IMAGE PROCESSING TECHNIQUES FOR EXTRACTION OF
WIND FIELDS FROM LIDAR AEROSOL BACKSCATTER

A Thesis in
Electrical Engineering

by
Brady R. Bickel

© 2012 Brady R. Bickel

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2012
The thesis of Brady R. Bickel was reviewed and approved* by the following:

Timothy J. Kane  
Professor of Electrical Engineering  
Thesis Adviser

Richard L. Tutwiler  
Professor of Electrical Engineering

Kultegin Aydin  
Professor of Electrical Engineering  
Interim Head of the Department of Electrical Engineering

*Signatures are on file in the Graduate School.
Abstract

The tracking of winds and atmospheric features has many applications, from predicting and analyzing weather patterns in the upper and lower atmosphere to monitoring air movement from pig and chicken farms. Doppler lidar systems exist to quantify the underlying wind speeds, but cost of these systems can sometimes be relatively high, and processing limitations exist in these systems. There is also interest in deployment of these systems where Doppler lidar cannot be easily used. Currently, image correlation techniques are used to calculate wind vectors with lidar aerosol backscatter. By using data from a simple aerosol lidar system, we attempt to extend these processing methods to improve the resulting correlations. By preconditioning the aerosol backscatter imagery, wind calculation uncertainty can be reduced, leading to potential benefits such as reduction of noise sensitivity, and shorter lidar scan times. Image thresholding and bilateral filtering are applied to the backscatter imagery, and it is shown how these techniques help to improve the correlation quality. Also, use of an image registration method for wind calculation is tested and results are compared to the correlation techniques. Use of these preprocessing and registration methods show promise in the improvement of the wind detection problem using inexpensive aerosol lidar setups with sophisticated processing algorithms.
## Table of Contents

List of Tables .......................................................... vi
List of Figures .......................................................... vii
Acknowledgments ........................................................ ix

Chapter 1. Introduction ............................................... 1
  1.1 Background and Motivation ....................................... 2
    1.1.1 Digital Particle Image Velocimetry .......................... 5
    1.1.2 Gridded Cross-Correlation ................................. 6
  1.2 REAL Lidar ....................................................... 6

Chapter 2. Data Format and Conditioning ............................ 9
  2.1 Running Median High Pass Filter ............................... 10
  2.2 Cartesian Mapping with Modified Bilinear Interpolation ..... 12
  2.3 Temporal-Median Image Creation ............................... 17
  2.4 Histogram Normalization ....................................... 18

Chapter 3. Preprocessing Methods for Image Correlation Wind Detection ............................................... 19
  3.1 Base Image Correlation ........................................ 20
  3.2 Image Correlation with Preprocessing ......................... 21
    3.2.1 Hard Limiting Feature Extraction ....................... 22
    3.2.2 Bilateral Filtering Feature Smoothing ................... 22

Chapter 4. Image Registration ....................................... 27

Chapter 5. Results .................................................. 30
  5.1 Correlation Methods ........................................... 31
    5.1.1 Raw Correlation ........................................... 31
      5.1.1.1 Example 1 ........................................ 31
      5.1.1.2 Example 2 ........................................ 35
      5.1.1.3 Example 3 ........................................ 38
    5.1.2 Hard Limited Correlation ................................ 41
      5.1.2.1 Example 1 ........................................ 41
      5.1.2.2 Example 2 ........................................ 45
      5.1.2.3 Example 3 ........................................ 48
    5.1.3 Bilateral Filtered Correlation ............................ 51
      5.1.3.1 Example 1 ........................................ 51
      5.1.3.2 Example 2 ........................................ 55
      5.1.3.3 Example 3 ........................................ 58
  5.2 Registration Method ........................................... 61
List of Tables

5.1 Peak Variances ................................................. 65
5.2 Correlation/Registration Wind Calculation Comparison in Pixels .... 66
5.3 Correlation/Registration Wind Calculation Comparison in Meters per Second .................................................. 66
5.4 Algorithm Execution Times ..................................... 66
# List of Figures

1.1 Horizontal Volume Scanning Aerosol Backscatter Lidar ............... 2  
1.2 Vertical Volume Scanning Aerosol Backscatter Lidar .................. 3  
2.1 REAL Example of Unfiltered Backscatter Scan ...................... 10  
2.2 REAL Example of Filtered Backscatter Scan ............................. 11  
2.3 Bilinear Interpolation ........................................... 13  
2.4 Modified Bilinear Interpolation ..................................... 16  
3.1 Correlation Frame 1 Subset ......................................... 20  
3.2 Correlation Frame 2 Subset ........................................... 21  
3.3 Correlation Frame 1 Subset Hard Limited ............................. 23  
3.4 Correlation Frame 2 Subset Hard Limited ............................. 24  
3.5 Correlation Frame 1 Subset Bilateral Filtered ....................... 25  
3.6 Correlation Frame 2 Subset Bilateral Filtered ....................... 26  
4.1 Registration Frame 1 ................................................. 28  
4.2 Registration Frame 2 ................................................. 28  
5.1 Example 1: Raw Input 1 .............................................. 32  
5.2 Example 1: Raw Input 2 .............................................. 33  
5.3 Example 1: Raw Correlation .......................................... 34  
5.4 Example 2: Raw Input 1 .............................................. 35  
5.5 Example 2: Raw Input 2 .............................................. 36  
5.6 Example 2: Raw Correlation .......................................... 37  
5.7 Example 3: Raw Input 1 .............................................. 38  
5.8 Example 3: Raw Input 2 .............................................. 39  
5.9 Example 3: Raw Correlation .......................................... 40  
5.10 Example 1: Hard Limited Input 1 .................................... 42  
5.11 Example 1: Hard Limited Input 2 .................................... 43  
5.12 Example 1: Hard Limited Correlation ................................ 44  
5.13 Example 2: Hard Limited Input 1 .................................... 45  
5.14 Example 2: Hard Limited Input 2 .................................... 46  
5.15 Example 2: Hard Limited Correlation ................................ 47  
5.16 Example 3: Hard Limited Input 1 .................................... 48  
5.17 Example 3: Hard Limited Input 2 .................................... 49  
5.18 Example 3: Hard Limited Correlation ................................ 50  
5.19 Example 1: Bilateral Filtered Input 1 ............................... 52  
5.20 Example 1: Bilateral Filtered Input 2 ............................... 53  
5.21 Example 1: Bilateral Filtered Correlation ........................... 54  
5.22 Example 2: Bilateral Filtered Input 1 ............................... 55  
5.23 Example 2: Bilateral Filtered Input 2 ............................... 56  
5.24 Example 2: Bilateral Filtered Correlation ........................... 57
<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.25</td>
<td>Example 3: Bilateral Filtered Input 1</td>
<td>58</td>
</tr>
<tr>
<td>5.26</td>
<td>Example 3: Bilateral Filtered Input 2</td>
<td>59</td>
</tr>
<tr>
<td>5.27</td>
<td>Example 3: Bilateral Filtered Correlation</td>
<td>60</td>
</tr>
<tr>
<td>5.28</td>
<td>Example 1: Registration Frame 1</td>
<td>61</td>
</tr>
<tr>
<td>5.29</td>
<td>Example 1: Registration Frame 2</td>
<td>62</td>
</tr>
<tr>
<td>5.30</td>
<td>Example 2: Registration Frame 1</td>
<td>62</td>
</tr>
<tr>
<td>5.31</td>
<td>Example 2: Registration Frame 2</td>
<td>63</td>
</tr>
<tr>
<td>5.32</td>
<td>Example 3: Registration Frame 1</td>
<td>63</td>
</tr>
<tr>
<td>5.33</td>
<td>Example 3: Registration Frame 2</td>
<td>64</td>
</tr>
</tbody>
</table>
Acknowledgments

