11-14-2013

# A Generalized Adaptive Mathematical Morphological Filter for LIDAR Data

Zheng Cui

*Florida International University*, cuiz@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A GENERALIZED ADAPTIVE MATHEMATICAL MORPHOLOGICAL FILTER

FOR LIDAR DATA

A dissertation submitted in partial fulfillment of

the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Zheng Cui

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Zheng Cui, and entitled A Generalized Adaptive Mathematical Morphological Filter for LIDAR Data, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

_____
Xudong He

_____
Nagarajan Prabakar

_____
Peter J Clarke

_____
Keqi Zhang, Co-Major Professor

_____
Shu-Ching Chen, Co-Major Professor

Date of Defense: November 14, 2013

The dissertation of Zheng Cui is approved.

_____
Dean Amir Mirmiran
College of Engineering and Computing

_____
Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2013

DEDICATION

I dedicate this dissertation to my dear wife, Yue Wang, for her love, understanding, and constant support; to my lovely son, Jeffrey Cui; and to my father, Guowen Cui, and my mother, Boli Cui, for their love and encouragement.

# ACKNOWLEDGMENTS

ABSTRACT OF THE DISSERTATION

A GENERALIZED ADAPTIVE MATHEMATICAL MORPHOLOGICAL FILTER

FOR LIDAR DATA

by

Zheng Cui

Florida International University, 2013

Miami, Florida

Professor Shu-Ching Chen, Co-Major Professor

Professor Keqi Zhang, Co-Major Professor

Airborne Light Detection and Ranging (LIDAR) technology has become the primary method to derive high-resolution Digital Terrain Models (DTMs), which are essential for studying Earth's surface processes, such as flooding and landslides. The critical step in generating a DTM is to separate ground and non-ground measurements in a voluminous point LIDAR dataset, using a filter, because the DTM is created by interpolating ground points. As one of widely used filtering methods, the progressive morphological (PM) filter has the advantages of classifying the LIDAR data at the point level, a linear computational complexity, and preserving the geometric shapes of terrain features. The filter works well in an urban setting with a gentle slope and a mixture of vegetation and buildings. However, the PM filter often removes ground measurements incorrectly at the topographic high area, along with large sizes of non-ground objects, because it uses a constant threshold slope, resulting in "cut-off" errors. A novel cluster analysis method was developed in this study and incorporated into the PM filter to prevent the removal of the ground measurements at topographic highs.

Furthermore, to obtain the optimal filtering results for an area with undulating terrain, a trend analysis method was developed to adaptively estimate the slope-related thresholds of the PM filter based on changes of topographic slopes and the characteristics of non-terrain objects. The comparison of the PM and generalized adaptive PM (GAPM) filters for selected study areas indicates that the GAPM filter preserves the most "cut-off" points removed incorrectly by the PM filter. The application of the GAPM filter to seven ISPRS benchmark datasets shows that the GAPM filter reduces the filtering error by 20% on average, compared with the method used by the popular commercial software TerraScan. The combination of the cluster method, adaptive trend analysis, and the PM filter allows users without much experience in processing LIDAR data to effectively and efficiently identify ground measurements for the complex terrains in a large LIDAR data set. The GAPM filter is highly automatic and requires little human input. Therefore, it can significantly reduce the effort of manually processing voluminous LIDAR measurements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

**Chapter 1**

**INTRODUCTION**

LIDAR (Light Detection And Ranging) has become a widely used technology in surveying and industrial measurement applications in recent years. Ground-based scanning LIDAR systems can produce very high resolution point clouds of three-dimensional objects at millimeter accuracy; three-dimensional models, such as DEMs [2], DTMs, and digital surface models (DSMs), can be generated from these high-resolution topographic data. These topographic information is critical for a wide variety of applications, including engineering projects (e.g., transportation, mining reclamations, urban planning), hydrology and floodplain management, corridor mapping (e.g., for roads, telecommunications), landside analysis, geological studies, and natural-resource assessments. Therefore, effective and efficient methods to analyze and process these three-dimensional data are very important to all the applications [3].

The general processing of ground-based LIDAR data has some major components for most of the applications. It includes data interpolation, data filtering, objects classification, and model generation. Many methods have been developed in each component. Among these methods, the filtering methods which focus on the separation of ground and non-ground objects are very critical to most of the applications. Since LIDAR data is normally high-resolution data, there would be huge volume of data points in the surveyed area; it is very important to develop some automatic processing methods to separate ground and non-ground objects with as little human interactions as possible. Although many filtering methods have been studied, they all have their own advantages and limitations on different kinds of terrain types. It is very difficult for any method to

handle a data set with complex terrain types mixed together. Due to this reason, an adaptive filtering solution was proposed in this dissertation. It will provide an adaptive framework for the filtering method to make the LIDAR processing more automatic and rely less on human interactions.

## 1.1    Challenges

Since the LIDAR data normally contains a huge number of points in the surveyed area, it would make the processing of them very difficult in terms of efficiency, effectiveness and automation. The most challenging work of LIDAR data processing is to find some methods to process varied data sets in a generalized way without human interactions [12]. LIDAR data processing normally involves some common procedures, such as data retrieving, storage, interpolation, filtering, and model generation. Each procedure needs to have appropriate methods to accomplish the task in terms of correctness, efficiency, effectiveness, and automation. Among these procedures, data retrieving and storage will affect how fast the data can be read and written, which will influence the processing speed significantly. Another critical issue of data retrieving and storage is how to retrieve the data for different processing purposes. Since one surveyed area could contain different terrain types and a tremendous number of data points, how to split the data into smaller parts and process them separately is very critical for the effectiveness of different interpolation and filtering methods. Because different methods are suited for different terrain types, smaller pieces of a large data set would speed up the processing significantly for some methods.

The interpolation procedure is a very important preparation step for the filtering procedure. Most grid-based filtering methods need to have interpolation before filtering. Since the data set will be partitioned into grids before filtering, some of the grids would be empty; therefore good interpolation methods would guarantee the correctness and effectiveness of the filtering methods. Some filtering methods might need special interpolation to make them work properly.

Filtering methods are the key procedure for the LIDAR processing in that these methods will identify and extract different points based on the application needs. The correctness and effectiveness of the filtering procedure would affect the model generation results significantly. Therefore, this will be the major topic discussed in this dissertation.

Model generation is very important for many applications, because it is a more intuitive way for the user to view the data. Good model generation methods would help people analyze and research the data in an efficient and interactive way.

## 1.2 Proposed Methods

In this dissertation, several methods have been proposed for the interpolation and filtering procedure in the LIDAR processing. Many experiments have been done on different terrain types. The results demonstrated the improvement of interpolation and filtering quality.

### 1.2.1 Outlier Removal and Grid Interpolation

Most of our LIDAR processing methods are based on the grid data structure, therefore, there would be some empty grids left after a data set was gridded. Accordingly,

the interpolation procedure is very important before the filtering process, in that it will remove non-related points, such as outlier points, and interpolate useful points in the empty grids. In this dissertation, some morphological outlier removal methods were discussed to filter out extremely high and low elevation points from the original data set.

Different filtering methods might need some special interpolation methods to fill meaningful points into the empty grids in order to benefit the filtering procedure. In this dissertation, a grid-based priority interpolation method was proposed to interpolate empty grids based on customized priority. This solution would benefit the filtering method in certain terrain types [11]. It can be used for a multi-pass filtering solution. In this dissertation, a multi-pass morphological filter was proposed to process the data set in multiple rounds. The grid-based priority interpolation method was used between each filtering pass. This multi-pass morphological filter offers a means to filter the terrain with large non-ground features under relatively small filtering window, while the morphological filter itself has difficulty on these complex terrain data sets.

## 1.2.2   Cluster-Based Morphological Filter

Mathematical morphology is an important methodology in LIDAR filtering. It has many variant methods, which are normally grid-based filtering. However, due to the method's intrinsic weakness, they would normally have top cut-off problem, such as cut-off of the mountain ridge. This problem would be deteriorated when the morphological filtering window increases to a large size. It would cut off many ground surface points, especially on the undulated terrain. In order to solve this problem, a cluster analysis mechanism [10] was introduced to help judge the ground and non-ground points during

morphological filtering. Combined with morphological filtering, this cluster analysis method would prevent the cut-off problem on many complex terrain types. Also, this cluster-based method would provide a foundation for the adaptive trend analysis method proposed in the next section.

## 1.2.3   Adaptive Trend Analysis Morphological Filter

Since the morphological filter normally uses windows with different sizes to filter the data, for each window size, a different threshold value has to be selected as the criteria for separating ground and non-ground objects. Most of the morphological filtering methods use the constant value for each window size, which would make the filter not suitable for the complex terrain types. Because constant threshold values for the same morphological filter window implies uniform ground slope in the data set, it would work well in the simple terrain areas, such as flat terrain, while it would not work very well in the complex terrain areas, especially undulating terrain, such as mountain areas. An adaptive trend analysis method [12] was proposed in this dissertation, which was combined with the progressive morphological filter, to make the filter automatically select the filtering thresholds for all the points under different filtering window sizes according to the local terrain variation.

## 1.2.4   GUI-based LIDAR Data Processing System

A GUI-based LIDAR data processing system is indispensable for the LIDAR research. Users need to interactively view the data source, in order to select appropriate methods to process the data set according to the application requirements. The processing

results also need to be displayed in a proper way and analyzed for research and processing purposes. Therefore, a GUI-based LIDAR processing tool was proposed for users to display and process the data interactively. Since LIDAR data processing normally involves huge volumes of data and many source data files, a batch processing tools was proposed in our GUI-based LIDAR processing system. Users can select the process methods from a list of implemented methods, configure the processing parameters, and add different kinds of jobs into a task list for batch processing.

1.3    Contributions

The methods proposed in this dissertation have effectively solved the problem in the interpolation and filtering procedures of LIDAR processing. They are combined to provide a complete set of LIDAR processing solutions. This set of solutions streamlines many LIDAR processing procedures with fewer human interactions.

First, a grid-based priority interpolation method was proposed to interpolate empty grids based on the empty grids' boundary. It provides a mechanism to interpolate empty grids according to the boundary points around a filtering shape with customized priority. Users can define the interpolation priority according to their application needs. This interpolation method helps filtering method effectively remove the filtering remains from large non-ground objects.

Second, combined with the grid-based priority interpolation, a multi-pass morphological filter was proposed to process a complex data set in multiple rounds. It would help the filtering of the terrain with large non-ground objects that cannot use large

filtering window. The filtering remains of large non-ground objects will be removed in multiple filtering passes.

Third, a cluster analysis method was proposed to provide a mechanism for filtering methods to adjust the filtering results by analyzing the cluster property, which shows the local terrain shape in general. A new filter, called cluster-based morphological filter was proposed by combining this cluster analysis method with the progressive morphological filter [51]. It improves the progressive morphological filter on undulating and complex terrains. It can also be embedded into some other filters in order to correct errors and improve results.

Fourth, combined with the cluster analysis method, a trend analysis method was proposed to provide a mechanism for filtering methods to adaptively select filtering parameters according to the local terrain shape, because it can automatically analyze and detect the variation of local terrain. A new filter, called adaptive trend analysis morphological filter, was proposed by embedding this trend analysis method into the progressive morphological filter [51]. This filter provides a generalized filtering mechanism to filter different terrain types by adaptively selecting filtering parameters. This mechanism can be extended to dynamically select different filtering methods for different parts of a data set, based on the terrain type.

Fifth, many LIDAR data processing related methods have been implemented in a GUI-based LIDAR processing system software, which offers researchers and users a good tool to display, analyze and process the LIDAR data. It also provides the researchers a mechanism to test and compare the performance of different LIDAR processing related methods.

1.4    Scope and Limitations

This dissertation mainly focuses on the study of LIDAR filtering methods. It has the following limitations:

1. Since the proposed filtering methods are based on the three dimensional information of LIDAR data, it is very hard to distinguish an artificial terrain shape with human construction or landscape. For example, if a terrain was cut or landscaped as a shape of building, it is hard for the filter to decide what points should be removed.

2. The filtering results depend on the resolution of LIDAR data. If the data source does not have enough points due to the density, it would affect the filtering results significantly. Furthermore, if the data source has too many empty parts, it would affect the filtering results when processing the data set as one single area. For example, if the surveyed area is a long and narrow strip, and the boundary of the data set is very large, it would make a large empty area between the boundary and data strip.

3. It is difficult for a filtering method to automatically detect some terrain features' information, such as the largest non-ground object's size, at the very beginning of the processing. Therefore, some parameters, such as maximum filtering window size, are difficult to select programmatically. These parameters might need to be selected by users through visual analysis.

4. When comparing the accuracy of different filters, there would be some subtle differences. For example, for grid-based methods, the accuracy of the filtering results is based on the gridded results of reference points, while the accuracy of non-grid

methods is based directly on the reference points. There would be some arguments on the accuracy between different types of methods. Therefore, it would be better to have the same standard to compare different types of filters. For example, we can grid the non-grid method's results to compare them with the grid-based method's results. However, it might not be perfectly appropriate to use the same standard to judge the accuracy of different methods.

## 1.5 Outline

The organization of this dissertation is as follows:

Chapter 2 gives the literature and background reviews in the areas of LIDAR data interpolation, filtering and model generation methodologies.

Chapter 3 focuses on an overview of our LIDAR processing framework with all its components. Each component is briefly introduced and discussed, and the relationships between components are shown.

Chapter 4 discusses the data pre-processing step in our framework, exceptional data handling and data interpolation in detail. A gird-based priority interpolation method and a multi-pass morphological filter are proposed and discussed in detail.

Chapter 5 discusses the proposed cluster-based morphological filter in detail, which helps the morphological filter solve the cut-off problem of filtering certain types of terrains.

Chapter 6 discusses the proposed trend analysis method in detail, which is combined with the cluster method introduced in chapter 5. It provides a mechanism for the filters to automatically and adaptively estimate filtering parameters, which are related

to the terrain shape. It can be coupled with any filter that needs local terrain shape information, and makes the filter parameters' selection automatically and adaptively with fewer human interactions.

Chapter 7 introduces a GUI-based LIDAR data processing system for model generation and mapping. It includes many LIDAR processing methods developed in our research work and other commonly used LIDAR processing related methods.

Chapter 8 summarizes with the conclusions and the future work of this research.

**Chapter 2**

**BACKGROUND AND RELATED WORK**

In this chapter, a detailed survey of LIDAR technology and data processing methods are overviewed. It covers the existing research in the fields that are related to the important components of the proposed framework, which is discussed in the following chapters.

The general processing of ground-based LIDAR data has some major components for most of the applications. It includes data interpolation, data filtering, objects classification, feature extraction, and model generation. Many methods have been developed in each part. Data interpolation is a very important preparation step for many filtering methods. Normally it involves the error point removal (e.g. outliers and redundant point removal), data gridding, and empty grids interpolation. The filtering methods are the key part of the LIDAR data processing, which would provide appropriate filtering results according to different application purposes.

2.1 Airborne LIDAR Data Collection

In this section, LIDAR technology, mapping, and data acquisition are overviewed. The general LIDAR data collection procedure will be explained. In addition, the common data format of LIDAR will be introduced.

2.1.1   What is LIDAR

LIDAR stands for Light Detection And Ranging, which is a widely used remote sensing technology in recent years. It uses a laser beam to bounce between the aircraft

and ground, assessing distances from the camera to the ground at set points. The laser beams can penetrate vegetation, allowing measurement to be conducted, even when vegetation is thick. LIDAR surveys satisfied many topographic mapping needs for various applications.

2.1.2    Airborne Laser Mapping

Airborne laser mapping is an emerging technology in the field of remote sensing, which is capable of rapidly generating high-density, geo-referenced digital elevation data with an accuracy equivalent to traditional land surveys but significantly faster than traditional airborne surveys.

Compared to traditional survey methods, airborne laser mapping offers lower operation costs and post-processing costs. The cost to produce the data is significantly less than other types of traditional topographic data collection, which makes it an attractive technology for a variety of survey applications. These survey applications can provide low cost, high-density, and high-accuracy geo-referenced digital elevation data.

Airborne laser mapping uses a combination of three mature technologies: Light Detection And Ranging (LIDAR), highly accurate Inertial Reference Systems (IRS) and the Global Positioning System (GPS). By integrating these subsystems into a single instrument mounted in a small airplane or helicopter, it is possible to rapidly produce accurate digital topographic maps of the terrain beneath the flight path of the aircraft.

The absolute accuracy of the elevation data can be less than 15 cm, and the relative accuracy can be less than 5 cm. The absolute accuracy of the data depends on the operating parameters, such as flight altitude, but is usually from 5 cm to 1 meter.

2.1.3   Data Acquisition, Storage and Processing

Airborne laser mapping technology is illustrated in Figure 2-1, which shows the general idea of the airborne laser mapping mechanism. This system was used for the airborne laser terrain mapping in Broward County of Florida.
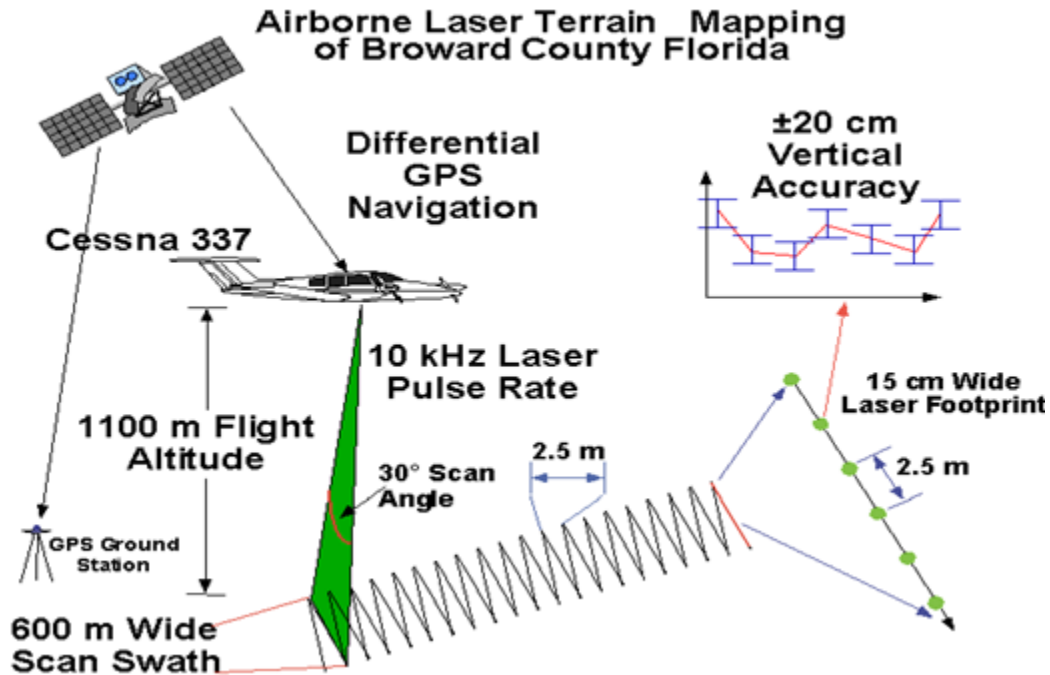


Figure 2-1 Airborne laser mapping project in Broward County Florida

In terms of operational procedure, a pulsed laser range finder mounted in the aircraft accurately measures the distance to the ground by recording the time in which a laser pulse shoots and reflects back to the aircraft from the ground or from objects such as constructions and vegetation. Since the speed of light is known, the elapsed time is converted to an accurate distance or slant range. Some instruments record multiple returns from a single laser pulse to capture a vertical profile along the slant range. A scanning or rotating mirror is used to provide coverage across the path of the aircraft with swath widths dependent on scan angle and operating altitude. Simultaneously, the IRS

subsystem records the roll, pitch and heading of the aircraft to determine its orientation in space, while the GPS subsystem provides the precise location of the aircraft through a differential kinematic solution. During post-processing, the IRS orientation and GPS position solutions are combined with the laser slant ranges to accurately calculate three-dimensional coordinates for each laser return.

