral data exploitation, theory and applications. chapter Chapter 1: Hyperspectral Imaging Systems.

[29] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. *In Proc. 19th Intl. Conf. on Machine Learning*, 2002.

[30] I. Konstantinidis. private communication, 2007.

[31] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE signal processing magazine*, 13(3):43–64, 1996.

[32] R. Lokhande, K.V. Arya, and P. Gupta. Identification of parameters and restoration of motion blurred images. pages 301–315, April 2006.

[33] S. Mallat. *A wavelet tour of signal processing*. Academic Press, 1998.

[34] C. Mayntz and T. Aach. Blur identification using a spectral inertia tensor and spectral zeros. *IEEE image processing*, 2:885–889, 1999.

[35] M.E. Moghaddam and M. Jamzad. Motion blur identification in noisy images using fuzzy sets. pages 862–866.

[36] P. Parrilo, S. Lall, F. Paganini, G. Verghese, B. Lesieutre, and J. Marsden. Model reduction for analysis of cascading failures in power systems. *Proceedings of the American Control Conference*, 6:4208–4212, 1999.

[37] V.P. Pauca, J. Piper, and R.J. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and its Applications*, 416(550):29–47, 2006.

[38] S. J. Reeves. Optimal space-varying regularization in iterative image restoration. *IEEE transactions on image processing*, 3(3):319–324, 1994.

[39] I. M. Rekleitis. Steerable filters and cepstral analysis for optical flow calculation from a single blurred image. In *Vision Interface*, pages 159–166, Toronto, May 1996.

[40] I.M. Rekleitis. Visual motion estimation based on motion blur interpretation. Master's thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, 1995.

[41] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290(550):2323–2326, 2000.

[42] B. Scholkopf, A. Smola, and K. uller. Nonlinear component analysis as a kernel eigenvalue problem, 1998.

[43] J. Shlens. A tutorial on principal component analysis.

[44] J. L. Starck. Deconvolution in astronomy: A review. *Publications of the Astronomical Society of the Pacific*, 114(800):1051–1069, 2002.

[45] T. G. Stockham, T. M. Cannon, and R. B. Ingebretsen. Blind deconvolution through digital signal processing. *Proceedings of the IEEE*, 63(4):678–692, 1975.

[46] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. 2007.

[47] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity, 2004.

[48] Z. Wang, A. C. Bovik, and E. P. Simoncelli. *Handbook of Image and Video Processing, Structural approaches to image quality assessment.* Alan Bovik, Academic Press, 2005.

[49] Z. Wang and E. P. Simoncelli. Translation insensitive image similarity in the complex wavelet domain. IEEE Int'l Conf Acoustics Speech, Signal Processing, 2005.

[50] Z. Wang, E. P. Simoncelli, and A. C. Bovik. similarity for image quality assessment. I37th Asilomar Conf. on Signals, Systems and Computers, 2003.

[51] A.L. Warrick and P.A. Delaney. Detection of linear features using a localized radon transform. *IEEE*, pages 1245–1249.

[52] Y. Yitzhaky, G. Boshusha, Y. Levy, and N. S. Kopeika. Restoration of an image degraded by vibrations using only a single frame. *Optical engineering*, 39(8):2083–2091, 2000.

[53] Y. Yitzhaky and N. S. Kopeika. Identification of the blur extent from motion-blurred images. *Proceedings of SPIE–the international society for optical engineering*, 2470:2–11, 1995.

[54] Y. Yitzhaky and N. S. Kopeika. Restoration of motion blurred images. *Proceedings of SPIE–the international society for optical engineering*, 3164:27 – 37, 1997.

[55] Y. Yitzhaky, I. Mor, A. Lantzman, and N. S. Kopeika. Direct method for restoration of motion-blurred images. *Journal of the Optical Society of America. A, Optics and image science*, 15(6):1512–1519, 1998.