Many thanks to Dr. Shane Mayor of California State University, Chico, for graciously providing data and background material for use in this thesis. The results described are very much dependent on his generosity and spirit of sharing.

Thanks to my advisor, Dr. Tim Kane, for his patience and guidance through the process. Also thanks to my committee member, Dr. Tutwiler, for providing valuable input and a unique perspective to the problem. Finally, much gratitude is owed to Eric Rotthoff, with whom I have had many discussions about the topic at hand.

Most of all, thanks goes especially to my wife, Ann, who gave up much while this work was completed. This would not be possible without her.
Chapter 1

Introduction

Using aerosol backscatter lidar returns to detect wind fields is a problem that has numerous applications. There exist Doppler lidar systems that measure the frequency shift of light scattered by aerosol particles. These systems are expensive, and require high precision setups. Therefore, attempts to extract wind features from non-Doppler lidar scans have been done in the past. Doppler lidars also have a trade-off between velocity accuracy and range resolution. These non-Doppler techniques are of interest here, and the methods described herein do not experience any of these limitations. We expand on these methods in this thesis. The instrument we focus on is a simple volume scanning aerosol backscatter lidar, which produces backscatter as shown in Figures 1.1 and 1.2[1]. The images used for this processing are not captured instantaneously, that is, there is a finite time to capture each backscatter profile. Therefore, the resulting images do not perfectly represent an instantaneous state of the aerosol particles. In the examples used, typical capture time is between 10 and 30 seconds.

This chapter provides an overview of the current state of single channel lidar wind detection. This introduces the need to improve upon these methods, which is the goal of this thesis. Section 1.1 includes motivation for improving the wind detection algorithms
Fig. 1.1. Horizontal Volume Scanning Aerosol Backscatter Lidar. Range rings are 500 meter increments.

presented herein, as well as background of the current lidar detection paradigm. Raman-Shifted Eye-Safe Aerosol Lidar (REAL) system, its relevant technical specifications and its current uses are described in Section 1.2.

1.1 Background and Motivation

Lidar scans are used to detect small aerosol particles, which cannot be detected by radar, due to the smaller wavelength of light. Underlying wind velocities can be
Fig. 1.2. Vertical Volume Scanning Aerosol Backscatter Lidar. Range rings are 1 kilometer increments.

calculated through the detection of these aerosols. Calculation of these winds have many applications, ranging from plume detection in agricultural applications as in [2], to classification and characterization of atmospheric features. As described in [3] and [4], lidar is used for detection and measurement of atmospheric parameters, such as density, temperature, humidity, aerosols, and wind velocity. Lidar has also been useful for measuring climatological change indicator particles, such as ozone, nitrogen dioxide, and carbon monoxide. Another application is the measurement of turbulent eddies, as described in [5].

Many meteorological applications of Doppler lidar exist, such as characterizing sea breeze initiation and evolution, investigation of the structure of severe downslope windstorms, and observation of turbulence parameters in the lower troposphere [6]. Lidar
systems can be ground based, but can also be airborne or satellite based. These non-ground lidar platforms, however, are often used for applications such as terrain mapping.

Due to the limitations of the Doppler lidar systems, there is a research interest in providing low-cost, high quality solutions that deliver similar capabilities using high resolution lidar returns. Simple, non-Doppler lidar systems are relatively inexpensive; however, the problem space is dramatically different from a Doppler lidar system.

Therefore, due to high cost, it is worth investigating the possibility of doing simple aerosol lidar wind detection. In fact, this type of detection has been attempted, as will be discussed. We investigate the current algorithms to detect and identify wind direction and speed. We also discuss the limitations of the algorithms, and how they can be improved upon.

While the Doppler lidar problem is a remote sensing problem, the problem as stated in a single channel lidar system is primarily an image processing problem. Without highly calibrated Doppler lidar systems, algorithms must be developed that are able to recognize a variety of features, extract relevant measurements from the features, and correlate and track these features across time. In [7], Mayor has performed cross-correlations on images derived from the raw lidar profiles to track aerosols from one image frame to the next. Mayor and Eloranta have shown also that the measurement of lidar scan velocities correspond well with anemometer measurements.

The goal of this thesis is to extend these ideas to enhance the detection concept presented in [7]. Sections 1.1.1 and 1.1.2 describe the state of the art wind detection methods using lidar. Eventually, it would also be desirable to provide a real-time, unsupervised processing wind detection and characterization capability.
1.1.1 Digital Particle Image Velocimetry

One possible approach to the feature tracking problem is to use Digital Particle Image Velocimetry (DPIV) [8]. This is a flow visualization technique that has applications in fluid flow systems. This technique correlates sampled digital images and correlates the points across images. The different images are taken at two separate times, at a capture rate that is an appropriate length to detect motion in the image. An optimal sampling period between images is as small as can be allowed by the physical lidar hardware, restricted only by the averaging required to receive a strong enough backscatter to perform the feature identification. The operation between the two images is a two-dimensional discrete cross-correlation function:

\[
C(i', j') = \sum_{i=-k}^{k} \sum_{j=-l}^{l} f(i, j)g(i + i', j + j')
\]  

(1.1)

where \(f\) and \(g\) are two consecutive lidar scans, converted into two dimensional Cartesian images.