After each flight, LIDAR and GPS data are downloaded to a computer and processed by proprietary software (e.g. Optech Airborne LIDAR Software) to produce Universal Transverse Mercator (UTM) X, Y coordinates and ellipsoidal heights of each laser return. Positional accuracy was improved by calculating a precise aircraft trajectory using the KARS software. Elevations were converted from GPS ellipsoidal heights to NAVD88 orthometric heights with the NGS GEOID99 model. Data from overlapping swaths were checked for internal consistency, combined and subdivided into over tiles (e.g. hundreds of 1-km$^2$ tiles). Each tile was then gridded using the nearest neighboring interpolation to produce a certain resolution DEMs.

Airborne laser mapping data is stored in ASCII files, shown in Table 2-1. Each line in the file represents a point that consists of four elements: three-dimensional coordinates and the laser reflection intensity of the object at this point. Normally, there are millions of points for a study area. For example, 140 million points are surveyed for the Florida Broward County hurricane flood vulnerable area (140 km$^2$). It is difficult to process them together based on the capacity of current workstations. Thus, the data set needs to be divided into smaller pieces for further processing. After processing each piece of data, the results have to be merged for the final outcome.

Table 2-1 Format of ALTM data in ASCII file

| X | Y | Z | Intensity |
|---|---|---|---|
| 573200.00 | 2891200.00 | -19.21 | 96 |
| 573400.00 | 2891250.20 | -23.57 | 75 |
| 573600.28 | 2891200.00 | -23.02 | 72 |
| 573500.25 | 2891050.96 | -23.40 | 13 |
| 573300.52 | 2891050.88 | -19.57 | 119 |

## 2.2  Interpolation of Data

After the LIDAR data collection, three-dimensional information and intensity information of the scanned terrain were acquired. This information consisted of the major objects on which our research focused. The first challenge of our research is how to interpolate the data for different purposes. There are many interpolation methods which can be used for LIDAR data. Most of them are region based methods, which means the data set will be partitioned into regions such as grids, blocks, and triangles, etc. The empty regions would be interpolated by non-empty regions in some ways. The most popular interpolation methods normally belong to some common categories, such as spatial relationship methods, statistical methods, and curve-fitting methods.

## 2.2.1  Spatial Relationship Interpolation Methods

In the spatial relationship interpolation methods, the Nearest Neighbor method is a popular one, because it not only runs quickly but effectively for many cases as well. This method is very effective and efficient, especially when data are evenly distributed. If

there are only small numbers of empty grids in the data set, it would be very efficient to use this method to fill the empty grids by interpolating them with neighbor grids. This method is the most commonly used interpolation method in our LIDAR processing framework too, because LIDAR data are normally enormous and evenly distributed.

Triangulation interpolation is another popular spatial interpolation method, which is normally based on Delaunay triangulation [4]. It uses the original data points to create triangles by connecting between data points. The points would be connected in such a way that no triangle edges are intersected by other triangles. Each triangle would form a surface, which could be used to interpolate each empty grid. This method is also very effective on the data set distributed evenly, but the computation time would be much longer.

## 2.2.2    Statistical Interpolation Methods

Geostatistical interpolation techniques [25] are based on statistics and are used for more advanced prediction surface modeling. Kriging [25] is a very popular geostatistical interpolation method, which can produce visually engaging maps from irregularly spaced data. Kriging belongs to the family of linear least squares estimation algorithms, and it is a customizable method to achieve accurate or smoothing interpolation by specifying the appropriate variogram model.

2.2.3    Curve Fitting Interpolation Methods

Minimum Curvature is a popular interpolation method in earth science. This method will generate a smoothest possible surface as a thin and linearly elastic plate passing through each of the data values with a minimum amount of bending [50].

The local polynomial method is another useful interpolation method, which interpolates empty grids by using weighted least squares fit within a certain search range.

2.3    Filtering Methods

Since LIDAR data includes tremendous information of ground and non-ground objects, how to automatically and efficiently acquire a high-quality DEM (Digital Elevation Model) according to the user's requirement has been a hot topic in recent years. To achieve high-quality DEM, the filtering methods are the most critical component of the LIDAR processing. There are many methods which have been developed for different types of purposes. Based on these methods' processing principles, they fall into the following categories, which are surface-based, region-based, segmentation-based, and slope-based filters. Each category has its own assumptions for the ground and non-ground objects, which results in the advantage and disadvantage of each method. The data structure of all the methods would fall into three types, which are original data points, TIN (Triangulated Irregular Network) [2], and grid.

2.3.1    Surface-based Filter

Surface-based filter methods normally assume the terrain has a spatially continuous surface, accordingly least-squares surface fitting is commonly used in these

methods. Kraus and Pfeifer [21] proposed an interpolation and filter method, which belongs to this kind of filter. The disadvantage of this method is that it is more likely to filter out points at breaklines. Brugelmann [7] proposed an improved method, which detects the breaklines in the filtered data in order to identify the terrain characteristics. In Brovelli et al. [5], they used the Spline method to interpolate the original data, then the residual between the observations and interpolated surface were calculated, and finally the points were classified by comparing the residual with the threshold.

## 2.3.2    Segmentation-based Filter

Segmentation and clustering are widely used techniques for points' classification, and many methods have been implemented based on them to separate discrete LIDAR point clouds. Some researchers defined multiple types of points to categorize the data set. For example, Filin [14] separated the ground features into four types of clustering classifications by using point position, elevation difference from neighboring points, and information of the point to its tangent plane. The clustering algorithm combines these attributes into a 7-tuple vector feature space. The approach proposes a surface class and identifies points associated with the class [14], and then group points according to the neighborhood system with the criteria of a predefined minimal number of points per cluster and accuracy threshold. This process is carried out repeatedly, until no meaningful surfaces can be proposed. Finally, it will extend each cluster until no more points can be added, and then merge clusters with similar attributes. Jacobsen and Lohmann [17] also used the segmentation method to classify data points into several classes to separate terrain points.

Tóvári and Pfeifer [45] proposed a segmentation-based ground filtering method, which has two major steps. A segmentation process is first initialized by a region-growing method from a randomly picked seed point. Then, the method gradually adds neighboring points based on three criteria, which are similarity of normal vectors, distance of candidate point to the adjusting plane, and distance between current point and candidate point [45]. Finally, it separates data points into segments according to surface objects. The second step uses those segments as initial elements, and carries out a least-squares linear interpolation involved with an adaptive weight function, which is used to minimize the weights for segments from non-ground objects. Experiments were done on a small and relatively flat area with two buildings. The results showed that this method overestimated the non-ground surface by showing an upward curve in the building area. This was partly caused by some non-ground objects that were not removed but were involved in the surface interpolation with a smaller weight.

Since many segmentation and cluster-based filtering experiments were tested on relatively flat ground surfaces [14][17][37][45], further experiments on more complicated surfaces with rough terrain are necessary to evaluate the performance, because surfaces with less homogenous height texture could be challenging for these methods [29].

2.3.3   Region-based Filter

Region-based filter methods normally assume the ground objects are relatively lower points while the non-ground objects are relatively higher points.

In Masaharu and Ohtsubo [27], the lowest points of a small region were selected as the ground points, and the elevation difference was calculated between non-ground

points and the mean value of the surrounding selected ground points in a certain range. When the elevation difference was greater than one sigma, the point would be marked as non-ground point.

The mathematical morphological method is a widely used technique in the region-based filter. Mathematical morphology is a theory and technique for the analysis and processing of geometrical structures based on set theory. It is a commonly used method to process digital images, but it can be utilized on graphs, surface meshes and many other spatial structures as well. Mathematical morphology has two fundamental operations, which are *dilation* and *erosion*. These two operations are commonly used to dilate (*dilation*) and reduce (*erosion*) the size of features in binary images. They can also be combined to generate *opening* and *closing* operations. In the *erosion* and *dilation* operations, a predefined shape called a structural element is used to probe and be compared with the image or object. The conclusion can be achieved on how this shape fits or misses in the image or object.

The key process of the *dilation* operation is to use the structural element to probe the image. When the structural element is positioned at a given point and it touches the object, then this point will appear in the results of the transformation. Otherwise, it will not. On the other hand, for the *erosion* operation, when the structural element is positioned at a given point, if it is included in the object, then this point will appear in the results of the transformation. Otherwise, it will not. The following figures show the *dilation* and *erosion* operations.

The Source Image

The anchor pixel

The Structural Element

Dilation Result
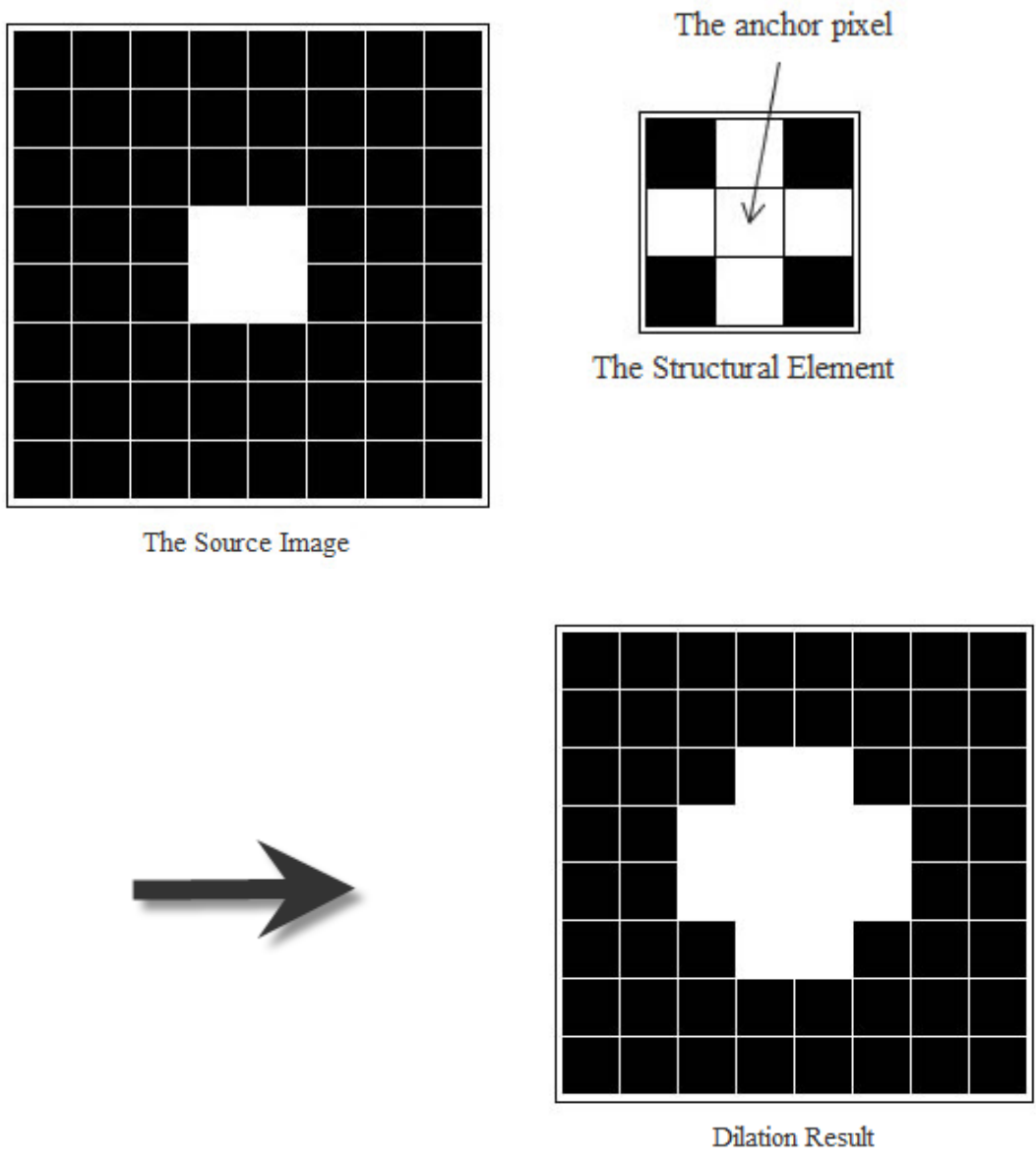
Figure 2-2 Dilation operation

The anchor pixel

The Structural Element

The Source Image

Erosion Result
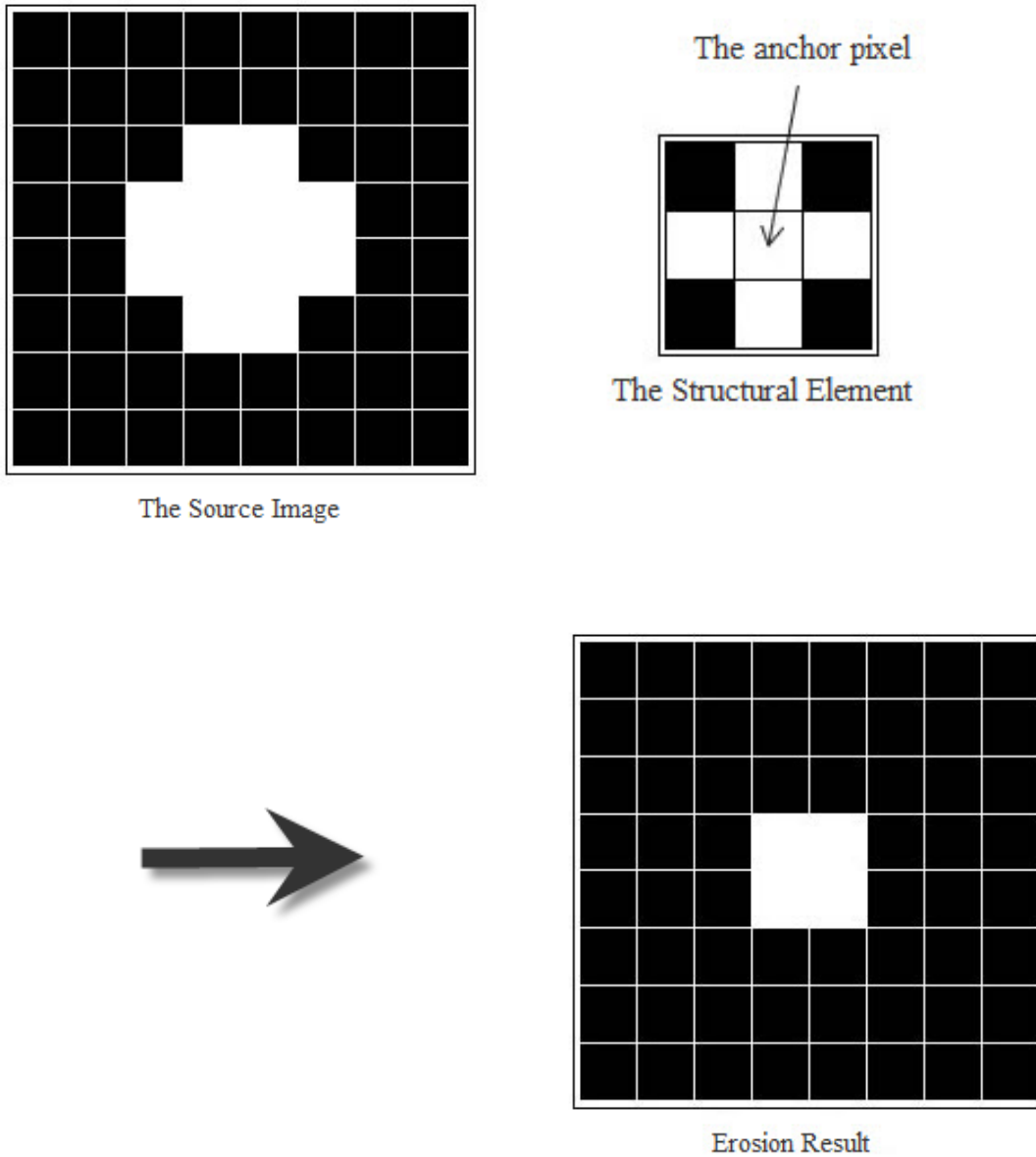
Figure 2-3 Erosion operation

Kilian et al. [18] first proposed to use mathematical morphology to filter non-ground points. The disadvantage of this method is that ground points would be filtered out when the filtering window is too large. Petzold et al. [32] proposed a method which used the moving filter window from large scale to small scale to filter the points

iteratively. This method first created a rough terrain model by finding the lowest points in a relatively large moving window. The points with elevation difference greater than the threshold were filtered out, and a more precise terrain model was achieved. The process was repeatedly carried out along with reducing the window size, until a final terrain model was achieved. The results of this method are affected by the final window size and the final threshold, below which points are expected to be terrain points [32]. A small window size would not remove points on the large buildings and keep them as ground points. A relatively large window size would remove the small local discontinuities and smooth the terrain on a large scale. The elevation difference threshold would affect the results significantly. A high threshold in the final step would classify many vegetation points as ground points, while a small threshold would filter out more small terrain discontinuities. The selection of the parameters depends on the terrain types remarkably. For example, the parameters would be different on the flat and mountainous area.

Morgan and Tempfli [31] utilize the morphological filter as a core part for separating terrain and non-terrain segments. In the proposed method, it first grids the data set with appropriate cell size according to the point density. Then the grid data set is interpolated by using Nearest Neighbor interpolation. After the interpolation, morphological filter was applied to separate terrain and non-terrain points under different sizes of moving windows. The filter used weight function to assign different points with various weights and then filter out non-ground points. The weight of each point depends on the filtering window size and the band width in the window. The band width of a window refers to a certain height range formed by the deepest point and some other points higher than it in the filtering window.