A weakness of the DPIV idea is that it attempts to track all particles (pixels, in this case) in a given image. While in theory this sounds useful, in certain applications, the image sequence is such that the background noise is strong, varies in a chaotic manner, or is simply not of interest. Another possible downfall is that high sampling periods may not sufficiently capture the movement of the particles, due to deformations and turbulence between the sampling instances. This is also not ideal to do for an entire image if there is a non-uniform wind over the entire geographic area, which is often the case for the scale of the lidar scans.
1.1.2 Gridded Cross-Correlation

In [7], Mayor and Eloranta describe a variation of DPIV as explained in Section 1.1.1, but processes the correlations on a series of subsets of the images. The larger images are gridded into smaller sub-images, and cross correlation is performed in each of these regions. Then, the correlation peak corresponds to the two-dimensional shift that occurred in the sub-region. The overall wind vector flow can be characterized by looking at the overall trends of wind shifts across the entire image. Choosing the size of the grid is an important part of computing a quality correlation. If too small a window is chosen, it is possible that the features of interest shift out of the window between scans. If too large a window is chosen, variations in the wind features across space could be lost due to a noisy correlation peak. Therefore, selection of the window size can be sized based on an estimate of reasonable wind speed, or estimates of the winds from prior frames. Feature deformation such as vorticity and shear can not be detected if the window is too large. Because of the nature of lidar scans, there is often a relatively low sample rate (e.g. tens of seconds to sample a single image). Because of this low rate, in high wind speed scenarios, it becomes more difficult to track the winds to a higher degree of accuracy. Image window sizes must increase, and inversely, correlation accuracy and resolution of small deformations decrease.

1.2 REAL Lidar

The Raman-shifted Eyesafe Aerosol Lidar (REAL) system is an aerosol backscatter lidar system operated by Mayor at California State University, Chico. The data used
herein was obtained from this source. In [9], Dr. Mayor describes some of the technical details of REAL that are relevant:

- operates at a wavelength of 1.54 microns
- eyesafe system, within region with highest maximum eye-safe energy (MEE)
- transmits with 170 mJ of energy per pulse
- 40 cm diameter optics
- reads two channels of data

While REAL does read two channels of radial data, the second detection channel is in place to add backscatter depolarization sensitivity to REAL[9]. For this application, we are only interested in one single channel of data, which is sufficient in creating the images and optical flow necessary to analyze the data. Therefore, the second channel of data is ignored, but could be considered for future applications.

The data collected from REAL that is used in the following analysis was part of a program funded by the National Science Foundation (NSF) called Canopy Horizontal Array Turbulence Study (CHATS) [10]. The goal of this program is to compare the results of the cross correlation algorithm to anemometer measurements. The goal of this thesis is not to evaluate the relationship or accuracy of the lidar measurement with respect to anemometer measurements, but rather to improve and expand upon the correlations techniques already in use.

Chapter 2 explains the processing required to convert the raw lidar data into usable images for further processing. Chapter 3 describes the preprocessing enhancements
to the images for correlation. Chapter 4 introduces a new application of an image processing tracking and registration algorithm to detect the lidar optical flow. Chapter 5 offer detailed experimental results, and Chapter 6 summarizes the results and provides conclusions as well as future research topics applicable to this problem.
Chapter 2

Data Format and Conditioning

First, the coordinate spaces needed to be translated, to work with the lidar scans as images. Appendix A details the REAL data format and what steps were taken to read and verify the input data. The steps to achieve this translation are as follows [7]:

1. Perform running median high pass filter

2. Create Cartesian image with linear interpolation

3. Subtract temporal-median image from image sequence

4. Histogram normalization

Typically, there is a need for a range-squared correction to be applied to the radial data; however, in this case, the instrument applied the correction, so it would be redundant to range-squared correct the data. The reason a range-squared correction is needed is to compensate for the reduction of return amplitudes due to electromagnetic propagation as range increases.

The goal of performing all of these operations is to provide an image sequence that highlights the non-background high spatial frequency (strong edge) features. The median filter enhances these features, while the background subtraction removes stationary features that contaminate the cross correlation. Interpolation is important to
Fig. 2.1. REAL Example of Unfiltered Backscatter Scan

ensure a quality conversion between the two coordinate systems. For reference, the difference between filtered and unfiltered images can be shown in Figures 2.1 and 2.2, which have been created with Dr. Mayor’s data visualization tools[10]. These images were not directly used in this research; they are provided for illustration purposes, only.

2.1 Running Median High Pass Filter

The running median high pass filter is a nonlinear filter used to remove statistical outliers within the filter window. The filter preserves high spatial frequency edge components in the image. This filtering is performed on the raw lidar data, in the radial direction. That is, for each lidar profile, an independent data filter operation is performed.
The filter operation requires a filter length parameter, specified in meters. For the processing described, a half filter length of 150 points was used. This value determines how many points are to be included in the single dimensional filter. The filter operation differs from a traditional median filter, in that the median value of the points in the filter window at any given time is subtracted from the center point in the window. A typical median filter replaces the center point with the median value. This subtle difference creates a filter output that is excellent at highlighting the outliers and effective removal of background drift [11].

For half-filter length \( k \) and number of input data samples \( N \):

\[
x(n) = x(n) - \text{median}(x(n - k), \ldots, x(n + k)) \quad \text{for} \quad n = k, \ldots, N - k
\]  

(2.1)
This nonlinear process creates an image that has higher intensity transient features.

2.2 Cartesian Mapping with Modified Bilinear Interpolation

Once the data has been median filtered, the radial lidar data profiles must be mapped into a two-dimensional Cartesian plane. Since the REAL scans occur on a single plane, the resulting image exists on the plane of the original lidar scan. The analysis performed here is on the data where the elevation is fixed. These horizontal scans tend to have more visually interesting features, as opposed to vertical, constant azimuth scans. These interesting features are essential to test the types of image processing algorithms examined here.

Proper interpolation is an important part of processing the raw radial data into a quantized two-dimensional image grid. Interpolation techniques vary from the very simple to implement to the more complicated. As usual, there is a tradeoff between algorithm simplicity and performance. As expected, the simpler algorithms suffer from poor performance, whereas the interpolation methods that are more robust produce smoother, higher fidelity images, particularly at the high spatial frequency edges. For example, nearest neighbor interpolation is the simplest to implement, but provides the least smooth image, and uses the information in the surrounding points least effectively.

Bilinear interpolation is a common method of image interpolation. Though the name suggests that the interpolation is linear, it is actually a type of quadratic interpolation. The quadratic operation results from considering two orthogonal linear interpolation operations jointly. The likelihood is low that any given point on the grid is at
The standard bilinear interpolation concept is shown here. For rectangular grid interpolation, each interpolated point is a function of the four surrounding points. The exact location as a sample in the lidar profile. Therefore, some kind of interpolation needs to be done.

The bilinear image interpolation technique is described and illustrated in [12]. In Figure 2.3, an arbitrary point \((x_i + x_f, y_i + y_f)\) where \(x_i, y_i\) are the integer components of the point to be interpolated and \(x_f, y_f\) are the fractional components. The interpolated values are computed using the following:

Assuming \(x = x_i + x_f, y = y_i + y_f,\) and \(I \left( \begin{bmatrix} x \\ y \end{bmatrix} \right)\) is the image intensity at point \((x, y)\):}

\[
I \left( \begin{bmatrix} x' \\ y' \end{bmatrix} \right) = (1 - x_f)(1 - y_f)A + x_f(1 - y_f)B + y_f(1 - x_f)C + x_f y_f D \tag{2.2}
\]

where
However, the bilinear interpolation equation in (2.2) does not readily apply to the problem at hand. The raw data captured by any lidar system is captured as a series of backscatter points at a given azimuth/elevation combination at any given time. The azimuth and elevation are varied to image the desired volume scan. Data read this way creates a polar plot image, where each line radiates from the source system. Bilinear interpolation as described above is for a Cartesian style image.

As a result, the bilinear interpolation needs to be modified to deal with this situation. The first step of the interpolation remains unchanged. The positions and intensity values of the points of the image are dictated by two things: the image range and the image resolution. If the entire lidar scan is included in the image, the $x, y$
coordinates are determined by the following:

\[ M = \left\lceil \frac{|y_{max} - y_{min}|}{y_{res}} \right\rceil \] \hspace{1cm} (2.7)

\[ N = \left\lceil \frac{|x_{max} - x_{min}|}{x_{res}} \right\rceil \] \hspace{1cm} (2.8)

where \( y_{max}, y_{min}, x_{max}, \) and \( x_{min} \) are the maximum and minimum values on the plane of the scan, and \( y_{res} \) and \( x_{res} \) are the resolutions in each respective direction, which are parameters for image creation that drive the final size of the image. Then, \( M \times N \) defines the image size in the Cartesian coordinate system.

Then, to convert the lidar scan from polar to Cartesian coordinates, each point on the grid is converted one at a time. To calculate the interpolation, each point in the target grid image is considered in turn. For the Cartesian conversion, the lidar system receiver position on the image can be arbitrarily selected, provided the lidar points fit on the final grid. Then, the distance from the source to the point in question is computed. The angle to the source is also computed. When the angle is computed, the data slice at the closest azimuth is used, and the two points at the nearest distances to our current image point.

For a given point on the Cartesian grid, it is a relatively simple matter to determine the two nearest beams represented in the raw data. To facilitate this, the laser source location should be recorded when converting to the Cartesian grid. This point, \( x_s, y_s \), represents the laser source. Then, the azimuth, \( \theta \) (or elevation when working with vertical scans) can be computed as a function of \((x_s, y_s)\) and pixel coordinates \((x, y)\) as follows:
Fig. 2.4. Modified Bilinear Interpolation. Interpolation involves radial interpolation followed by angular interpolation.

\[ \theta = \arctan \frac{y - y_s}{x - x_s} \]  

Then, from a given image set, the two beams closest to this angle are considered for interpolation. For each of these beams, the two points surrounding the distance of interest are to be considered. The distance, \( d \), of the Cartesian point of interest can be computed in a similar way to the angle, as follows:

\[ d = \sqrt{(x - x_s)^2 + (y - y_s)^2} \]  

Then, to complete the modified bilinear interpolation, consider the values of the four nearest raw points, \( I(r_1, \theta_1) \), \( I(r_2, \theta_1) \), \( I(r_1, \theta_2) \), \( I(r_2, \theta_2) \), and the point to be interpolated, \( I(r, \theta) \). Figure 2.4 visually describes the interaction between the points. The intensity of the interpolated point is expressed as:
\[ I(r, \theta) = \frac{(\theta_2 - \theta)(r_2 - r)}{(\theta_2 - \theta_1)(r_2 - r_1)} I(r_1, \theta) \]

\[ + \frac{(\theta_2 - \theta)(r - r_1)}{(\theta_2 - \theta_1)(r_2 - r_1)} I(r_2, \theta_1) \]

\[ + \frac{(\theta - \theta_1)(r_2 - r)}{(\theta_2 - \theta_1)(r_2 - r_1)} I(r_1, \theta_2) \]

\[ + \frac{(\theta - \theta_1)(r - r_1)}{(\theta_2 - \theta_1)(r_2 - r_1)} I(r_2, \theta_2) \] (2.11)

When this operation is done for every pixel location, a well-interpolated image results. Note the primary difference between the bilinear interpolation and the Modified Bilinear Interpolation is that the second phase interpolates the intensities on a constant radius between the two nearest angles, whereas in the traditional bilinear interpolation the final stage of the interpolation is an interpolation on a straight line.

### 2.3 Temporal-Median Image Creation

After the images have been properly filtered and interpolated, a temporal-median image needs to be created. The temporal-median image is meant to provide a temporal average of the image over a time period. By averaging each median image (as computed in sections 2.1 and 2.2) over a given time, and subtracting this average from each individual median image, stationary features are removed from each image. These fixed features will then be removed as candidates for correlation and segmentation, depending on the tracking algorithm.
2.4 Histogram Normalization

In order to take advantage of the entire dynamic range of the image, a step of histogram normalization is performed. Typically, this would be done by utilizing the entire dynamic range. However, in this application, the resultant values are computed and stored as 64 bit precision floating point numbers. In this case, the quantization error is low, and the dynamic range of quantization is quite large. As a consequence, no additional histogram normalization was performed.

Although for data processing, the dynamic range is not an issue, the dynamic range is needed to be taken into account when generating imagery for the intermediate results. Due to the nature of this problem, there were many times when visualizing the image provided important insights into which features to analyze, and provided a way to qualitatively judge the algorithm performance. Because of this, histogram normalization needed to be performed.