Pixel_weight =

$$\frac{window\_size - min\_window\_size}{max\_window\_size - min\_window\_size} \cdot \frac{band\_width - (point_{ht} - deepest\_point\_ht)}{band\_width} \quad [31]$$

In the equation, the window size can be determined by the building sizes in the data set. The minimum and maximum sizes of the windows are determined by the expected minimum and maximum building size, respectively. These two window sizes are varying for different data set. The band width represents the expected range of terrain elevation in the window. The band width is the expected range of terrain elevation in the window [31]. In order to separate terrain and buildings, the band width has to be smaller than the minimum building height. The weight threshold has to be defined for classifying terrain and non-terrain points.

In Lohmann et al. [26], they used the linear prediction and dual rank filter, which is based on morphological filtering, to eliminate points above ground, and compare the filtering results with the least square surface at the non-ground objects and breaklines.

Wack and Wimmer [47] assumed local lowest points are the ground points and use the Laplacian of Gaussian method to filter out non-ground points with abrupt change of elevation from a large grid to small grid.

Zhang et al. [51] used a moving filter window from small scale to large scale to do the *open* operation of mathematical morphology. The elevation difference after *open* operation was compared with the predefined threshold for each filter window size. The point was filtered out if the elevation difference is over the threshold.

There are several proposed methods [10][11][12] in our framework, which are based on the mathematical morphological method.

2.3.4   TIN-based Filter

When using TIN as the data structure to process LIDAR data, it would have both surface-based and region-based filters' advantages. In Axelsson [2], he assumed that the local lowest points are the ground points and used them as the seed points for the TIN model. By comparing the angle and distance made by each observation point and TIN facet with some thresholds, the point was classified. Furthermore, it introduced the mirror point criteria to check the points at the breaklines. Sohn and Dowman [44] used a similar method, but it took four ground points to form the TIN model. They classified the ground points by Minimum Description Length (MDL) iteratively.

2.3.5   Slope-based Filter

Slope-based filter normally assumes that there are some abrupt elevation difference between ground points and non-ground points, which means there would be a relatively greater slope at the non-ground points. Vosselman [46] utilized the mathematical morphological method and created a kernel function based on terrain slope to filter non-ground points. Sithole [40] improved Vosselman's method by using gradient map to automatically adapt the change of terrain.

2.4  Filtering Methods Comparison and Critical Issues

LIDAR points normally consist of ground points, non-ground points and noise points. The ground points are normally from the bare-earth terrain surface, in which the points are usually the lowest points in a local area. Non-ground points are normally from the objects above the bare-earth terrain, such as vegetation, buildings, and constructions.

The noise points are normally some unexpected or error measurements, such as the noise from LIDAR devices or birds.

2.4.1    Ground Measurement Physical Characteristics

Ground points normally have general physical characteristics, which are commonly used as assumptions in many filtering methods. These general physical characteristics can be divided into the following categories [29]:

1.  Lowest Elevations [29]. Ground points usually have the lowest elevations among their neighboring points in a certain local area. Many filtering methods used this as an assumption to search the initial ground surface points for filtering.

2.  Ground Surface Steepness [29]. Surface slope between two neighboring ground points is normally smaller than between a ground and non-ground point. Therefore, many filtering methods use the surface slope threshold as the criteria to distinguish ground and non-ground points. However, the surface slope threshold values varied significantly on different terrain types. Relatively flat urban surfaces may have a lower threshold, while complex surfaces, such as mountain terrain or high-relief forest canopy surfaces, would have steeper slopes and need a greater threshold to separate ground and non-ground points.

3.  Ground Surface Elevation Difference [29]. Elevation difference between two neighboring ground points is normally smaller than between the ground and non-ground point. Therefore, many filtering methods use the elevation difference threshold as the criteria to distinguish ground and non-ground points.

4. Ground Surface Homogeneity [29]. Ground surfaces are relatively continuous and smooth. Trees and buildings are the major non-ground features that should be removed from the measurements. However, trees are usually less smooth in texture than ground and building surfaces, and the morphological method can be utilized to remove them.

## 2.4.2 Difficulties in LIDAR Ground Filtering

LIDAR ground filtering methods rely on the different assumptions, which are based on the ground characteristics. However, when certain ground measurements are mixed with some specific non-ground objects, it would result in severe difficulties in the filtering. The following features are the major cause of the problems for some filtering methods:

1. Low vegetation, such as shrubs.
2. Complex constructions, such as bridges.
3. Buildings with complex shapes and different sizes.
4. Hill cut-off edges.
5. Mixing of terrain types.
6. Lack of reliable accuracy assessment.

When the low vegetation measurements are mixed with ground surfaces, these low elevation measurements are prone to be identified as ground surfaces. It is difficult to distinguish these low vegetation measurements from variable terrain surface measurements, because the elevation change on the low vegetation measurements is very similar to the terrain variation. The distribution of these low elevation measurements

might help the filter to identify these low objects measurements by some methods, which are based on the physical and distribution characteristics, such as morphological methods. However, some artificial landscapes have vegetation mixed with terrain, which is more difficult for the filtering methods.

Complex constructions, especially complex bridges, are also difficult for the filters. Because there are some ramps and parts of the bridges are connected with ground surfaces smoothly, it would make it very difficult for the filter to separate the objects from ground surfaces.

Many filters are not able to handle hill cut-off edges well, because they normally have a sharp change on the slope or elevation, which is not normal on natural surfaces. In reality, there are some terrain types which have these physical characteristics, such as cliffs, shores and riverbanks [29]. They would make many slope-based filters mislabel the ground measurements as non-ground objects because their slopes are over the threshold.

Large and variable sizes of buildings would be difficult for some filters, which are based on moving windows. This is because when the moving window size is too large, it would remove many ground surface points as well.

## 2.5  Terrain Model Generation

Traditional methods of LIDAR data visualization normally focus on two-and-a-half-dimension visualization. There are several types of commercial software, which provide functionalities to build up 3-D models. The general idea is to combine LIDAR data with terrain photos, such as digital aerial photos, by geo-referencing those images

and picking up the texture of objects. Suddhasheel Ghosh and Bharat Lohani [15]
proposed a framework to automatically generate 3-D models. It utilized some commercial
and free software packages, such as Matlab, GCC, and SGI OpenInventor, to accomplish
the whole task. It first used a utility FACET developed in MATLAB to generate plannar
faces. Then, it used the utility TextureMap, which was developed using GCC, to pick up
the texture from aerial photographs. Third, a utility FACET2IV developed in GCC was
used to generate an inventor file with extension *.iv* based on the two previous steps'
results. Finally, the inventor file was displayed in the SGI OpenInventor.
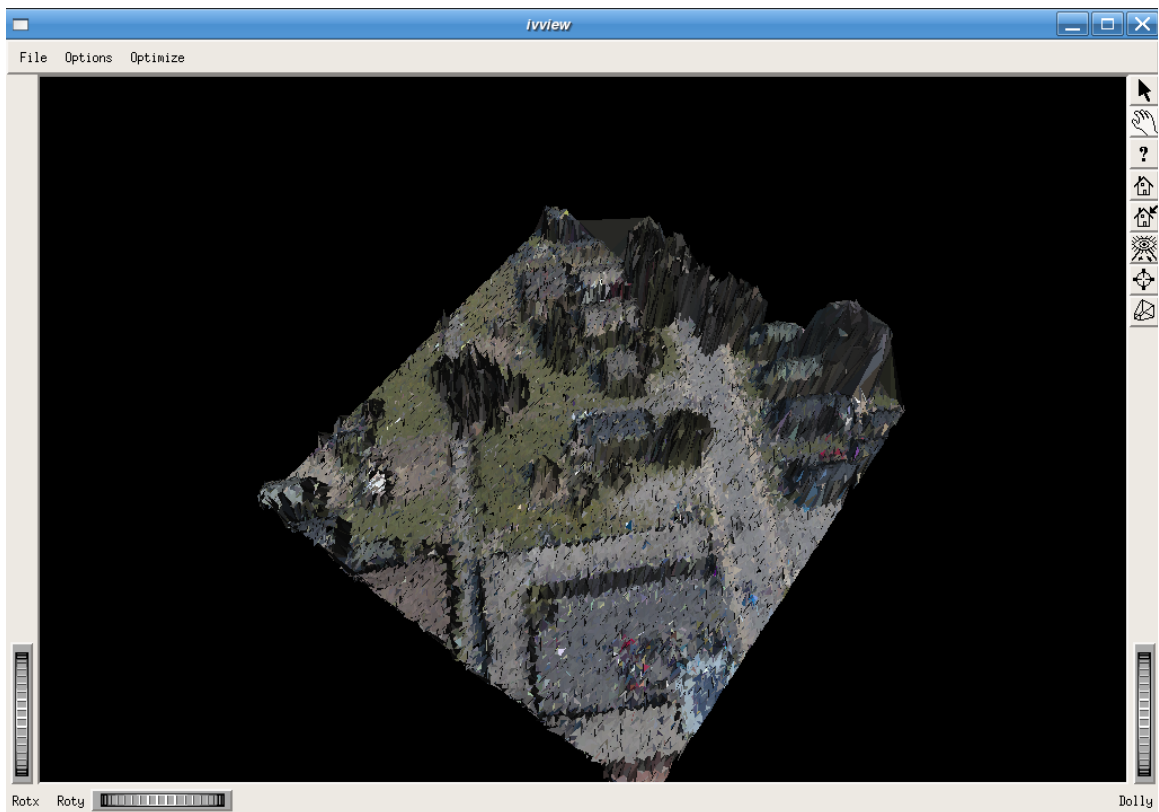


Figure 2-4 3-D model displayed in OpenInventor

## Chapter 3

## OVERVIEW OF THE FRAMEWORK

In this chapter, a general overview of our LIDAR processing framework will be shown. The LIDAR processing framework [9][54][55] proposed in this dissertation works like a sequential product line. Since the core research object is the LIDAR data, all the steps are involved with the data sequentially. From the processing order perspective, we can classify the framework into three major parts or steps. The first one is the data pre-processing step, which includes the data storage, data partition, data sparse, data removal and data interpolation. The second step is the data filtering step, which is the core procedure in the framework. It involves various kinds of filtering methods. The third step is the presentation step of the data, which includes generating 2-dimensional and 3-dimensional images and models. We will overview each step in the following sections.

### 3.1    Resampling Data

The first step of our framework is a data preparation step for later processing according to its purposes. Due to the computer processing ability and the complexity of data sets, the original LIDAR data are too huge to store and process directly. We need some mechanism to organize the data in such a way that we can store them as relatively smaller pieces. In our framework, some data sparse and retrieval methods were dedicated for this purposes. Since there are erroneous and redundant data points in the original survey data, we have to remove these points for better filtering results in the later processing steps. The error data removal procedure is dedicated to this purpose.

Since there were huge amounts of data that contained both useful and non-useful information for the user's requirement, one of the important works in the pre-processing step is to resample the data. In other words, we have to shrink the data and get some representative points in a certain area, or retrieve a certain amount of data from a certain area.

### 3.1.1 Sparse Data

Since the average space of contiguous points in the survey area is very dense, we should select some points to represent a certain block of the area. The size of a certain block of area can be defined according to the application's requirements. Normally, the block of area is in the shape of a rectangle, circle, triangle or polygon. The simplest shape is the rectangle or square grid. We can achieve a sparse data set after this operation.

In order to acquire sparse data, we first have to acquire the boundary of the survey area from the data points' file, and then split the original data into grids. The size of the grid can be decided according to the terrain type of the survey area or the density the user needs. If there is no significant difference of terrain type in the survey area, the size of the grid can be relatively large. Otherwise, it can be relatively small.

More specifically, we can process the original data into sparse data by the following steps below (X, Y and Z are the three-dimensional value of the point):

1) Scan all the data in the source file; acquire the number of the data points in the file.

2) Get the boundary of the survey area by computing the minimum and maximum values of both X and Y among the data, and create a rectangular

boundary. We could also shift the boundary rectangle to some integer value, because it might make it easier and more uniform for the grid operation on the same data set in other processing.

3) Split the whole area into grids with given width and length inside the boundary rectangle, while each point belongs to one grid.

4) Scan each data point, and check which grid it belongs to; choose the one that satisfies certain criteria as the representative point of each grid. For example, we can choose the lowest elevation (Z value) point as the representative point of each grid.

The sparse method is illustrated in Figure 3-1. We can split the whole area into grids. The coordinates of the left-bottom point is made by the minimum X, Y values or user-defined position; the coordinates of the right-top point is made by the maximum X, Y values or user-defined position. The direction of X is horizontal in the figure, and the direction of Y is vertical. We split the area into a number of columns along the X direction, and a number of rows along the Y direction. Thus, each grid in the figure can be indexed according to its index of column and row. The index of a grid can be counted as follows:

Index of Grid = IndexY * Columns + IndexX

IndexY refers to the index of Y's direction, and IndexX refers to the index of X's direction. By using the index of the grid and the X and Y directions, we can scan all the data points, find the grid into which each point falls, and find the representative point among each grid. The points falling in the same grid have the same index of the grid, and the one with the lowest elevation in the grid can be the representative point of that grid.

We can also use other criteria to choose the representative point in a grid, such as the median point. The selection of the representative point relies on the application objective.
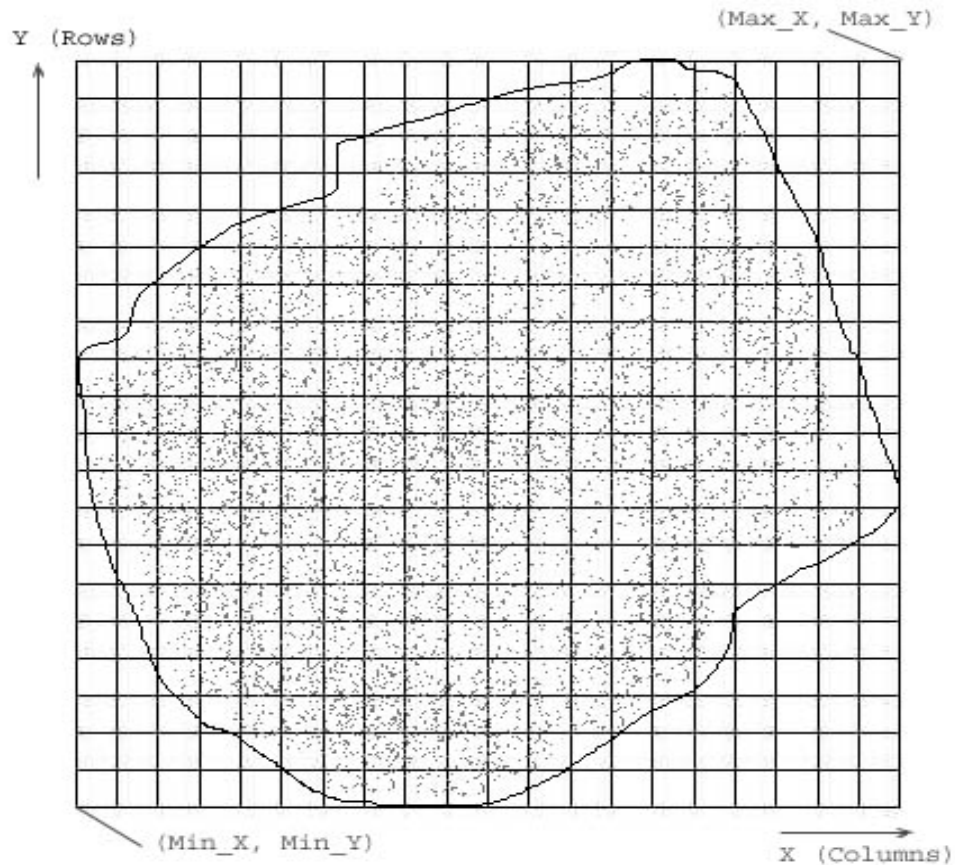


Figure 3-1 Method for sparse data

3.1.2   Data Retrieval

From the users' perspective, they may want to retrieve the data in a certain area to study; therefore effective data retrieval methods would be very helpful. In our framework, two data extraction methods are widely used. One is the grid-based method, and the other is the polygon-based method. The grid-based method is simple and fast, because it normally uses the existing partition grid to retrieve data. As we mentioned in the previous

section, the data set was partitioned into grids with given size of the grid. We can find each grid by its index, and can also extract a rectangle area by its bounding box, which can be decided by the index of the corner grid and the coverage along the X, Y directions. For some applications, it needs to retrieve a certain area of data with some buffer zone. It would be very easy to retrieve the buffer zone based on the indexed grids by collecting data from its neighboring girds.

The other commonly used extraction method is the polygon-based method. A polygon boundary of the data is defined first, and then we can use some appropriate algorithm to retrieve the data in the polygon. There are many algorithms for testing whether or not a point is within a polygon. Many algorithms utilize area computations, and many others work only for convex polygons or polygons without concavities.
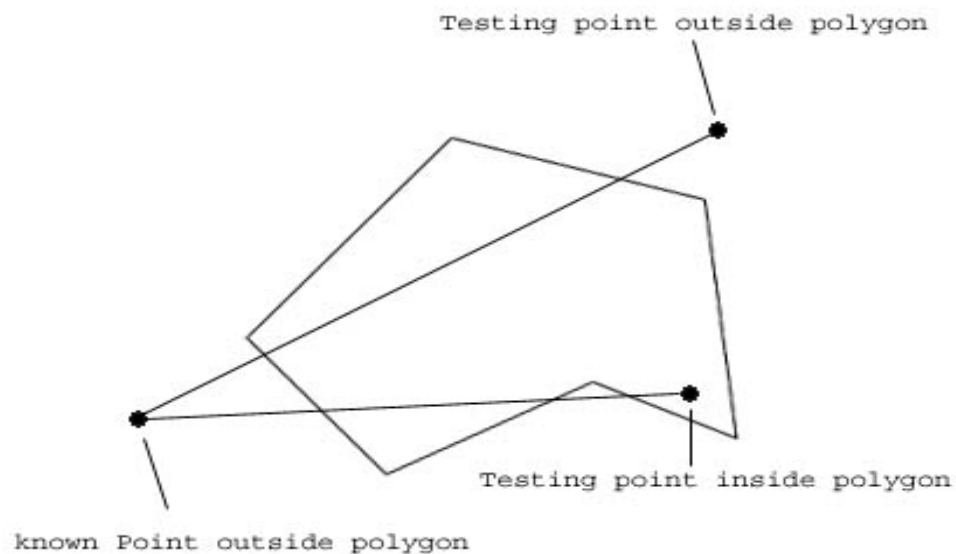


Figure 3-2 Jordan Curve Theorem

34

The simplest algorithm for point-in-polygon testing is the Jordan Curve Theorem [16] (shown in Figure 3-2). This simply states that a line between a point known to be outside a polygon and the testing point will cross the polygon boundary an even number of times if the testing point is outside the polygon; and an odd number of times if the testing point is inside the polygon.

This theorem provides solutions in all cases, except when the lines either touch a vertex or run parallel to an edge. The parallel problem is often significant. Because the outside points can simply be chosen as vertically above or below any given test point, many map lines run parallel to the axes.

Thus, in these cases, we can process the polygon retrieval according to the following steps:

1) Scan the vertex of the polygon, and acquire the number of vertexes.

2) Scan the original data, and acquire the amount of the data.