To do this, while reading in the data for a particular scan sequence, noting the global minimum and maximum across all images was necessary. Then, by using the matplotlib method imshow, which accepts minimum and maximum values, the dynamic range gets adjusted automatically to the values specified. Once the histogram normalization is complete, the images can be displayed in sequence for smooth, continuous visualization.
Chapter 3

Preprocessing Methods for Image Correlation Wind Detection

The history of using cross correlation for tracking of wind features is detailed in [13]. In section 3.1, we revisit the correlation techniques done in the past. In section 3.2, we visit some new variations on these correlation techniques, in an attempt to improve the correlation results. Specifically, some nonlinear filtering is performed on the images prior to image correlation. For the purposes of this research, we will be examining in detail the differences between the base correlation method, and the correlation method with these preprocessing algorithms. Exhaustive wind detection over a large image is not the focus here.

Figures 3.3 and 3.4 show the results of thresholding the image to a hard limit, where all pixel values less than a threshold are set to zero. Finally, Figures 3.5 and 3.6 are the resulting images after performing bilateral filtering. The details of the preprocessing performed are in subsequent sections in this chapter.

For reference, the units on all images are expressed in pixels. Actual distances can be computed, given that the resolution of the images are 2 meters per pixel. Additionally, in Chapter 5, the wind calculations are expressed in meters per second.
3.1 Base Image Correlation

The details of the base image cross correlation method used in this thesis can be found in [7]. It is a straightforward cross correlation of two images, where the input images are same-sized rectangular sections from larger images, as described in Section 1.1.1. The larger source images are two consecutive lidar scans, registered with one another, which happens automatically according to the preprocessing described in Chapter 2.
In this section, the correlation algorithm above is modified by preprocessing algorithms. These algorithms are intended to condition the input signal to increase the peak to noise ratio on the resulting correlation peak. The improvement in peak quality is based on the idea that the preprocessing reduces or smooths noise present in the image. Section 3.2.1 provides details on a hard limiting algorithm, and section 3.2.2 explores the use of the bilateral filter on the input images prior to correlation. The image preprocessing techniques selected and detailed here are just examples of techniques that could
be used to enhance the images. They were selected because they are nonlinear filters, which will change the resulting peak in a way that reduces peak noise; however, different nonlinear techniques could be used in the future.

3.2.1 Hard Limiting Feature Extraction

This section explores the use of a hard limit threshold in eliminating weaker features or features not of interest in the input images. First, an intensity threshold is selected. Then, each pixel is visited, and if the pixel intensity value is less than the threshold, the intensity value is set to zero. This method is a type of crude image segmentation [14] via a histogram technique, where image pixel intensities above a certain level in the histogram are preserved, and everything else is zeroed.

There are several issues to consider when doing this. The first issue is selection of a threshold. This can be done with a histogram technique to determine a cutoff to nullify pixel intensities not of interest. Figures 3.3 and 3.4 show the results of applying a hard limiting threshold to the example images. It is obvious that the low-intensity features were eliminated, while the high intensity pixel values are retained. Optimum selection of a best threshold is left to future work.

3.2.2 Bilateral Filtering Feature Smoothing

In this section, the use of a bilateral filter is explored as an image preprocessor before the image correlation step. The bilateral filter smooths regions of the image with similar intensities, while preserving high spatial frequency edges. By doing this, images are better segmented, and as a result, tighter image value histograms result. These
tighter histograms are useful for segmentation techniques, if that is a chosen processing step. The filter also helps to isolate image features, and reduces background noise.

The bilateral filter, as defined in [15], is:

$$h(x) = k^{-1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\xi)c(\xi, x)s(f(\xi), f(x)) d\xi$$  \hspace{1cm} (3.1)$$

where $f(x)$ is the input image, $h(x)$ is the output image, $c(\xi, x)$ is the closeness function, $s(f(\xi), f(x))$ is the photometric similarity function, and $k^{-1}$ is the normalization factor, to preserve the input image amplitude, where $k$ is defined as:
Fig. 3.4. Correlation Frame 2 Subset Hard Limited

\[ k(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, x) s(f(\xi), f(x)) \, d\xi \]  

(3.2)

The closeness function and photometric similarity functions are described generically above, but the underlying function is often modeled as a Gaussian filter. The Gaussian filter is what was used for this current research; it is fully explained in [15]. Figures 3.5 and 3.6 show the results of bilateral filtering on our example images. Notice the smoothing of the background noise, while the feature edges have retained their sharpness.
To achieve the smoothing desired, a $13 \times 13$ square aperture filter was used, with a color sigma (photometric similarity) of 19, and a spatial sigma (closeness function) of 21. The filter size was selected to cover an expected region of similar wind attributes. A filter size that is too small would not allow much smoothing to be done, and a filter size that is too large would cause too much blurring. A 13 pixel square (26 meters on a side) filter provides an appropriate amount of smoothing, given the expected feature sizes. The filter used here comes from the openCV[16] Smooth operation configured as a bilateral filter.

Fig. 3.5. Correlation Frame 1 Subset Bilateral Filtered
Fig. 3.6. Correlation Frame 2 Subset Bilateral Filtered
Chapter 4

Image Registration

In this chapter, we look at the theory and application of image registration in regards to the lidar scan problem space. Image registration is the matching of multiple images that have different coordinate systems, for reasons including: images taken over different times, from different sensors, or from different viewpoints [17]. The act of registering images attempts to remove all variations that make the two images different from one another. The successful use of image registration is particularly challenging in the aerosol lidar application because the features being tracked are constantly deforming over time. The use of registration is demonstrated here to test the appropriateness of the application to this domain.

Although the correlation technique can be effective at extracting wind speed and direction measurements, there is a limitation in using the correlation method alone. If it is desired to actually track not just the wind speed and direction, but also the particulates that are traversing through the wind, then identification of the matter combined with registration between images is important in order to track the feature over time.

To test image registration, the cvCalcOpticalFlowPyrLK module from the openCV [16] module in python [18] was used. This module implements the Lucas-Kanade optical flow tracking algorithm [19]. The standard Lucas-Kanade algorithm is a least squares minimizer that estimates the motion of a frame in two dimensions. This estimation is
parameterized on the partial derivatives of the change in pixel intensities along each
dimension and across time.

Fig. 4.1. The first frame in the registration example with the tracking points circled.

Fig. 4.2. The second frame in the registration example with the tracking points circled. Notice the circled points roughly correspond with the points on the first frame.
Selection of the registration points is likely a difficult problem in itself; here, edge points of a well defined feature were manually selected as candidates for tracking. An edge detection algorithm and/or corner detection algorithm could potentially help with the automated selection of points of interest. This particular problem could cause more difficulty for edge detection algorithms due to the irregular edges, and their tendency to continuously morph.

The tracker is implemented in a pyramidal implementation, as described in [20]. The pyramidal implementation is an iterative implementation, where the pyramid is built recursively. The pyramid height (number of recursions) is typically 2, 3, or 4. In general, it is not worth creating a pyramid of more than 4 levels, due to diminishing returns. Each image and level $L$, $I^L(x,y)$, is defined as a function of the image at the previous level, $I^{L-1}(x,y)$, as follows:

$$I^L(x,y) = \frac{1}{4} I^{L-1}(2x,2y) + \frac{1}{8} (I^{L-1}(2x-1,2y)+I^{L-1}(2x+1,2y)+I^{L-1}(2x,2y-1)+I^{L-1}(2x,2y+1)) + \frac{1}{16} (I^{L-1}(2x-1,2y-1)+I^{L-1}(2x+1,2y+1)+I^{L-1}(2x-1,2y+1)+I^{L-1}(2x+1,2y+1))$$

(4.1)

The Lucas Kanade algorithm is used to compute the optical flow at the deepest pyramidal level $L_m$. The result is used as an initial guess to use the same algorithm at level $L_{m-1}$. This is repeated until image $L_0$ is reached, which gives a displacement estimate on the original image.
Chapter 5

Results

In this chapter, the results and analysis are described in comparing the correlation techniques of [7] and processed image correlations. Section 5.1 provides correlation results for the unmodified and preprocessed cases. Section 5.2 shows processing results from the Lucas Kanade registration technique. Finally, the results of all methods are compared in Section 5.3.

The examples shown here are from three different times/geographic locations from the REAL test data. The data snapshots here were taken on April 1, 2007 in the early morning hours. For each correlation method, the same three examples are used. The first example has a strong feature, one that obviously deforms, but maintains a roughly constant density. This example helps to test the ability of tracking a relatively easy “blob” feature. The second example shows strong features, but there is a higher amount of dissipation, and a significant amount of feature morphing when compared to the first example. Finally, the third example is a lower intensity feature image. This example is meant to really test the algorithms in scenarios where aerosol densities are quite low. The primary challenge with low density images is that the features have a tendency to blend in with the background noise.
5.1 Correlation Methods

The following sections describe the results from each of the correlation methods. Sections 5.1.1, 5.1.2, 5.1.3 show experimental results for unprocessed, hard limited, and bilateral filtered images, respectively. We will analyze the results and see what winds these correspond to.

5.1.1 Raw Correlation

This section contains results for raw correlations (i.e. the input images were not preprocessed prior to two dimensional cross correlation). The imagery that follows is for three separate examples, each highlighting interesting aspects of the processing. The three examples are broken out into Sections 5.1.1.1, 5.1.1.2, and 5.1.1.3.

5.1.1.1 Example 1

This example shows a prominent feature, relatively large in scale. The feature deforms slightly from the first to the second frame. A single prominent peak appears in the correlation result. Figures 5.1 and 5.2 are the input images, and Figure 5.3 is the output correlation.
Fig. 5.1. Example 1: Raw Input 1
Fig. 5.2. Example 1: Raw Input 2
Fig. 5.3. Example 1: Raw Correlation
5.1.1.2 Example 2

In Example 2, strong features are correlated, and a second peak emerges, reflecting a secondary component of the wind. Figures 5.4 and 5.5 are the input images for Example 2, and the output correlation is represented in Figure 5.6.

Fig. 5.4. Example 2: Raw Input 1
Fig. 5.5. Example 2: Raw Input 2
Fig. 5.6. Example 2: Raw Correlation
5.1.1.3 Example 3

Here, the correlation of subtle, weaker features is demonstrated. The variations in the correlation image reflect the uncertainty of the primary wind component. Figures 5.7 and 5.8 are the input images, and Figure 5.9 is the output correlation.

Fig. 5.7. Example 3: Raw Input 1
Fig. 5.8. Example 3: Raw Input 2
Fig. 5.9. Example 3: Raw Correlation
5.1.2 Hard Limited Correlation

In this section, the results are presented for the same three examples shown in Section 5.1.1, but the inputs are nonlinearly filtered with a hard limiting algorithm that nullifies all pixel values that are less than a predefined threshold.

5.1.2.1 Example 1

Here, when the hard limit is applied to the image, the desired feature is clearly enhanced in each frame (Figures 5.10 and 5.11). The correlation peak is clear, and the peak in Figure 5.12 is qualitatively better than the raw peak in Figure 5.3.
Fig. 5.10. Example 1: Hard Limited Input 1
Fig. 5.11. Example 1: Hard Limited Input 2
Fig. 5.12. Example 1: Hard Limited Correlation
5.1.2.2 Example 2

In this example, the high intensity features in Figures 5.13 and 5.14 are preserved. This yields a peak in Figure 5.15 similar to the raw correlation in Figure 5.3, but as in Example 1, the peak variance is smaller. Additionally, the secondary peak is more prevalent in the raw correlation than in the hard limited correlation, which indicates that the hard limiting operation is especially good at enhancing the primary component.

![Hard Limited Image 1](image.png)

Fig. 5.13. Example 2: Hard Limited Input 1
Fig. 5.14. Example 2: Hard Limited Input 2
Fig. 5.15. Example 2: Hard Limited Correlation
5.1.2.3 Example 3

This example highlights the effectiveness of the hard limiting algorithm in reducing the ambiguity of the peaks in the final correlation. The raw features are less intense than those in Examples 1 and 2; however, an appropriate threshold is chosen to highlight enough of the features.

Fig. 5.16. Example 3: Hard Limited Input 1
Fig. 5.17. Example 3: Hard Limited Input 2
Fig. 5.18. Example 3: Hard Limited Correlation
5.1.3 Bilateral Filtered Correlation

The examples for correlation with bilateral filtering conditioning are shown in Sections 5.1.3.1, 5.1.3.2, 5.1.3.3.

5.1.3.1 Example 1

In this example, the filtered images show strong features, with smoothed background, as seen in Figures 5.19 and 5.20. The smoothing of the features and the background serve to smooth similar intensity regions, which have the net effect of reducing the variance on the peak in Figure 5.21.
Fig. 5.19. Example 1: Bilateral Filtered Input 1
Fig. 5.20. Example 1: Bilateral Filtered Input 2
Fig. 5.21. Example 1: Bilateral Filtered Correlation
5.1.3.2 Example 2

In this example, the bilateral filter applied to the input images produce the images in Figures 5.22 and 5.23. As in Example 1, the strong features are preserved, albeit smoother, and the lower intensity pixels are smoothed. This creates a correlation surface in Figure 5.24, that, as in Example 2 of the hard limited algorithm, enhances the intensity of the primary peak while attenuating the secondary peak due to a secondary shift. This peak modification could be advantageous, especially if there is an interest in unambiguously selecting the strongest component of wind.