3) Get the boundary of the survey area by computing the minimum and maximum X, Y value among all the data. Create a reference point outside the boundary.

4) Check each point in the data file, and state whether or not it is inside the polygon created by the vertex loaded in Step 1 by using the Jordan Curve Theorem. If it falls into the polygon, add it into the results.

With the polygon extraction method, we can easily retrieve the data points in a certain boundary the user would like to study.

### 3.1.3   Error Points Removal

During LIDAR data collection, there might be some abnormal points collected. For example, if the laser is shooting on a bird's body, there would be a very high point collected. Sometimes, the laser would penetrate the window on the top of the roof and reach the floor in the building. Accordingly, it would have a very low point at the roof position. These abnormal points are normally outliers and have side effects on the later data processing, because they would introduce error terrain or non-terrain information and make mistakes in the results for most of the methods. Therefore, we have to use some methods to remove these points. In our framework, the mathematical morphology is used to filter out these outliers, and the detail of the method is discussed in the later chapters.

### 3.1.4   Data Interpolation

In our framework, the data set is normally split into rectangular tiles for further processing. By this way, we can partition a large survey area into smaller pieces to process, and they can be processed simultaneously. There might be some empty areas at the boundary or inside of each tile of data. We have to interpolate these empty areas in order to get a complete terrain surface to process. There are many interpolation methods available. From the effectiveness and efficiency perspectives, we would normally use the nearest neighbor method for grid-based data, because it is the simplest and fastest method for evenly distributed data. The Kriging method [25] is another good choice in our framework for generating surface or shade relief maps. Also, a priority interpolation was proposed in this dissertation, and the details of the method will be discussed in the later

chapters. Some commercial software, such as Surfer and ArcGIS, also offer good interpolation tools, which are used in our framework.

Among all the tools, the gridding tool in Surfer was frequently used for the interpolation purpose. It offers many interpolation options and parameters, from which users can choose. Figure 3-3 shows the interface of the Surfer 9 gridding tool. There are many gridding methods from which to choose. The gridding results can further generate 2-D or 3-D maps.
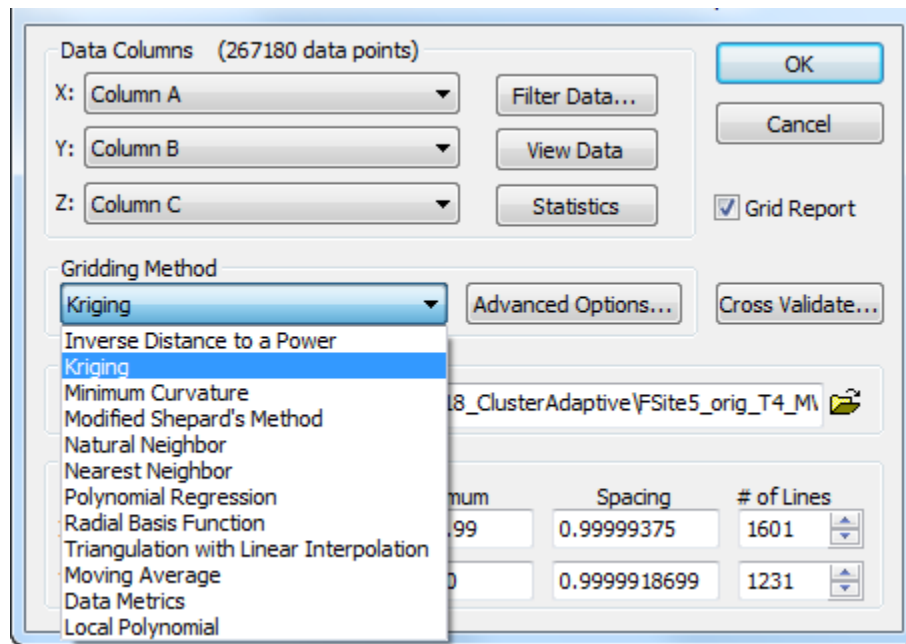


Figure 3-3 Surfer grid tool

## 3.2 Filtering Methods

After the first-step processing, the original data set is partitioned into tiles with proper size. Outliers are removed, and the empty regions are interpolated according to

further processing needs. Then, we can start the core processing of our framework, which involves all kinds of filtering methods.

There are so many filters that have been developed and studied in recent years. They are normally within the following categories:

- Segmentation- and Cluster-based Filters

- Morphological Filters

- Directional Scanning Filters

- Contour-Based Filters

- TIN-Based Filters

Every filter in each category has its pros and cons. We will discuss them in the later chapters. In this dissertation, we focused on morphological filters, and developed more algorithms on them.

### 3.2.1 Progressive Mathematical Morphological Filter

Zhang et al. [51] proposed a method to remove non-ground objects using a progressive morphological filter. This method is based on mathematical morphological filtering. In their method, the input LIDAR data set was gridded into mesh for 1-D or 2-D morphological filtering. An increasing window size was used during each morphological filtering step. The threshold of elevation difference in each filtering step was a variable, which is suitable for removing different sizes of non-ground objects. It is a very efficient and effective method. As long as the window size is greater than the non-ground objects, and there are enough ground objects with the window size as the reference points, those non-ground objects would be filtered out with the rational elevation difference threshold.

The algorithm was implemented in our LIDAR processing software [9] as one of the filtering tools.

### 3.2.2   Cluster-based Progressive Morphological Filter

Although the progressive morphological filter proposed by Zhang et al. [51] was a very efficient and effective filtering method, and it has very good performance on the flat terrain with all kinds of non-ground objects, it would make significant errors along with an increasing filtering window size on undulating terrain, such as mountains and sand dunes. In order to solve these problems, a cluster-based morphological filter was proposed in this dissertation, which would use a cluster mechanism to check filtered points and improve the progressive morphological filter when dealing with such terrain type mentioned above. The details of this method will be discussed in the later chapters.

### 3.2.3   Trend-based Adaptive Morphological Filter

In the progressive morphological filtering method proposed by Zhang et al. [51], input LIDAR data set was gridded into mesh for 1-D or 2-D morphological filtering. An increasing windows size was used during each morphological filtering step. The threshold of elevation difference in each filtering step was variable, which is suitable for removing different sizes of non-ground objects. It is a very efficient and effective method. As long as the window size is greater than the non-ground objects, and there are enough ground objects with the window size as reference points, those non-ground objects would be filtered out with a rational elevation difference threshold. However, the parameters of the filtering method are fixed under each filtering window size. This would work without

significant error when the terrain is relatively flat, but it would not work very well in an undulating terrain type, such as a mountainous area or on sand dunes. Therefore, we need some method to adaptively select the filtering parameters, such as filtering thresholds.

A new mechanism of selecting morphological thresholds is proposed in this dissertation, which would make the filter more adaptive and automatic. This mechanism, called the trend analysis method, was introduced in the filtering procedure. Combined with the cluster analysis method, this method will help the filtering procedure automatically choose the threshold based on the analysis of local terrain characteristics during the progressive morphological filtering and make the filtering adaptive to different kinds of terrain types.

The essentials of this adaptive filtering method include two major analysis methods for estimating the filtering threshold for each point. One is the cluster analysis method, and the other is the trend analysis method. The cluster analysis method is used to build up some collections of point clusters for each row or column of the grid data; while the trend analysis method is used to analyze the local surface trend for each row or column. If combining these two analysis methods, a threshold for each point under a certain window size can be calculated. Therefore, it would fulfill the adaptive filtering, which means it is not required for the filter users to have much experience and knowledge on the filtering method. Also, users do not need a lot of familiarity with the surveyed area.

3.3    Terrain Model Generation

Terrain model generation is the processing step after filtering. This processing step is application oriented, because the model which would be created is based on the different application requirements. All the processing methods generally belong to the categories of object footprint detection, object separation, object recognition, contour map generation, and three-dimensional model generation.

3.3.1    Object Footprint Detection

After the filtering process, we can acquire two sets of data: one is ground point data, and the other is non-ground point data. In terms of different applications, we can either retrieve the bare-earth ground points to generate a terrain model, or extract the non-ground objects to generate a 2-D or 3-D objects' model.
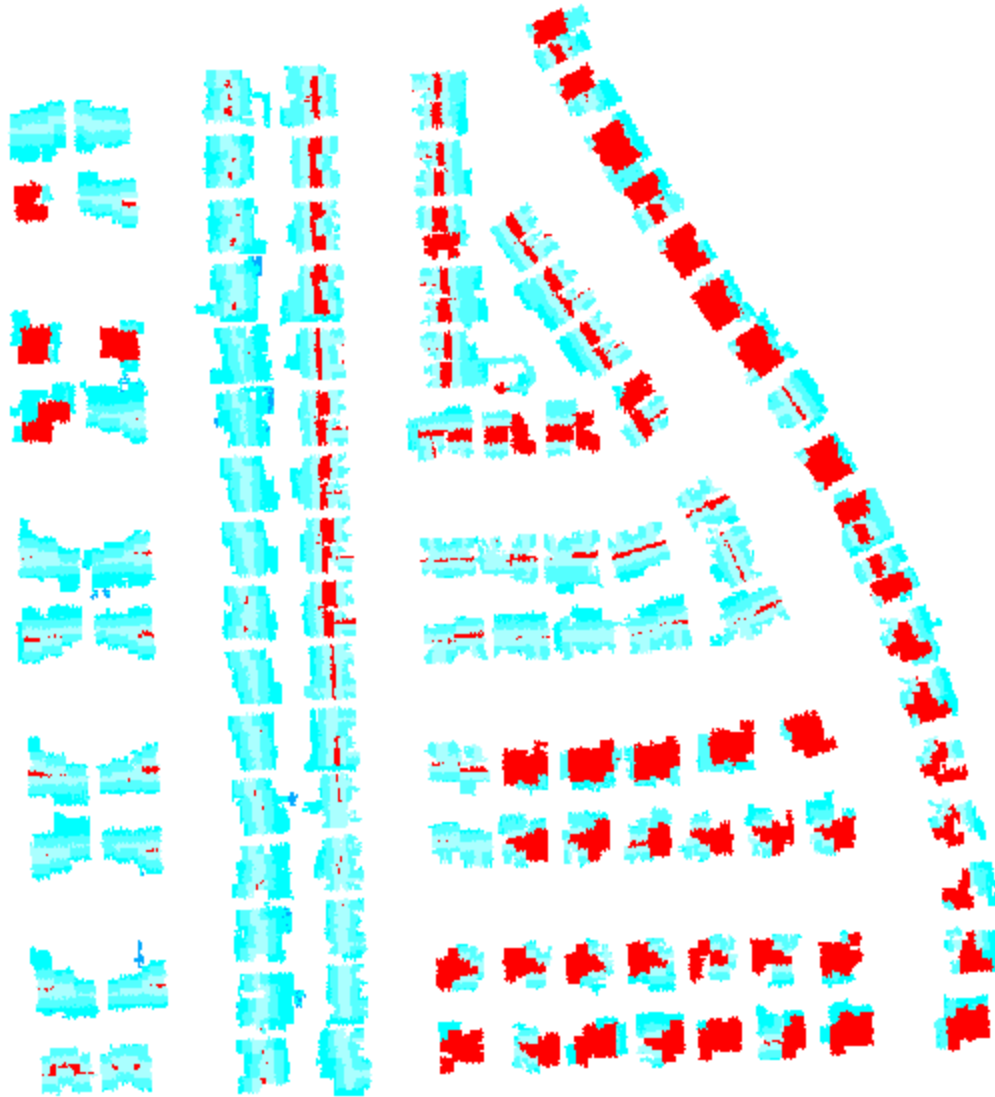
Figure 3-4 Non-ground object points acquired from filtering process

Figure 3-4 shows the non-ground object points acquired from the filtering process. To study the non-ground objects for many applications, the shape of the objects is the fundamental information for research. In terms of 2-D shape, we have to find the boundary of the object, which is called the object's footprint. In our framework, data is partitioned into grids, and most of methods are normally grid based. Grid-based data is much easier for the footprint detection of objects. The general idea of grid-based object

footprint detection is illustrated in the following procedure: First, we start collecting an object's points from any object point grid, and then expand the object's points set by checking the connectivity of its neighbor girds. This procedure is to check whether the object point has any connecting neighbor points. If it has some neighbor grid points, they are collected into the current object's points set. By expanding the object's points set in this way, we can finally reach all the point grids of this individual object. For different terrain and object types, we can use a different neighbor grid strategy for expanding the grids, which can be eight neighbor grids or four neighbor grids. Eight neighbor girds means the eight surrounding grids, and four neighbor grids means only the top, bottom, left and right neighboring grids. After expanding the connecting grids, we can get every individual object's points set. Then, we can detect the boundary of each individual object by continuing to search along its boundary. Finally, we can acquire the footprints of all the objects which are shown in Figure 3-5.

Figure 3-5 Footprints of non-ground objects

After we get the footprints of non-ground objects, this information can be stored and displayed in multiple ways. One useful way is to store the footprints in an ArcGIS shape file. In our framework, we implemented the conversion of the footprints' flat text file to an ArcGIS shape file. The polygon shape file in ArcGIS has several advantages. First, it not only stores the geometry shapes and point values, but also generates the index

file for quick searching. Second, it can be easily displayed and overlapped with other shape files, and images for better analysis. Third, it would be very easy to carry out geo-spatial queries and operations. Figure 3-6 shows how the polygon shape file was displayed in ArcMap.
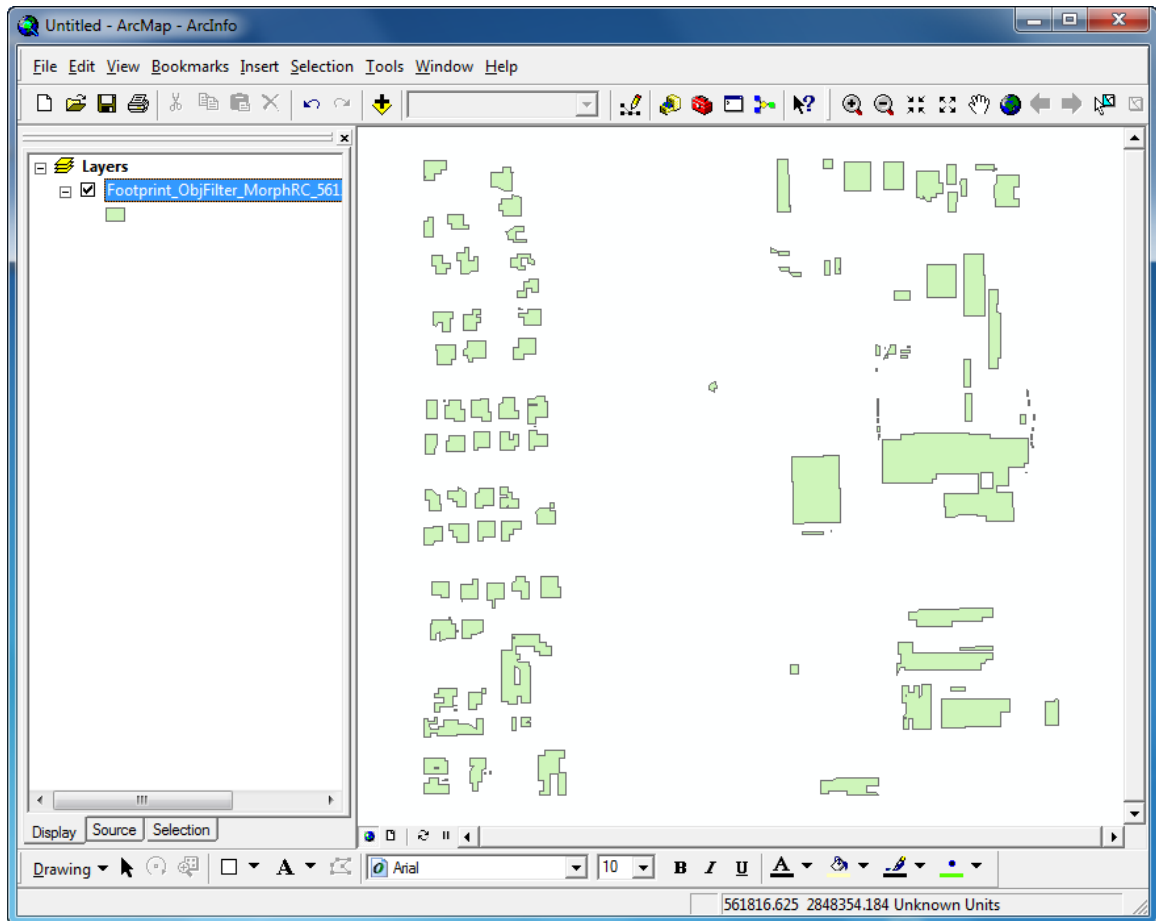


Figure 3-6 Polygon shape file displayed in ArcGIS

3.3.2　Object Separation

Since we can only separate ground and non-ground objects during the filtering process, the non-ground objects data set is a combination of all kind of objects filtered out. Different applications would need different kinds of objects, according to their research target; thus object separation is essential to many applications. Non-ground objects can normally be separated by their physical shapes. In terms of two-dimensional grid representations of LIDAR data, they can be separated by the coverage area and distribution. For example, to separate construction and vegetation, we can detect the objects' coverage area and distribution of their covered grids. Because constructions are normally larger than individual vegetation, and have more regular coverage shape, we can separate them by these two aspects. However, in the urban area, the buildings are usually mixed with vegetation. It would be more difficult to separate them with expanding size of the objects, because the buildings and vegetation grids would be connected and mixed together. To solve this problem, Zhang et al. [53] proposed a solution, which uses the region-growing algorithm based on a plane-fitting technique. The general idea of this method is as follows. After the connected region formed by the building and vegetation objects were identified, and then the region-growing procedure starts from an inside point. A best-fitting plane for this inside point and its eight neighbor grids is calculated by using the least squares method. A plane in a three-dimensional space is defined by $z = ax + by + c$. The parameters (a, b, c) of the best-fitting plane can be calculated by minimizing the sum of squares according to deviations (SSD) as the following equation [53]:

46

$$\min(\text{SSD}) = \min \sum_{(pk) \in M} (z_k - h_k)^2$$

where M is a set for the current point and its neighbors, and zk and hk are observed and plane-fitted surface elevations, respectively. The region-growing segmentation method would be carried out in such a way that it starts from the seed point of connected non-ground objects. The connected non-ground object points would be sorted in ascending order according to the SSD. The first seed point is the point with the minimum SSD. The neighbors of a seed point is judged by whether it can be collected into the region-growing set based on the plane-fitting technique and predefined threshold hT. If the elevation from the candidate point to this plane is less than the predefined threshold, it would be collected into the current region-growing set. This process is continued until no more points can be collected into the current region-growing set. Then we can start this procedure from those unchecked points in ascending order of their SSDs and repeat the region-growing procedure until all points are segmented.

### 3.3.3 Object Recognition

After the non-ground objects are segmented, individual objects need to be separated in the following steps. First, patches with a small area are compared with the predefined threshold to separate small coverage objects, such as patches of vegetation. Then, the small patches, such as chimneys, water tanks, and pipelines of buildings removed in the previous step, would be recovered based on whether they are completely surrounded by large patches. Third, isolated boundary points, which refer to points

without any inside neighboring points, would be removed. Finally, the remaining patches would be merged in terms of their connectivity.

After the building objects are identified, a raw footprint can be derived by connecting the boundary points. Since there would be too many details and too much zigzag noise along the boundary of the object, we need some way to reduce the noise of the footprint and make its shape more regular. There are several methods that have been developed to reduce the vertices of a noisy raw building footprint. Among them, the Douglas-Peucker algorithm [13] is a simple and effective one. The general idea of this algorithm is that it generalizes lines by forming a line connecting start and end points, and then recursively selects a left point with a largest distance to the line until a predefined distance threshold is reached. The Douglas-Peucker algorithm was implemented in our framework.

### 3.3.4    2-D/3-D Map Generation

For various analysis and visualization purposes, the LIDAR data can be shown in different 2-dimensional or 3-dimensional maps, such as contour maps, shade relief maps, surface maps, etc. There are many commercial software packages that support contour map generation. ArcGIS and Surfer are very powerful software for that. We can use them to draw colorful 2-D or 3-D maps with many customized styles, which would be very helpful for application users to analyze and visualize the data. Figure 3-7 shows a 3-D surface map of original data.
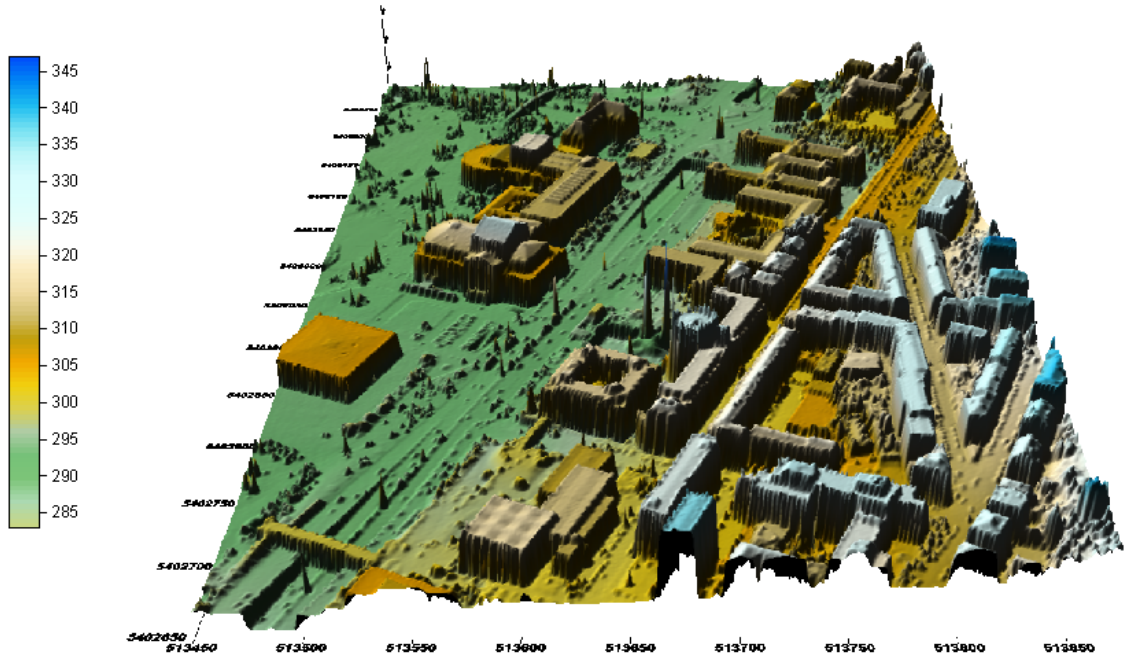
Figure 3-7 3-D surface map

After the LIDAR data was filtered according to different application requirements, variant style maps can be generated with the bare-earth or non-ground objects data. Figure 3-8 shows a shade relief map of the filtered data set.
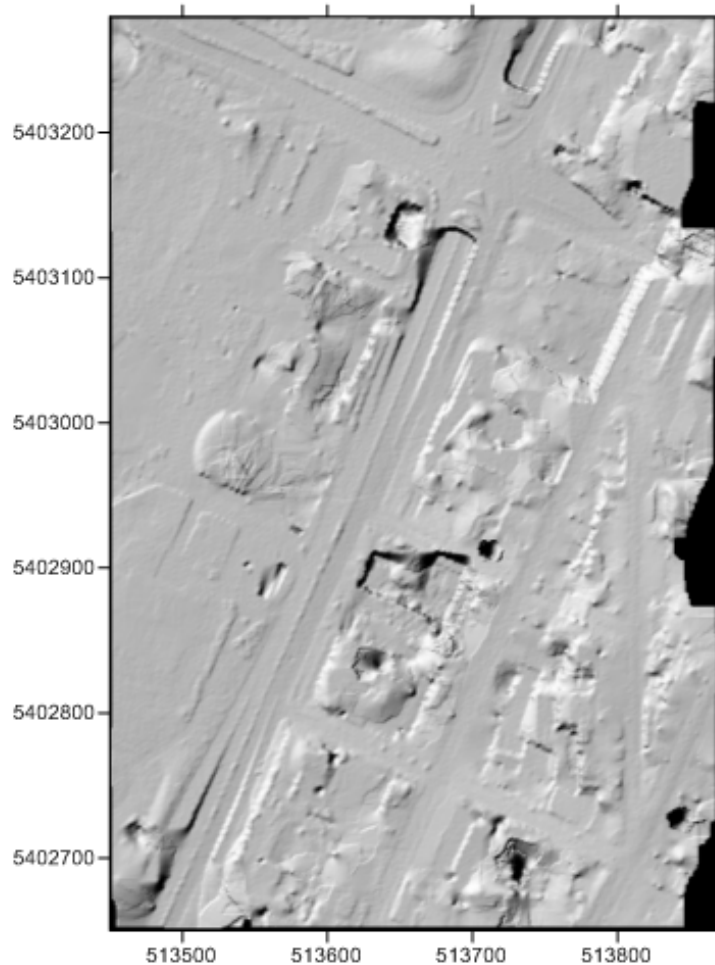
Figure 3-8 Shade relief map

3.4    Summary

In this chapter, the common components and methods of LIDAR filtering in our framework have been discussed and reviewed. It involves the data pre-processing, filtering and model-generation steps, which gives a general picture of LIDAR processing. In the later chapters, we will focus on some detailed issues of LIDAR filtering, such as interpolation methods, filtering methods, and parameters' selection of the methods.

# Chapter 4

## OUTLIER REMOVAL AND DATA INTERPOLATION

In our LIDAR processing framework, the error point removal and data interpolation step is very critical to the filtering step, because it is a preparation of data for different kinds of filters. This processing step would affect filtering results significantly. In this chapter, several outlier removal and interpolation methods used in our framework are discussed in detail.

## 4.1    Outlier Removal

During LIDAR data collection, there are some abnormal points collected, such as extremely high points caused by the laser shooting at power lines, or a bird's body, as well as extremely low points caused by the laser penetrating the building's window and reaching the floor in the building. These points are all error points for our processing, and are normally treated as outliers. Since these outliers would bring more noise and errors in our filtering processing, we have to remove them for a certain purpose.

## 4.1.1    Morphological Outlier Removal

The outliers are normally extremely high or low elevation points and sparsely distributed in the collected data. Since these unwanted points would affect the filtering results, we had better remove them first. Mathematical morphology is a good choice to remove these points. Since these outliers have significant elevation difference with nearby points, we can use a small window size (e.g. 1 to 3 unit grid sizes) to filter out these outliers. When filtering out extremely high outliers, we can use *open* operation,

which refers to *erosion* and then *dilation* operations, and when filtering out extremely low outliers, we can use *close* operation, which refers to *dilation* and then *erosion* operations. Experiments show that the outliers can be effectively removed by using these methods.

## 4.2    Interpolation Methods

Since many filter methods are grid-based methods, the data has to be partitioned and stored in the grid structure. The grids that covered the whole data set could have some empty ones that no data points fall into. For many grid-based filtering methods, these empty grids need to be interpolated. There are several interpolation methods that are widely used in our framework. They will be discussed in the following sections.

### 4.2.1    Grid-Based Nearest Neighbor Interpolation

The Nearest Neighbor Interpolation method is one of the simplest interpolation methods. It just uses the closest point's elevation value to interpolate the empty grid, and has relatively less computation time. In our framework, most of the methods are grid based. It would make most of the methods easier to implement and faster to execute. When using a square grid in the nearest neighbor interpolation, we can save tremendous computation time on calculating distance, while searching for the nearest neighbor according to the grid. The algorithm of this interpolation is simple and straightforward, because we just need to search for the nearest neighbor along each surrounding layer's grids from the inside to outside. If multiple points are found in the same surrounding layer, we could add some other criteria to choose one for the interpolation. Figure 4-1

52

shows the searching procedure of this interpolation. A search for the nearest neighbor point will be carried out along the grids in each layer from the inside to outside. If the nearest neighbor point in one layer was found, we need to check one more outer layer, because there might be a nearest neighbor point existing in the adjacent outer layer, but it would not exist outside of this outer layer.
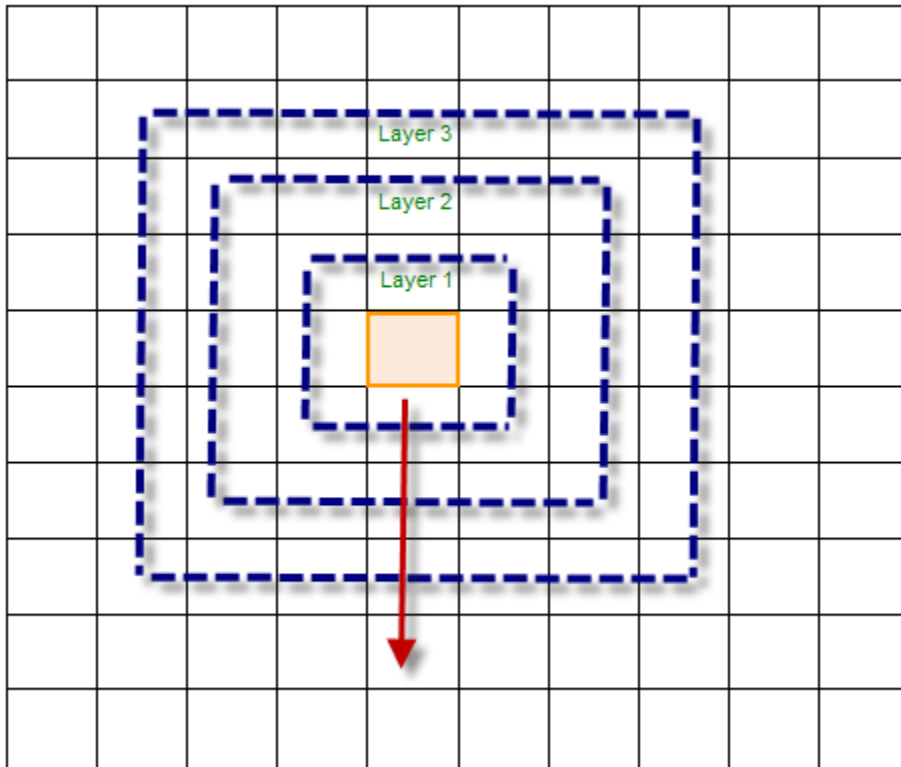


Figure 4-1 Grid-based nearest neighbor interpolation

## 4.2.2 TIN-Based Interpolation

TIN (Triangulated Irregular Network)-based interpolation is a widely used interpolation method, which is normally based on Delauney Triangulation [4]. Delauney Triangulation is a kind of triangulation of the convex hull of points in the diagram, in which every circumcircle of a triangle is an empty circle. Delauney Triangulation can be

used to generate the TIN model, which represents a surface by contiguous, non-overlapping triangles (shown in Figure 4-2). Delaunay Triangulation has several advantages over other triangulation methods, as shown below:

- A convex equilateral formed by two adjacent triangles has a greater minimum internal angle, thus reducing potential numerical precision problems created by long, skinny triangles.

- It ensures that any point on the surface is as close as possible to a node.

- The triangulation is unique and independent of the order in which the points are processed.
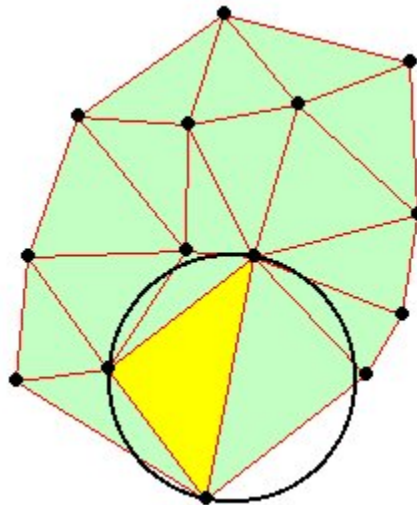


Figure 4-2 Delaunay triangulation

By adding each data point, the triangulation procedure would generate a set of triangles, which are satisfied with the geometry properties of Delauney Triangulation. A TIN model is formed by these triangles' facets. Any point can be added later into the TIN model and form a new TIN. The triangulation procedure will check which of the triangles' circumcircle the new added point would fall into, and those triangles would be destroyed

54

and new triangles would be formed with the new point and those triangles' edges. The triangulation facets made the TIN model a good way to estimate the elevation of any points on a terrain surface, which can be used to interpolate points as well.

### 4.2.3 Kriging Interpolation

Kriging interpolation [25] is a group of geostatistical methods used to interpolate the value of a random field. kriging is mathematically closely related to regression analysis; it uses variogram to express the spatial variation, and it minimizes the error of predicted values, which are estimated by spatial distribution of the predicted values. There are many kriging methods that are used nowadays. Many of them are commonly used in a variety of disciplines, such as environmental science, remote sensing, hydrogeology, mining, and natural resources. The Classical methods of kriging are simple kriging, ordinary kriging, and universal kriging. They all have their own assumption as follows:

- Simple kriging assumes a known constant trend: $\mu(x) = 0$.
- Ordinary kriging assumes an unknown constant trend: $\mu(x) = \mu$.
- Universal kriging assumes a general polynomial trend model, such as the linear trend model.

$$\mu(x) = \sum_{k=0}^{p} \beta_k f_k(x)$$

Kriging interpolation is commonly used in our framework for generating 2-D/3-D models. For example, many types of 2-D/3-D map model generation need the data to be interpolated first. Kriging interpolation often offers very good results for map generation.

4.2.4    Grid-Based Priority Interpolation

In our framework, many filter methods are grid-based methods. Some special interpolation methods would help the filtering methods and benefit from their advantages. A grid-based interpolation method, called priority boundary interpolation [11], was proposed to interpolate empty grids, which would help improve the morphological filters. Since the morphological filters use nearby grid points within the filtering window to carry out the filtering operations (e.g. *open* and *close* operations), the interpolation of empty grids closed to the non-ground objects would affect the results significantly. If these empty grids were interpolated by the nearby ground points rather than objects points, it would make the filter easier to remove the non-ground objects. Otherwise, if these empty grids were interpolated by the non-ground points, it would make the non-ground points' area bigger and require a larger filtering window size to filter out these non-ground points. Especially when we use the morphological filter repeatedly on the results generated by it, it would create more empty grids around non-ground points. Therefore, how to interpolate them to serve the next round filtering would be critical to the filter results. Due to this reason, we need to find a solution to interpolate these empty grids with some specific purpose. A prioritized interpolation method was introduced for this purpose. Figure 4-3 shows why we need the prioritized interpolation method to filter out the high elevation objects in the red rectangle box. The red dots in the box represent the high elevation objects from previous filtering results. When doing the interpolation of the empty grids around the high elevation objects, we had better use low elevation points, which are in blue dots, to interpolate the empty grids toward high elevation objects. In

this way, we can make the interpolation from low boundary points toward the objects, which would benefit from filtering out the high elevation objects.
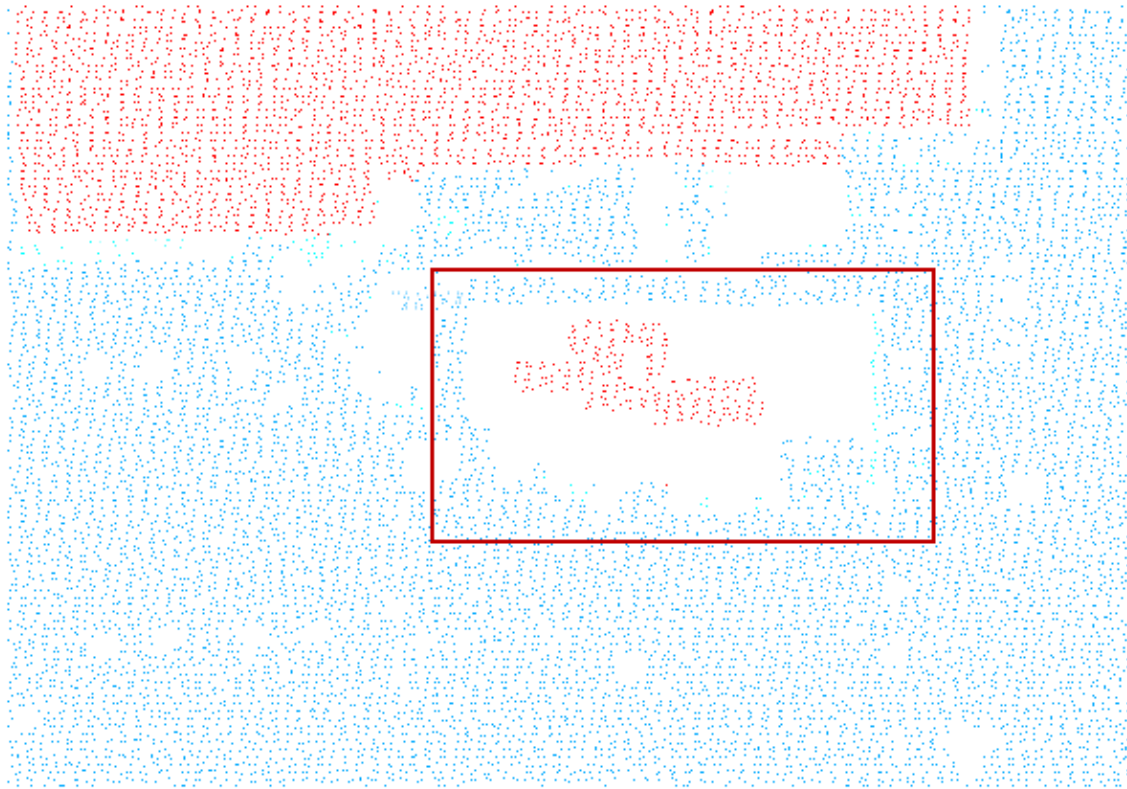


Figure 4-3 Priority boundary interpolation sample data

The essential idea of this priority boundary interpolation method is as follows. First, the empty grids would be identified as separate grid sets by expanding connected empty grids. Second, after these connecting empty grid sets are found, each of these empty grid sets will be interpolated, one after another. For each empty grid set, all boundary grids can be detected. The boundary grids can be separated as outside boundary grids and inside boundary grids, according to the spatial relationship with their connecting empty grids. Then, the empty grid set can be interpolated according to the defined priority rule. The priority rule defines in what order the interpolation will be

carried out. The priority rule reflects the purposes of the interpolation for the specific filtering methods. For example, if the goal is to interpolate empty grids as low as possible, which is close to the elevation of nearby ground points, the boundary points can be stored in an ordered queue according to the ascending elevation, and the empty grids can be interpolated along this order. The newly interpolated empty grids are treated as the new boundary points, because they form the new boundary of the empty grids. Each interpolated point is assigned a priority value based on its elevation, and inserted into the priority queue. The used boundary points are removed from the priority queue, and are not used in the future interpolation. By this means, the boundary points of current empty grid sets can be maintained in the priority queue until all the empty grids are interpolated.