Fig. 5.22. Example 2: Bilateral Filtered Input 1
Fig. 5.23. Example 2: Bilateral Filtered Input 2
Fig. 5.24. Example 2: Bilateral Filtered Correlation
5.1.3.3 Example 3

Figures 5.25 and 5.26 are the bilaterally filtered images for Example 3. The correlation surface in Figure 5.27 provides an interesting result. Simply by smoothing the features themselves, which are not much stronger than the background, the peak is considerably less noisy than the raw peak in Figure 5.9. Interestingly, the wide secondary peak in the raw correlation result almost completely disappears. This suggests that the features are perhaps smoothed together, blurring where the pixel intensities are not sufficiently different to be smoothed independently.

Fig. 5.25. Example 3: Bilateral Filtered Input 1
Fig. 5.26. Example 3: Bilateral Filtered Input 2
Fig. 5.27. Example 3: Bilateral Filtered Correlation
5.2 Registration Method

This section shows the imagery of the input and output of the Lucas Kanade image registration/tracking algorithm. Sections 5.2.1, 5.2.2, and 5.2.3 show results of the three examples used for correlation processing.

5.2.1 Example 1

In Example 1, a very prominent feature is present, and the Lucas Kanade algorithm does an excellent job of tracking the feature, even through a slight deformation of the feature. The first image with selected control points is in Figure 5.28, and the second image with computed control points is in Figure 5.29.

Fig. 5.28. Example 1: Registration Frame 1
5.2.2 Example 2

For Example 2, a strong feature is present, and its movement is tracked using the control points selected. The first image with selected control points is in Figure 5.30, and the second image with computed control points is in Figure 5.31.
5.2.3 Example 3

Here, the weakness of the Lucas Kanade algorithm for this particular application is shown. Since there are no strong features, the algorithm struggles to find the correct displacement from frame to frame. The first image with selected control points is in Figure 5.32, and the second image with computed control points is in Figure 5.33.

5.3 Results Comparison

A good metric for comparison of the correlation results is to measure the statistical variance of the peaks. Table 5.1 shows the pixel variances of the different smoothing
Fig. 5.33. Example 3: Registration Frame 2

techniques for all three examples. Variances from table are computed using random variable variance technique as described in [21]:

$$\text{var} = \frac{\sum p_i (x_i - E[x])^2}{\sum p_i}$$  \hspace{1cm} (5.1)

where $E[x]$ denotes the expected value of $x$, and $p_i$ and $x_i$ are the pixel intensity and pixel index, respectively. The variances are computed in the $x$ and $y$ direction that contain the maximum value, and a resultant variance is estimated based on these two values as:

$$\text{var} = \sqrt{\text{var}_x^2 + \text{var}_y^2}$$  \hspace{1cm} (5.2)

By this measure, for these examples, the hard limiting algorithm has the least variance on the peak, followed by the bilateral filtering algorithm. As expected, and as can be seen by visual comparison of the correlation images, the raw processed correlation peak has the most variance on the peak. The next step would be to compare how this variance changes as the hard-limit threshold changes.
Table 5.1. Variance of peaks. Units are in pixels

<table>
<thead>
<tr>
<th>Correlation Type</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>3956</td>
<td>1998</td>
<td>2156</td>
</tr>
<tr>
<td>Hard Limited</td>
<td>345</td>
<td>300</td>
<td>444</td>
</tr>
<tr>
<td>Bilateral Filter</td>
<td>510</td>
<td>350</td>
<td>771</td>
</tr>
</tbody>
</table>

In [13], Mayor et al has already compared the correlation method wind estimates with the measurements from a sonic anemometer. Therefore, the goal here is not to compare the correlation results to a known ground truth, but rather, to measure the difference in the estimate when applying preprocessing techniques before the correlation, or performing a completely different wind detection technique (the Lucas Kanade registration technique, in this case).

In Tables 5.2 and 5.3, the wind calculation results are shown. These results are in pixels to maintain uniformity with all measurements elsewhere, but the measurements can be converted to speed in \( \frac{m}{s} \) through the following equivalence:

\[
v_x = p_x \times \frac{\text{res}_x}{T_s} \quad (5.3)
\]

\[
v_y = p_y \times \frac{\text{res}_y}{T_s} \quad (5.4)
\]

where \( \text{res}_x = \text{res}_y = 2 \) are the \( x \) and \( y \) resolutions of the images in \( \frac{\text{meters}}{\text{pixel}} \), \( T_s = 10.9 \) is the scanning period in seconds, \( p_x \) and \( p_y \) are the speed measurements in \( \frac{\text{pixels}}{\text{frame}} \).

Table 5.4 shows an estimate of the computational complexity of the different processing algorithms discussed. This estimate comes from a direct measurement of the
Table 5.2. Comparison of correlation and registration wind velocity calculation results. Results are given in (x,y) pixel offsets.

execution times of the different examples tested. The Lucas Kanade algorithm is the clear winner in terms of execution time; however, any control point selection algorithm that would be used is not counted here. The dominant contributor to the execution time of the correlation algorithms is the two-dimensional cross correlation that is performed after preprocessing. The correlation could potentially be optimized, which may change the results here.

Table 5.3. Comparison of correlation and registration wind velocity calculation results. Results are given in m/s, angle pairs.

Table 5.4. Algorithm Execution Times in seconds. Hard Limiting and Bilateral Filtering are dominated by the correlation operation. Lucas Kanade does not include any point selection required for an automated process.
Chapter 6

Conclusions

In this thesis, it has been shown that the image correlation process for wind detection has the potential to be improved by using one of any number of nonlinear preprocessing image filters. Here, the concept of using image registration to track wind speed has also been shown. Section 6.1 provides an overall summary of the findings of this research. However, there are many additional activities that can be done to improve and verify these capabilities. In Section 6.2, the potential work that is a direct consequence of this research is described.