Since the proposed filter is based on the progressive morphological filter [51], the input data has to be gridded with a certain grid size and interpolated for the empty grids with some interpolation method. Each grid chooses a representative point from the points within the grid, and the filter treats each grid as one point. The filtering status of all the grids can be represented by a mark matrix. Each unfiltered point grid's status will be initialized with unfiltered point's mark ("0" or "-7777") and stored in the mark matrix. "0" means real, unfiltered points from the data set, while "-7777" refers to the interpolated unfiltered points [11].

Nearest neighbor interpolation was used in the first pass morphological filtering. The proposed priority boundary interpolation will be used after the first pass filtering. The second pass of morphological filtering is also based on the progressive morphological filter. During each step of progressive morphological filtering, a moving window will be used to filter non-ground objects. The point will be classified as a non-

ground point and marked with the current filtering window size value. A mark matrix is used to record the filtering status for data grids. Each grid point's filtering label will be updated and stored in the mark matrix during filtering.

The initial mark matrix contains only two kinds of values, which are either "0" or NULL mark value ("-9999"), which represent the empty grid. After interpolation, the empty grids are interpolated by a certain method, and the mark values are set as the interpolation value ("-7777"). If the initial mark is not changed until the end of the filtering process, the point with "0" or interpolation mark value indicates the point was classified as a ground point.

The algorithm description of this proposed priority boundary interpolation is as follows:

Algorithm description: *PQBoundInterp*
INPUT:

1. An array of input data points: *pts*
2. Minimum x, y coordinates: *min_x, min_y*
3. Grid size: *cellsize*
4. Minimum connected empty girds: *minConn*
   OUTPUT:

An interpolated array of points: *interpPts*

1.  grdPts ← GridPts(*pts*, *min_x*, *min_y*, *cellsize*) // Grid data set
2.  Initialize *mark* matrix with "0" or "-9999" marks
3.  *connMark* ← *findEmpConn*(*mark*);
4.  for i = 1 → size(*connMark*)
5.      [row col] ← find (*connMark* == i); // Find the index of all the empty grids in set i
6.      If size([row col]) < *minConn*, then continue;
7.      edgeIndex ← findBoundGrid(mark, [row col]);
8.      pq ← CreatePQ(edgeIndex); // Insert each edge grid into a priority queue based on the z value, lower z value has higher priority
9.      while the pq is not empty
10.         edge ← PopPQ(pq); // Pop the first element in the priority queue
11.         NBGrids ← *interpolateNB*(edge); // Interpolate its neighboring empty grids with connecting boundary grids points

59

12.          PushPQ(NBGrids, pq); // Insert interpolated empty grids into the boundary
             grids priority queue
13.      end while
14.  end of for loop of all steps


Algorithm description: *findEmpConn*

INPUT:

Mark matrix: *mark*

OUTPUT:

Mark matrix for empty grids sets: *connMark*

1.  Initialize *connMark* matrix with Non_Empty marks "0" // *connMark* matrix has the
    same size as *mark* matrix.
2.  setID = 0; // Initialize the empty grids set ID
3.  (rows, cols) ← size(*mark*);
4.  for i = 1 → rows
5.      for j = 1 → cols
6.          if (*mark*(i,j) == -9999 && *connMark*(i,j) == 0)
7.             setID ← setID+1;
8.            ptsQueue ← enque(pts(i,j));
9.            while (ptsQueue is not empty)
10.               pts(i,j)← deque(ptsQueue);
11.               *connMark*(i,j) = setID;
12.               for pts(m,n) in 8 neighboring grids of pts(i,j)
13.                  if (*mark*(m,n) == -9999 && *connMark*(m,n) == 0)
14.                     ptsQueue ← enque(pts(m,n));
15.                  end if
16.               end for
17.            end while
18.         end if
19.     end for j
20.  end for i


As in the algorithm description of *PQBoundInterp*, the input data will be gridded

first with a predefined minimum x, y coordinates and cell size (Line 1). The mark matrix

is initialized with "0" or "-9999" marks. "0" represents the real data point, while "-9999"

refers to the empty grids. In Line 3, *findEmpConn* procedure will find all the empty grids

and separate them into a set of non-overlapping connected grid sets. As long as one

empty grid has at least one empty neighbor (either in a four or an eight neighborhood

area), the current empty grid set will be expanded until no more connecting empty grids can be found. When checking connecting grids, four or eight neighboring grids can be searched. The searching directions are shown in Figure 4-4. In our own experience, searching four neighboring grids could sometimes avoid the inclusion of some unrelated empty grids in the expanding process. The *findEmpConn* procedure uses a *connMark* matrix to label each grid's property. The *connMark* matrix has the same dimensions as that of *mark* matrix. All the grids are initially labeled as "0" (meaning unprocessed grids) in *connMark* matrix, while each empty grid will later be labeled with its set ID.

|   | X |   |
|---|---|---|
| X | O | X |
|   | X |   |

| X | X | X |
|---|---|---|
| X | O | X |
| X | X | X |
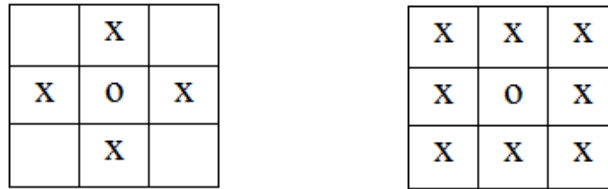
Figure 4-4 Four neighboring grids vs eight neighboring grids

All the connected empty grids will be collected into the same set and assigned a unique ID number. After this empty grid set searching step, each empty grid will be assigned a positive group ID number and stored in the mark matrix *connMark*. Grids with the same ID number form an empty grid set. The non-empty grids in the *connMark* will still have their initial value "0". Each empty grid set can be easily extracted by its set ID number. In Line 6, the total number of empty grids in each set is compared with a predefined threshold *minConn* to determine whether this empty grids set needs to be interpolated. User can use this threshold to eliminate those empty grid sets that are too small. If the *minConn* is 0, it will interpolate all the empty grid sets. Only the sets in which the number of connected empty grids is great than the minimum threshold will be

interpolated. This would save computing time for interpolating small empty grid sets which are formed by filtering small non-ground objects in the first pass. These areas are not the targets for filtering in the second pass.

In Line 11, *interpolateNB* is the interpolation procedure, which can use a different formula to calculate the interpolation value. The average elevation of neighboring edge points is a good choice for the interpolation value.

The procedure of the interpolation is illustrated in Figure 4-5. The orange grids are the initial outside-boundary grids, while the white grids are empty grids. The blue grid is the lowest point grid in the initial boundary grids. The red grids are the non-ground object remains with high elevation, which are surrounded by the empty grids. The green grids are the empty grids adjacent to the lowest grid, which is blue. Starting from the blue grid, the two adjacent empty grids in green can be interpolated; then, the blue grid will be removed from the priority queue and the two interpolated grids will be inserted into the priority queue as the new boundary grids. The lowest grid will be continuously used and removed from the priority queue and the above procedure will be repeated until there is no boundary grid left in the priority queue. By this means, the empty grids are interpolated from the lowest grids to the highest, and the non-ground objects that remains (red grids in the Figure 4-5) surrounded by the empty grids might only affect the interpolation values of its adjacent grids. If the empty grids near where the non-ground object remains were interpolated by the lower neighboring grids in the priority queue, the non-ground object remains would not be used for interpolation. This is very useful to filter out non-ground object remains with a small filtering window size in multiple passes.
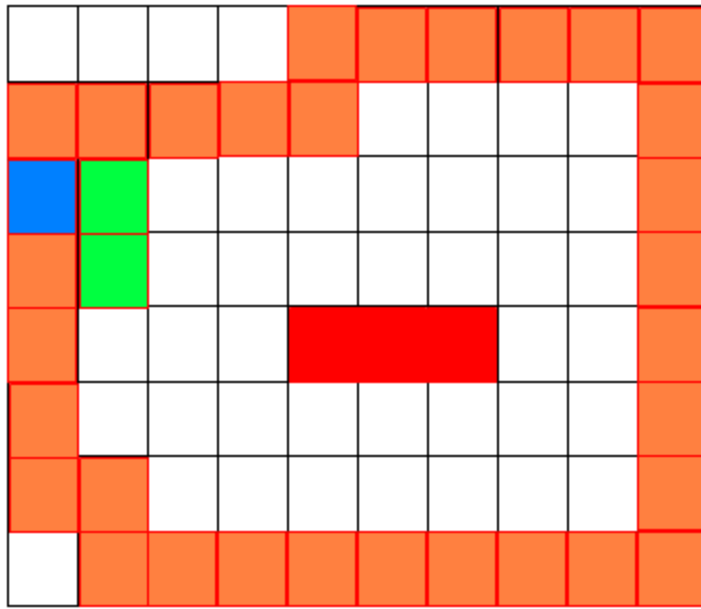
Figure 4-5 Empty grids interpolation from the lowest to the highest

The priority queue data structure is used in this algorithm, which lowers the time complexity and increases the processing speed. Since the time complexity of generating a priority queue initially is O($N$), and the time complexity of insertion and removal are O(log$N$), and the time complexity of searching for the connecting empty grids' group is O($N$), the time complexity of this algorithm is O($N$). The space complexity of this algorithm is O($N$), because the space complexity the of mark matrix and priority queue are O($N$).

The computing time of this method is much greater than the average elevation method, because we have to process interpolation in an order of reference points. In order to make this method more efficient, we can use a priority queue data structure to store the boundary and reference points. First, we need to insert all the boundary grids into the priority queue, and this priority queue is ordered by ascending elevation. We pop up the first element from the priority queue to interpolate its adjacent grids, and push

interpolated adjacent grids into the priority queue. By this means, we can keep the lowest elevation grid at the top of the priority queue, which is the key to making sure the empty grids are interpolated from the lowest to the highest. Also, a priority queue is a very efficient data structure to store and access the ordered data, because its time complexity is $O(\log N)$ in *insertion* and *deletion* operation.

4.2.5   Multi-Pass Morphological Filtering

The grid-based priority interpolation method is a special and dedicated interpolation method to interpolate the empty grids formed by some intermediate filtering result. This interpolation method can be used for a multi-pass filtering strategy. In our processing framework, a multi-pass morphological filter was proposed to process the data set in multiple rounds.

The mathematical morphological filtering method is one of the most widely used techniques in many filters. Among these filters, the progressive morphological filtering [51] is a very effective and efficient method to separate ground and non-ground objects from LIDAR data. More specifically, it can successfully filter and extract non-ground objects from relatively small to intermediate sizes, including trees, vegetation, and small size buildings. The progressive morphological filter can achieve ideal results on many terrain types, especially urban area with relatively small non-ground features. However, it is not working very well in the complex urban areas with large non-ground objects, such as large buildings and constructions. This could cause problems to some urban planning applications (e.g. flood zone planning, transportation planning) and location services, in that the filtering results cannot provide accurate boundary and elevation information for

64

large non-ground objects. The cause of this problem is that it would remove ground terrain objects along with filtering large size non-ground objects when using a large filtering window size. Especially in terrains with untypically low ground features such as channels, it could cause the cut-off problem. This problem could also be observed in the relatively flat terrain with complex non-ground features, if the data set contains large buildings or constructions. This is a common problem for morphological filters when the filtering window size is increased to a certain level.

Zhang et al. [51] use a moving filter window from small scale to large scale to filter out non-ground features. The main operation in this progressive morphological filter is the open operation. The elevation difference between the surface generated before and after the open operation would be compared with a predefined threshold for each filter window size. A point would be removed if its elevation difference is over the threshold. One of the common problems for morphological filters is the cut-off problem when the filtering window size is increased to a certain level. This is an intrinsic problem from the morphological filtering methodology. The morphological filter is a very effective filtering method on many terrain types, especially on flat terrains. However, it could cause the cut off problem on some flat terrains with large non-ground features or relatively low ground features. In order to avoid the cut-off problem on this kind of terrain, the window size used must be kept sufficiently small. However, this could cause incomplete filtering of large non-ground objects. To balance between over-filtering and under-filtering, a multi-pass filter is proposed to remove non-ground features in multiple rounds under relatively small window sizes so as to prevent the ground points being removed after the filtering window size is increased to a certain level. The priority

boundary interpolation method will be used to interpolate the intermediate result for further filtering of the remains of large non-ground features. The experiments show that the proposed method can achieve promising results, while morphological filter itself has difficulty on these complex terrain data sets [24][29].

In order to avoid the cut-off problem on some complex terrains, only relatively small filtering window sizes can be used. Because when the filtering window size is increased to a certain level, the cut-off problem is almost inevitable. Even with some local spatial smoothness justification [10], the morphological filter could still cause the cut-off problem on some complex terrains. In this case, a smaller filtering window size has to be used. However, it could cause incomplete filtering of some large non-ground features. Therefore, a multi-pass morphological filtering strategy is proposed to utilize relatively small window sizes to filter the data in multiple rounds so as to avoid the cut-off problem. And the proposed priority interpolation method will be used to interpolate the intermediate filtering result for a second pass of filtering in order to remove the remaining large non-ground features that cannot be removed with small window sizes.

The general idea of this filtering process is as follows. First, the progressive morphological filter is used to filter the data set in the first pass with relatively small filtering window sizes, which could prevent some ground terrain points from being removed under large filter window size. Second, the priority boundary interpolation is performed to interpolate the empty grids that represent the filtered non-ground areas surrounding the remaining areas of large non-ground objects. Then, the second pass of morphological filtering will use a series of filtering windows from large to small sizes for removing the remaining large non-ground objects. If there are still non-ground features

from large non-ground objects, this second pass filtering might be carried out until the remaining ground features are completely removed.

Since the proposed multi-pass filter utilizes the progressive morphological filtering [51] in each filtering pass which is based on grid data set, the input data set need to be gridded before processing. A representative point is chosen from the points within each grid, and the filter treats each grid as one data point. The filtering status of all the grids can be represented by a mark matrix. Each unfiltered point grid's status will be initialized as either a real unfiltered point from the data set, or an interpolated unfiltered point.

The general procedure of the proposed multi-pass filter is as follows.

1. outputData ← MorphFilter(inputData, window_size_set1, threshold_set1);

2. interpData ← PQBoundInterp(outputData);

3. outputData ← MorphFilter(interpData , window_size_set2, threshold2);

In Step 1, the procedure MorphFilter will carry out the progressive morphological filtering [51] on the input data with a series of window sizes and thresholds. The filtering windows are from small to large sizes, in ascending order. The elevation difference threshold series is pre-defined by the user. In this step, it is the same operation as the progressive morphological filtering. One critical parameter is the maximum filtering window size. In order to prevent large ground features from being removed, the maximum filtering window has to be limited to a certain level, which is determined by the ground terrain characteristics. A half window size is used as each step's filtering window size parameter. The half window size is the number of grids extending to the left or right from the current grid. Commonly used maximum half filtering window sizes for

1 meter cell sized grid data set are usually less than 20, and thus the full window sizes are less than 41. The maximum filtering window size is determined by the resolution of the grid data set and terrain features. Figure 4-6 shows the original complex data set with varying sizes of non-ground features (red areas). Since there are several large size buildings in the data set, only the use of a sufficiently large filtering window can completely remove them. However, a large filtering window could remove some ground surface points as well in areas not so flat or in cases of low elevation artificial constructions (e.g., channels). In order to avoid the cut-off problem when using the morphological filter, the filtering window sizes have to be limited under a certain level. Figure 4-7 shows the filtering result when using the maximum filtering window size 20 on a 1-meter cell size grid data. The result shows that there are a substantial amount of non-ground points from large buildings failed to be removed after Step 1, shown as the red areas in the green rectangles. Figure 4-8 shows the filtering result when using the maximum filtering window size 80 on a 1-meter cell size grid data. The result shows that it has significant cut-off problem in red rectangle area caused by large filtering window size. To avoid the cut-off problem under large filtering window, relatively small filtering window need to be used. However, the filtering remaining from large non-ground objects need to be handled.
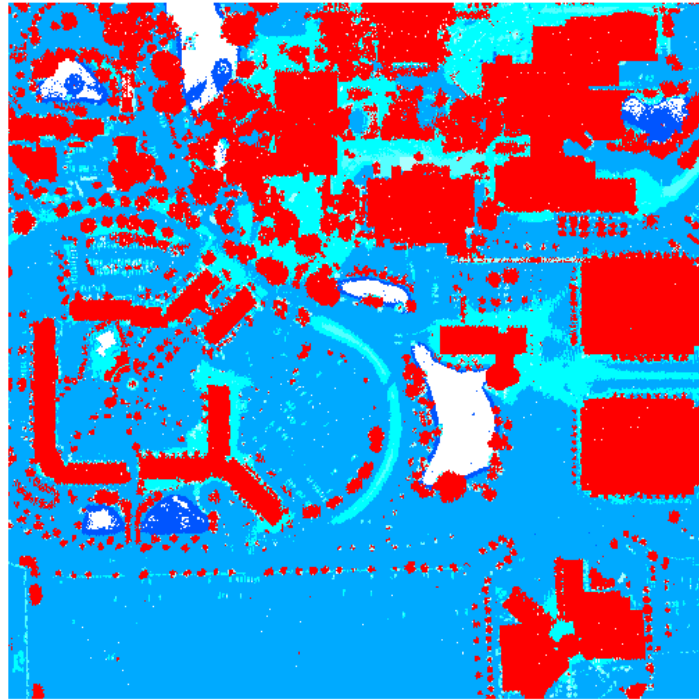
Figure 4-6 The original data set
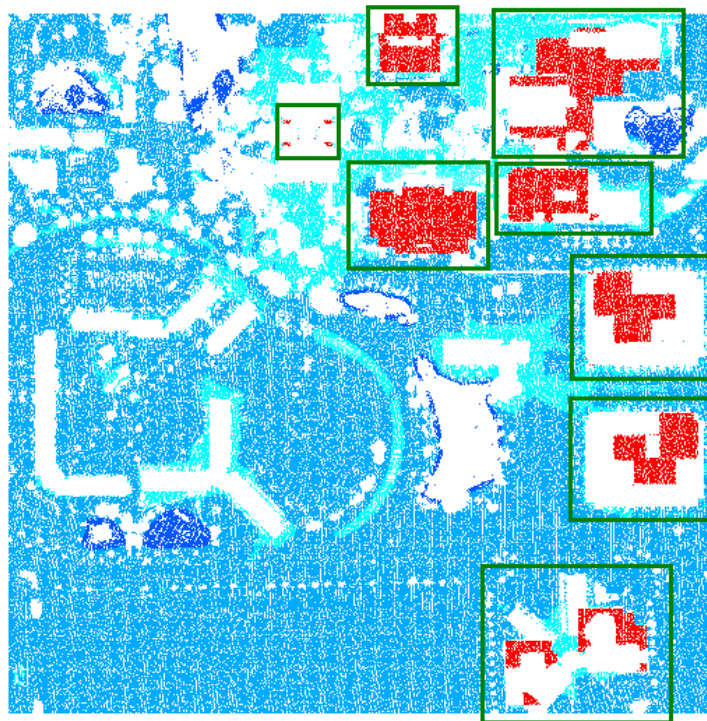