6.1 Summary

So, we compared the raw image correlation with correlation using two different preprocessing methods: hard limit thresholding and bilateral filtering. These different methods were tested on three different examples, and the peak qualities were quantified relative to one another. In addition to the correlation methods, we also examined the potential of using pyramidal Lucas Kanade feature tracking to follow the individual aerosol features.
The best overall algorithm in terms of minimizing variance, given the examples and processing parameters selected, is the hard limiting threshold algorithm as a preprocessor to the correlation. Although Lucas Kanade is potentially more efficient in terms of execution time, there are limitations when the features are not strong.

6.2 Future Work

The research done here leads to additional opportunities to continue approaching the aerosol wind detection problem from an image processing standpoint. Quantification of performance for a whole image sequence should be done using the smoothing and registration techniques described. Feature tracking using a method other than strict correlation should be investigated. If correlation is to be investigated further, the possibility of dynamic window shaping should be considered; selection of a window size based on the size of the feature could likely be more valuable in understanding the deformation characteristics of the individual features. On the contrary, using a smaller window size helps to interpret vorticity and shear deformations. Multiple levels of correlation could lead to more interesting conclusions. Affects of doing temporal filtering should also be investigated.

Selection of a threshold for the hard limiting algorithm could be optimized. Histogram techniques exist to aid in selection of the threshold. Otsu’s method may be useful in forming two or more classes for histogram grouping, which could help in selection of an optimum threshold.

The use of data that is not planar should also be considered. The REAL system captures data that is constant azimuth or constant elevation; three dimensional volume
scanning lidar systems exist. These systems present a unique challenge with an increase in the dimensionality of the problem. Grouping/clustering and orthorectification may be steps needed to handle the problem in this additional dimension.

The use of image registration in this problem is a largely unexplored area; as a consequence, there are many aspects of the problem that can be researched further. Image registration point selection is a critical part of the problem to be solved if registration is to be successfully automated with lidar data. Then, assuming the tracking points are well selected, and the registration process successfully tracks the wind features, classification of the movements is an important part of the problem to solve. This is solving a higher level of the problem, but extraction of this information could be very desirable. For example, it may be useful to identify where there is a high degree of vorticity or wind shear. Perhaps detection of a constant wind in a localized area is important. Each of these things could be a consequence of a successful image registration processing thread.

The lidar data here was processed to form two dimensional images for processing; however, there are additional steps that need to be taken to create images of the highest fidelity. Since the process of capturing the lidar data in a single frame has a finite duration (in the case of REAL, somewhere between 10 and 30 seconds, depending on azimuth spread), the images converted herein experience a temporal drift. That is, they are not snapshots in time, but rather an approximation that is roughly approximate. Combining the processing described with some kind of temporal interpolation between frames (such as B-spline interpolation) could help to create an image that more closely matches an aerosol density map in a single instant.
For the correlation aspect of the problem, there are several ways that the preprocessing ideas presented here could be extended. First, a step could be inserted which calculates where the features are, using some image segmentation techniques, instead of slicing the image into fixed size windows. Also, the correlation could be performed on multiple levels. For example, small image windows could be used, along with a larger window that encapsulates several smaller windows. The results of these correlations could be considered jointly to determine a wind estimate, or to determine large and small scale winds for a geographic region. Alternatively, a phase correlation technique, such as phase congruency, could be used, which may be advantageous for tracking non-rigid features.

The registration part of the problem can be explored many different ways in depth. First, selection of the tracking points is a very important part of the problem that needs to be solved. A corner detection algorithm, or similar feature detector, could be used to find optimum tracking points. Then, as part of post-processing of the point tracking, wind feature classification could be performed to determine the type of air motion, including: shear, vorticity, and turbulence. Adjacent features could be considered together at a higher level to glean some information about the overall optical flow. Finally, there is a lot of room to optimize the performance of these algorithms to be time efficient.
Appendix A

REAL Lidar Data Extraction

For reference, the REAL lidar data format is detailed in this appendix. The data itself consists of fixed length, contiguous blocks. All members are 32-bit floating point numbers. The data blocks consist of a fixed length header, followed by a variable length number of data samples, whose length is based on the number of words per channel value in the header. Each data record consists of two separate channels, and pertains to a single lidar return. Therefore, for a given image, many data records must be read in to cover the appropriate azimuth/elevation angle range. The 32-bit float fields in the header are detailed below, in the order they appear in the header.

- **hour**: This is the hour of day the data is collected
- **minute**: This is the minute of day the data is collected
- **second**: This is the second field corresponding to the time the data is collected
- **(blank)**: No data
- **(blank)**: No data
- **azimuth**: Azimuth angle (in degrees) relative to zero degrees north, where the lidar instrument is pointing at the time of data collection
- **elevation**: Elevation angle (in degrees) relative to horizon where the lidar instrument is pointing at the time of data collection
- **(blank)**: No data
- **(blank)**: No data
• (blank): No data
• month: Month of the year when the collection occurs
• day: Day of the month when the collection occurs
• year: Year the collection occurs
• (blank): No data
• (blank): No data
• (blank): No data
• (blank): No data
• latitude: Latitude in degrees (fractional possible) of instrument
• longitude: Longitude in degrees (fractional possible) of instrument
• range: Range (in meters) of the first data sample. Ranges of subsequent samples are \( \text{range} + k \times R \) where \( k = 1 \ldots N \), \( N \) is the number of data samples in a record, and \( R \) is the range resolution
• range resolution: Linear distance (in meters) between samples on lidar return
• (blank): No data
• thirty: Thirty fields in the header (\( 30 \times 4 \text{ bytes field} = 120 \text{ bytes} \))
• (blank): No data
• (blank): No data
• energy1: Laser energy measurement (not working properly)
• energy2: Laser energy measurement (not working properly)
• number of words per channel: Number of words (4 byte floats) per channel, to be read immediately following header
• scan number: Incremental scan number that this shot belongs to
Appendix B

Code Summary

This appendix explains the structure and functionality of the code used to complete the work described in this thesis. The work performed in this thesis was done primarily in the Python programming language. To facilitate mathematical operations, the numpy and matplotlib libraries were imported to enable MATLAB like numerical processing and data/image plotting.

- **correlation.py**: executes the correlation for the pre-selected image subsets. Moments are computed and output for all peaks. Images are saved to current directory.

- **lidar.py**: provide raw REAL data reading and conversion via the read method.

- **make_movie.py**: stitch all images from the lidar scans into a single movie; useful for identifying the wind features with the eye.

- **read_translate.py**: provides raw REAL data reading and filtering (processing described in Chapter 2).

- **registration.py**: perform registration on pre-selected image subsets, and save images to current directory.
References