Figure 4-7 The first pass filtering result
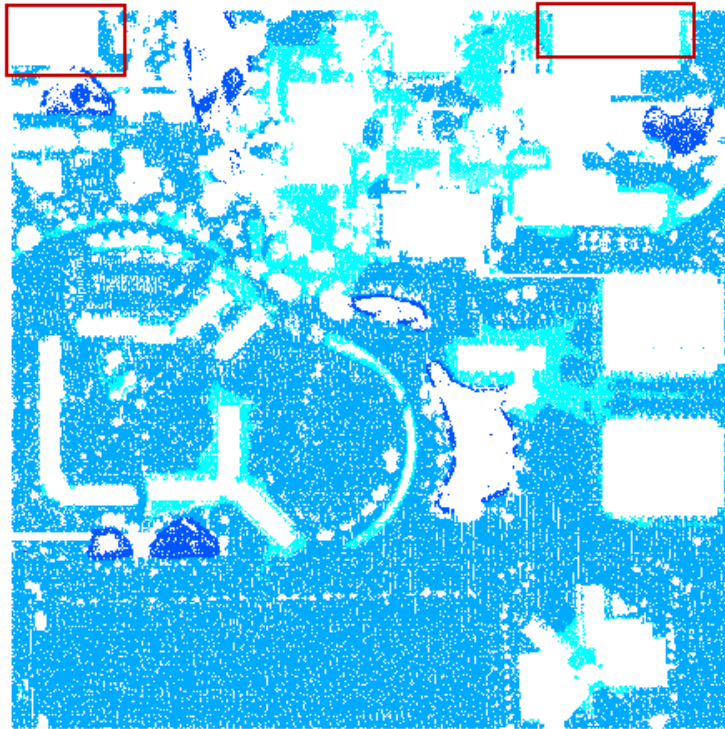
Figure 4-8 Filtering result with cut-off problem caused by large filtering window
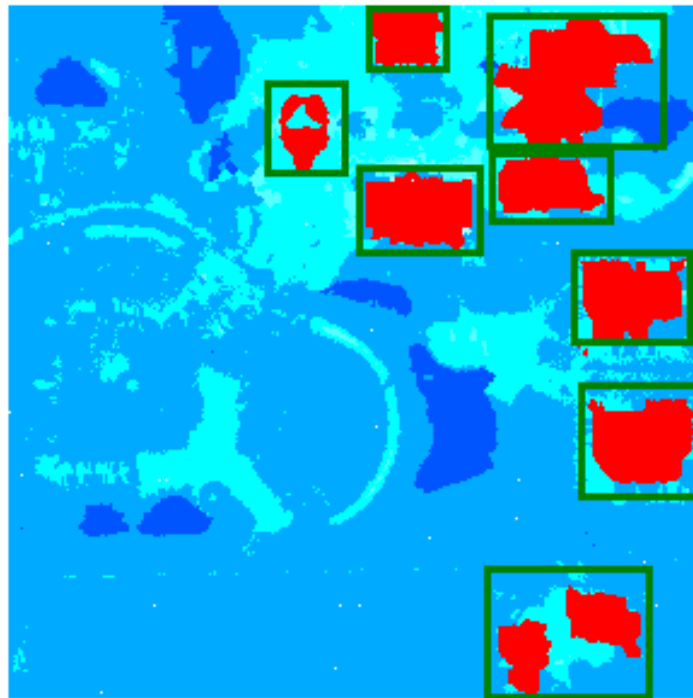


Figure 4-9 The result of applying the nearest neighbor interpolation

In order to further remove the remaining non-ground features, a second pass filtering has to be carried out. However, the empty grids resulted from the removed areas from Step 1 have to be interpolated, before the second pass filtering can be carried out. Since the goal of the second pass filtering is mainly to remove those large non-ground features, a good interpolation method is critical to the filtering. In the first pass filtering, a regular progressive morphological filtering is used in which the nearest neighbor interpolation method is used before filtering is applied. However, using the brute force nearest neighbor interpolation is likely to enlarge the remaining non-ground object area significantly and unnecessarily, forcing the use of a larger window size which would defeat the whole purpose. Figure 4-9 shows the result of directly applying the nearest neighbor interpolation without any constraint. As can be gleaned from this figure, the high elevation points from the boundary of remaining non-ground objects are undesirably involved in the interpolation. Consequently, in the interpolation result, the sizes of almost all the non-ground objects are unnecessarily enlarged in a significant way, causing potential cut-off problems to the second pass filter. Therefore, we must minimize the participation of high points in the interpolation, and use as many low elevation points in vicinity to interpolate empty grids as possible. A more sophisticated and dedicated interpolation method (in Step 2) is necessary for the second pass filtering (in Step 3). This interpolation and second pass filtering can be carried out repeatedly until all the non-ground features were successfully removed.

The second pass filtering in the proposed filtering strategy is also based on the progressive morphological filter. The filtering window sizes and thresholds are different from that of the first pass morphological filter. Since the targets of the second pass

filtering are the remaining points from the large non-ground objects, the filtering window sizes are applied in the descending order. The starting filtering window size is normally the same as the last filtering window size in the first pass, which is also the largest filtering window size in the first pass. Then, the window size is reduced and another round of filtering is performed. The elevation difference threshold for each window size would be relatively large, because it is used for filtering out the remaining large non-ground objects. The elevation thresholds for this purpose are normally much higher than that of each window size in the first pass.

The second pass filtering result is shown in Figure 4-11. The result shows that almost all the remaining large non-ground objects have been successfully removed from the interpolation result shown in Figure 4-10. The filter window sizes series is 20, 16, 8, 4, 2, and 1. The elevation difference threshold used in all rounds is 2.0 meters.
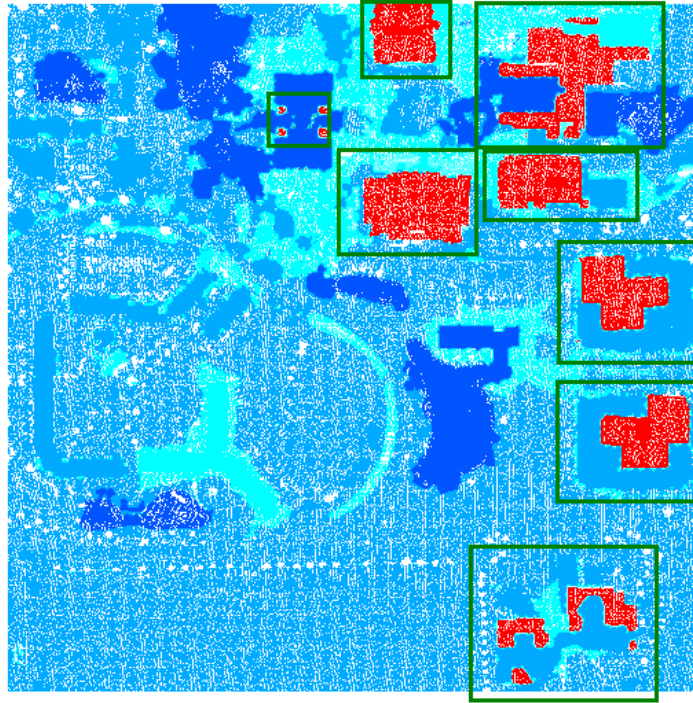
Figure 4-10 Priority boundary interpolation result
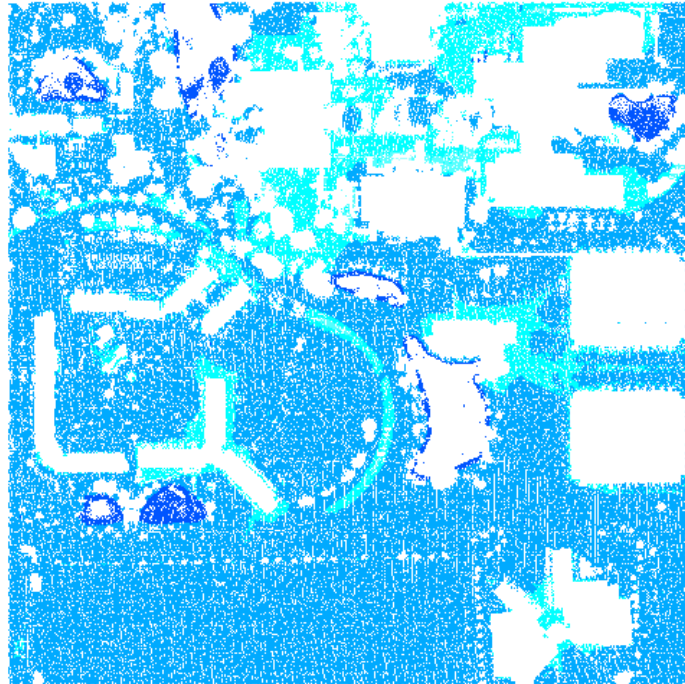


Figure 4-11 The second pass filtering result

## 4.3    Summary

In this chapter, some common outlier removal and LIDAR data interpolation methods have been discussed. These methods provide good solutions for the pre-processing of the data filtering and model generation.

A special and dedicated interpolation method, called Grid-based Priority Interpolation method, was proposed to interpolate the intermediate result for further removing the filtering remains of large non-ground features. This interpolation method was used in the proposed multi-pass morphological filter to process some terrain with large non-ground features. The experiments show that the proposed method can achieve promising results, while the morphological filter itself has difficulty on these complex terrain data sets.

In the later chapters, we will mainly focus on the filtering methods and their parameters' selections, which are the most critical issues of LIDAR processing.

# Chapter 5

## CLUSTER-BASED MORPHOLOGICAL FILTER

In this chapter, a proposed analysis and filtering method called Cluster-Based Morphological Filter is discussed in detail [10]. LIDAR is a widely used technology to measure terrain properties and topographic mapping nowadays. Many filtering methods have been developed to process the LIDAR data to generate a bare earth digital terrain model. Among these methods, mathematical morphological filtering is a very effective and efficient method to separate ground and non-ground objects from LIDAR data. It can achieve ideal results in the flat terrain and undulating terrain with small non-ground objects, while it does not work very well in the undulating terrain with large non-ground objects, because it would remove ground terrain objects along with filtering large size non-ground objects when using a large filtering window size. In this chapter, a cluster analysis and post-processing method is proposed to improve the morphological filtering and make it work better on more terrain types. Since there are still some non-ground object remains left after this cluster-based morphological filtering, some post-processing needs to be carried out to achieve ideal results. A dedicated interpolation method was combined with the post-processing of cluster-based progressive morphological filtering. The filtering results demonstrate that the proposed method is able to reserve terrain ground objects and remove large non-ground objects when the morphological filtering window size increases to a certain level. This process makes this method more effective on more terrain types, which cannot be handled well by other filtering methods.

5.1 Progressive Morphological Filter Review

Zhang et al. [51] proposed a method to remove non-ground objects by using a progressive morphological filter. In their method, an input LIDAR data set was gridded into mesh for 1-D or 2-D morphological filtering. An increasing filtering window size was used during each morphological filtering step. The threshold of elevation difference in each filtering step was variable, which is suitable for removing different sizes of non-ground objects. It proved to be a very effective and efficient method under certain sizes of filtering window on different terrain types, especially on the flat terrain type. As long as the window size is greater than the non-ground objects, and there are enough ground objects as the reference points within the window, those non-ground objects are filtered out with the rational elevation difference threshold. However, this method could make significant errors, along with increasing filtering window size on undulating terrains, such as mountains and sand dunes. This is the case because in the progressive morphological filtering process, the only criterion of identifying non-ground objects is based on the elevation difference between the original elevation value and the processed value returned by open operation, which includes erosion and dilation operations. This would not result in significant errors when the window size is small, because there are not many terrain variations in a small area. However, when the window size increases to a certain level which covers a large area that has significant variations on the terrain, the filtering process will result in categorizing some ground points as non-ground objects, because the elevation difference is above the threshold after open operation. As shown in Figure 5-1, some non-ground objects are on the mountainous surface, which has large variations on the terrain. The terrain surface points are shown in Figure 5-2; however, the

76

progressive morphological filtering will remove some ground points when the filtering window size increases to a certain level, which is shown in Figure 5-3 and Figure 5-4. Ground objects in the red rectangle box of Figure 5-3 would be filtered out as a flat surface, which is shown in Figure 5-4.

Since the elevation difference threshold in a large window size which covers a large area is not able to completely distinguish ground and non-ground objects, some other criteria have to be introduced to keep those ground points that represent the terrain surface not filtered along with non-ground objects. In this chapter, we therefore proposed a cluster-based method to help progressive morphological filtering make better results on the undulating terrain area.
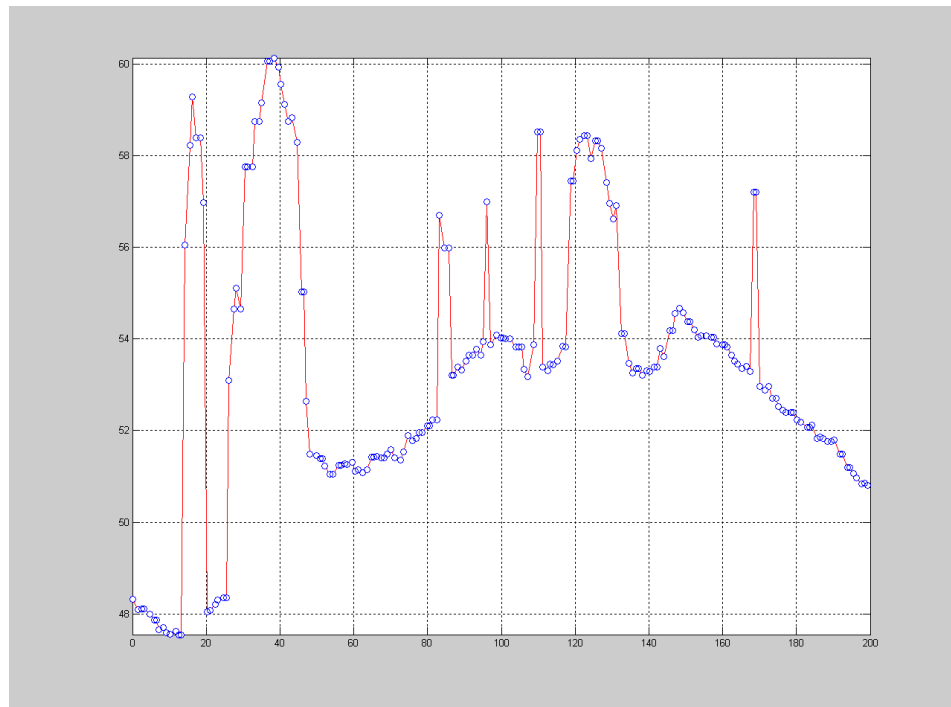


Figure 5-1 Undulating terrain
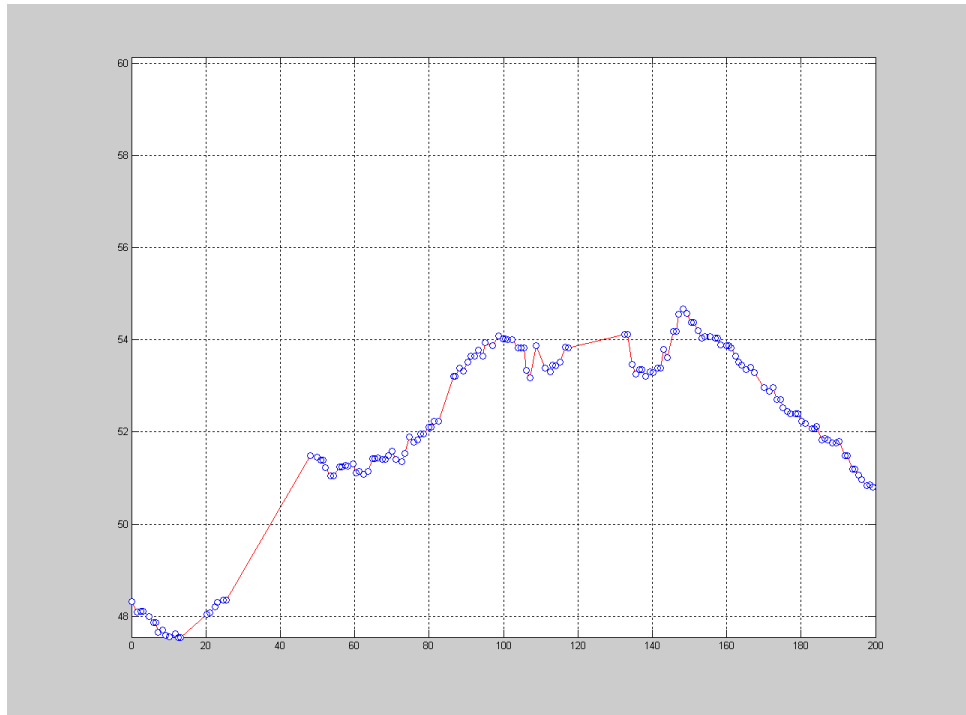
Figure 5-2 Terrain ground surface
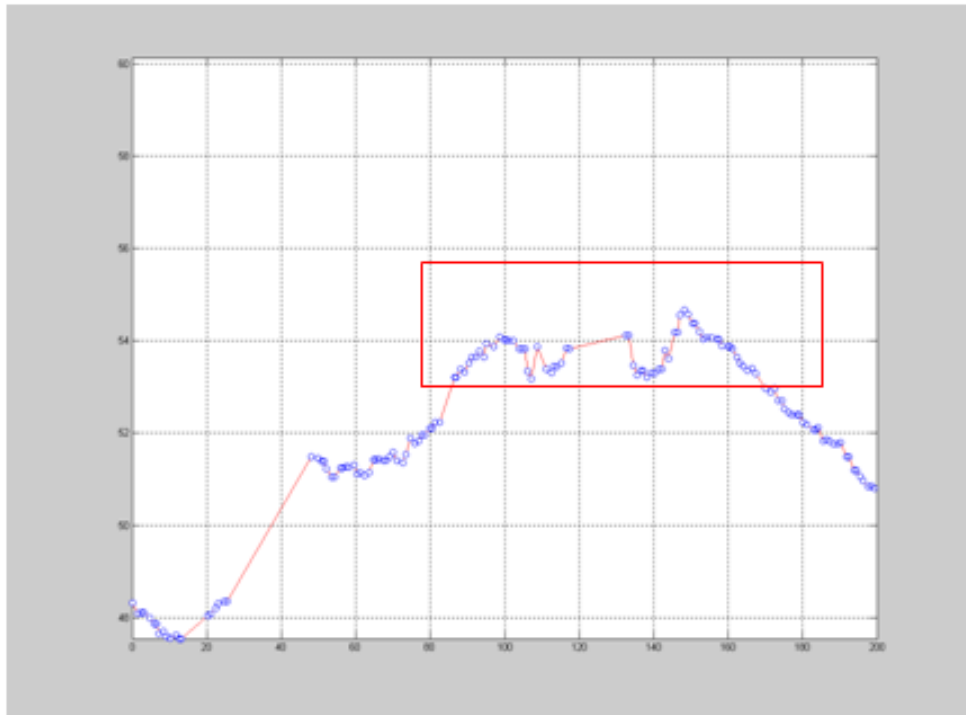


Figure 5-3 Ground surface points
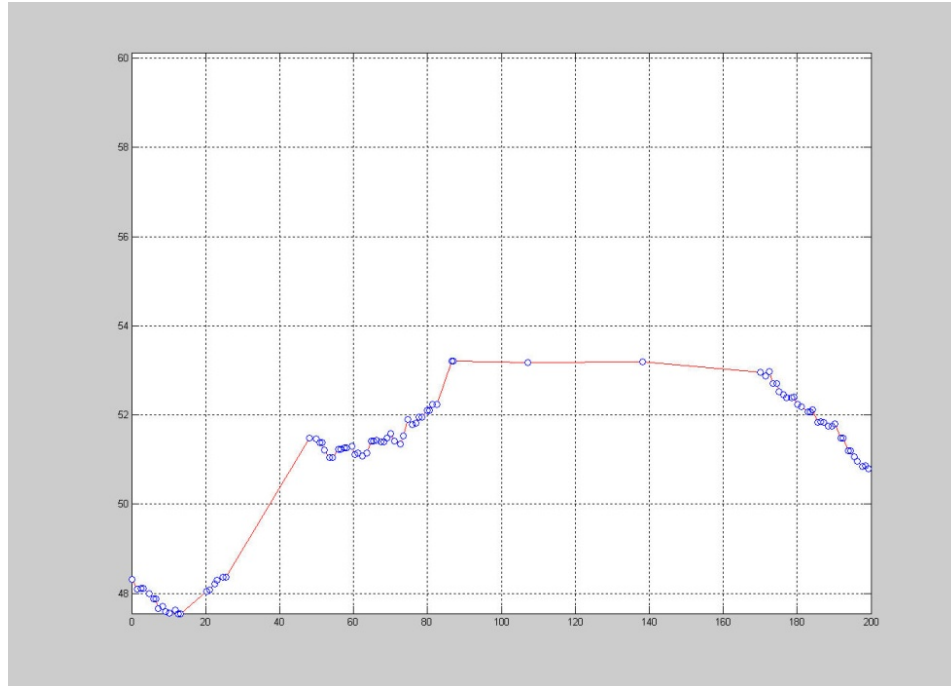
Figure 5-4 Ground surface after filtering

5.2 Cluster-Based Morphological Filter

A new mechanism of justifying ground and non-ground points is proposed to improve the morphological filter in this chapter. This mechanism, called the cluster analysis method [10], is introduced in the filtering procedure. Combined with the progressive morphological filter [51], this method will help the filtering procedure justify the ground and non-ground points identified from the progressive morphological filter and make the filtering results more accurate on a greater variety of terrain types. Since the progressive morphological filter would generate more errors in some undulating terrain such as mountainous areas, especially if there are some relatively large size non-ground objects in the terrain, such as large buildings or constructions, the progressive morphological filter would remove more ground points along with filtering large size

non-ground objects. The key to reducing the errors is to distinguish what kind of point removed during the filtering process is from large non-ground objects and what kind of point is from the ground surface.

The essential aspect of the cluster analysis method is to build up some collections of ground object point clusters for each row or column of the grid data before each filtering step. These collections of clusters will be used to justify the filtering results of the corresponding row or column after each filtering step. The clusters of each row or column are actually the candidate ground surface before each filtering step, some of which could be identified as non-ground objects in the following filtering steps, but they are not known until the final filtering step. If they are indeed ground objects, they would be preserved at the end of the filtering. Otherwise, they would be filtered out as non-ground objects eventually. After each filtering step, some non-ground objects could be identified and labeled with the filtering window size of the current filtering step. These identified non-ground objects will be justified by the clusters generated at the beginning of each filtering loop. If certain criteria are met, these identified non-ground objects will be labeled back to ground objects. That is how the cluster analysis method collaborates with the filtering procedure.

## 5.2.1   Cluster Generation

Since the proposed cluster analysis method and the morphological filter are both based on grid data set, the input data set need to be gridded before processing. Each grid chooses a representative point from the points within the grid, and the filter treats each grid as one point. The filtering status of all the grids can be represented by a mark matrix.

Each unfiltered point grid's status will be initialized with unfiltered point's mark ("0" or "-7777") and stored in the mark matrix. "0" means real, unfiltered points from the data set, while "-7777" refers to the interpolated unfiltered points.

To generate clusters for each row or column of the data grid, the data points will be scanned from the first to the last in each row or column, and unfiltered points with "0" or "-7777" marks will be collected into different clusters based on the elevation difference between consecutive unfiltered points. The method for collecting unfiltered points into the same cluster will be demonstrated on the row direction as follows. It would be working for the column in the same way. From the first unfiltered point of a row, the next unfiltered point is continuously searched and collected into clusters. There could be two scenarios for the next unfiltered point in terms of the spatial relationship. One is that the next unfiltered point is adjacent to the previous unfiltered point; the other is that the next unfiltered point is separated by some filtered points identified in the previous filtering steps. If the two unfiltered points are next to each other, the elevation difference between these two points will be compared with the predefined cluster threshold to determine whether they should be separated into different clusters. If the elevation difference of these two points is less than a predefined threshold for separating clusters, they will be collected into the same cluster; otherwise a new cluster will be created and the following unfiltered point will be collected into the new cluster. However, in the second scenario, the next unfiltered point is separated by some filtered points, and the criteria of separating clusters would be different. The slope of the two unfiltered points' elevation will be compared with a predefined threshold. To simplify these two cases, the same cluster threshold value is used. Thus, the cluster threshold would be a

slope threshold for both scenarios, which is shown in Line 8 of the algorithm description. Different thresholds could be given for the two scenarios if needed. By this means, a collection of clusters that represents the unfiltered points in each row or column of the grid will be generated.

Figure 5-5 shows the results of cluster generation in a row. Points within the same cluster are connected by red lines. There is no line connecting different clusters.

For each row or column of data, there could be some unfiltered points during each filtering step. The cluster analysis method will generate a collection of clusters based on those unfiltered points in each row or column. These clusters are treated as candidate ground point clusters, because some of them are on the ground surfaces, while others are on the non-ground objects' surfaces. The algorithm of cluster generation is described as follows:
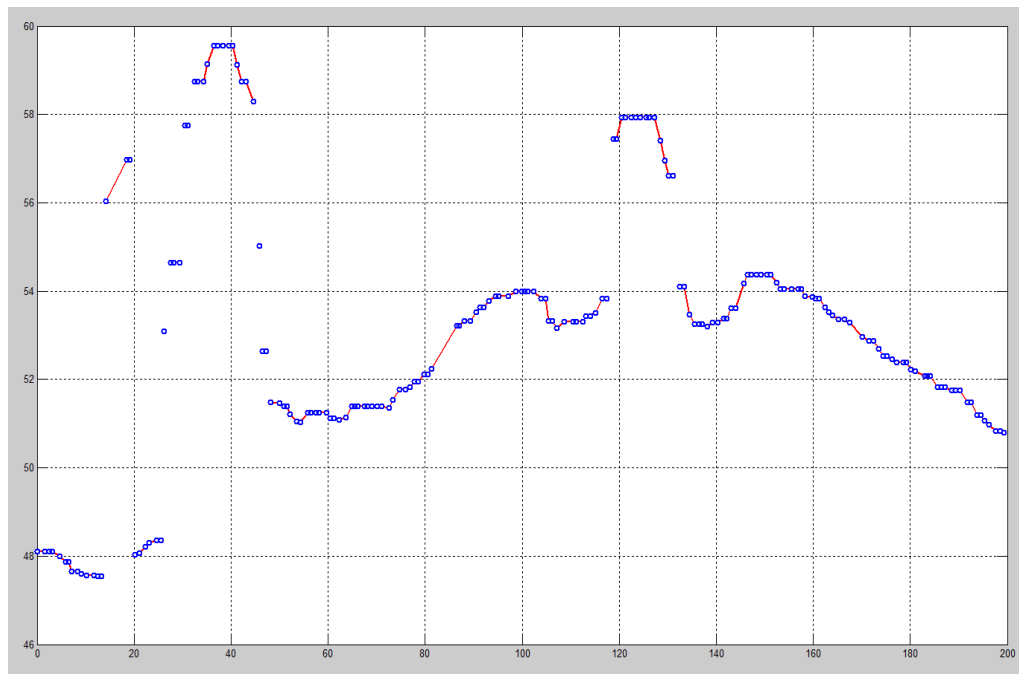


Figure 5-5 Clusters of data points

The algorithm of making clusters for each row or column is as follows:

Algorithm description: *MakeCluster*

INPUT:

   1.An array of z value for a row or column: *rz*
   2.Threshold to separate different clusters: *clusterThreshold*

OUTPUT:

      An array of cluster struct: *clusters*

1. Scan array to find the first unfiltered or interpolated unfiltered point and save the index in prevGrd
2. If prevGrd is NULL then return empty *clusters* array
3. j←1;   // Initial cluster index
4. *clusters*(1) = *CreateTrendCluster*(prevGrd);
5. for i=(prevGrd+1)→ n // n is the size of a row or column
6.    if *rz*(i) is a ground or interpolated ground point then
7.      preDiff = *rz*(i)-*rz*(prevGrd);
8.      if(abs(preDiff/(i-prevGrd))>*clusterThreshold*) then
9.         *clusters*(j).last = i-1; // New cluster is found, record the ending index for the previous cluster
10.         *clusters*(j).postDiff = -preDiff; // Store elevation difference between the last point of the current cluster and the first point of the next cluster
11.         j = j+1;   // Increase the cluster index
12.         *clusters*(j) = *CreateTrendCluster*(i); // Create a new cluster starting from *rz*(i)
13.      end if
14.    end if
15.    prevGrd ← i;
16. end for

The *MakeCluster* algorithm shows that all the points in a row or column will be scanned one after another, and checked whether they can be collected into the same cluster. The prevGrd is used to store the index of the latest found unfiltered point. If no unfiltered or interpolated unfiltered points were found, it would return an empty cluster array in Line 2. Otherwise, at least one cluster would be generated. *CreateTrendCluster* procedure is for generating and initializing a cluster struct. The first unfiltered point's index was stored when the cluster was created in Line 4. The algorithm will continually scan the input row or column data for the next unfiltered point. If the slope formed by the

elevation difference between the two unfiltered points is greater than the cluster threshold, a new cluster will be identified and created. All the unfiltered points that meet the cluster criteria will be collected into the same cluster until a new cluster is found. The index of the last unfiltered point in the same cluster will be recorded in Line 9. The elevation difference between the last point of the current cluster and the first point of the next cluster will be recorded in the current cluster's struct.

After scanning the whole row or column data, one or multiple clusters would be created if the row or column contains unfiltered points. Each cluster stores the indexes of all the unfiltered points that belong to it, and some other information, such as elevation difference between neighboring clusters. This information will be used in the cluster analysis procedure.

5.2.2    Cluster-based Progressive Morphological Filtering Algorithm

In this chapter, a cluster-based morphological filter is proposed by combining the cluster analysis method with the progressive morphological filter. This new filter can effectively prevent the terrain surface points from being removed when the filtering window size is increased to a certain level.

The structure of this cluster-based morphological filter is based on the progressive morphological filter [51]. The input data has to be gridded with a certain grid size and interpolated for the empty grids with some interpolation method, such as the nearest neighbor interpolation. During each step of progressive morphological filtering, a moving window will be used in the open operation, which includes erosion and dilation operations. After the open operation, the elevation difference of each corresponding pair

84

of grids will be calculated between the two surfaces, which are formed before and after the open operation. The elevation difference of each grid pair will be compared with the predefined threshold of the current filtering step to determine whether the point is a non-ground point. If the elevation difference is greater than the threshold, the point would be classified as a non-ground point and marked with the current filtering window size value. In the proposed method, after the filtering window size is increased to a certain level, a collection of clusters for each row or column will be generated before open operation. The cluster analysis will be carried out to justify the mark matrix results at the end of the filtering loop. The unfiltered point clusters are used to represent the candidate terrain surface based on the unfiltered points prior to the current filtering step. A mark matrix is used to record the filtering status for data grids. The algorithm description of this cluster-based morphological filter is as follows:

Algorithm description

MorphClusterFilterFile

INPUT: LIDAR Data File

OUTPUT: Ground Points File

1.  Grid Input file

2.  Interpolate empty grid

3.  Use MorphClusterFilter to process grid data

4.  Output ground points to file

MorphClusterFilter

INPUT:

1. Gridded and interpolated LIDAR data: *Z*

2. Progressive morphological filtering Steps: *steps*

3. Initial cluster windows size to start cluster analysis: *cluster_window_size*

4. Threshold to separate different clusters: *clusterThreshold*

5. Morphological filtering threshold array: *threshold*

6. Morphological filtering window size array: *window_size*

OUTPUT:

Mark Matrix which represents the filtering results: *mark*

1.  Initialize *mark* matrix with "0" or "-7777" marks
2.  for k = 1 to *steps*
3.  for each direction (by row and by column)
4.    for each row/column in grid
5.      zcurr ← *Zi*; // extract the ith row/column's z values
6.      if *window_size*(k)>= *cluster_window_size* then
7.        clusters ← MakeCluster(zcurr, *clusterThreshold*);
8.      end if
9.      zopen ← Morphopen(zcurr, *window_size*(k));// *Open* operation
10.     zdiff ← zcurr – zopen;
11.     if zdiff(i, j) > *threshold*(k) then
12.       *mark*(i, j) ← *window_size*(k); // Update mark
13.     end if
14.     if *window_size*(k)>= *cluster_window_size* then
15.       UpdateMark(*mark*, clusters); // Update mark
16.     end if
17.     *Zi* ← zopen; // Set the ith row/column's z values
18.   end of each row/column
19. end of for loop of each direction
20. end of for loop of all steps

At the beginning of the algorithm in Line 1, the mark matrix is initialized with all

"0" or "-7777" marks, which means that all the points are treated as unfiltered points at

first. "0" means real points in the input data, while "-7777" represents interpolated points. Both marks represent unfiltered data points. The points' marks could be changed in multiple steps of filtering. The steps of the filter, the filtering window size of each step, and the threshold associated with each window size are predefined by the user. Users can also control when the cluster analysis will be activated in the filter by defining *cluster_window_size*. In Line 6 of the algorithm, after the filtering window size is greater than the predefined *cluster_window_size*, the cluster analysis will be activated in the filter. *MakeCluster* procedure will generate the clusters of unfiltered points for each row or column. *Morphopen* will carry out the *open* operation, which includes *erosion* and *dilation* operations. The elevation difference between the two surfaces that are acquired before and after open operation will be compared with a predefined threshold and used to update the mark matrix accordingly for the filtered points. The *UpdateMark* procedure will perform cluster analysis to find whether there is any point that is filtered out in the current step that can be reset to unfiltered points.

In the filtering procedure, the mark matrix represents each grid's filtering status. If the point in the grid is identified as a ground point, it will be set as the ground point's mark value ("0"); if the point is identified as a non-ground point, it will be marked with the current filtering step's windows size, which is a positive value. Therefore, a point can be identified as a non-ground point by the mark that reflects the filtering window size. It also shows the point in which filtering step it was filtered out. An interpolation mark with a negative value is used to represent the interpolated points for the empty grids. If the interpolated point is identified as a non-ground point, it will be marked as the negative

value of the current filtering window size, which is always greater than -7777. The real data points and interpolated points can be easily distinguished in this way.

The initial mark matrix contains only two kinds of values, which are either "0" or interpolation mark value ("-7777"). If the initial mark is not changed until the end of the filtering process, the point with "0" or interpolation mark value indicates the point was classified as a ground point. In each morphological filtering loop, the mark matrix may contain up to four kinds of values, as summarized in Table 5-1.

Table 5-1 Mark values of real and interpolated data point

| Mark value | Meaning |
|---|---|
| 0 | Before filtering: Real data point<br><br>After filtering: Real ground point |
| -7777 | Before filtering: Interpolated data point<br><br>After filtering: Interpolated ground point |
| Positive window size | Real filtered point (non-ground point) |
| Negative window size | Interpolated filtered point (interpolated non-ground point) |

Since the proposed method is based on progressive morphological filtering, an increasing window size is used in each filtering step [51]. Therefore, points filtered in each step will be marked with the window size. Real filtered points and interpolated filtered points will be marked with the positive and negative window size values, respectively. This labeling strategy makes it easy to track in which step the points were filtered out, which reveals at what scale the points were removed (as non-ground points). It provides more information for the cluster analysis. Furthermore, this labeling strategy

is also very helpful for outputting the filtering results, because real ground points and interpolated ground points can be easily separated in this way.

Before each morphological filtering step, there could be some unfiltered points in each row or column. The proposed method will generate a collection of unfiltered point clusters based on each row or column's mark values and their elevation differences. These clusters will help justify the marks of points filtered out in this filtering step.

### 5.2.3 Cluster Analysis

As the progressive morphological filter, which is based on the grid of data with a given cell size, the cluster analysis is also based on a grid. The analysis executes along each row and column of the data set. At the initial filtering step, all the points will be marked as ground points. Even the empty grids will be interpolated with some values and marked as ground points. While non-ground points will be identified and marked in each filtering step of the progressive morphological filter. Therefore, at the beginning of the first filter step, all the points are actually candidate ground points; they might be identified and marked as non-ground points after each filtering step. The detailed marking strategy will be discussed later. The main idea of the cluster analysis method is to use the cluster information to check whether there are ground points or candidate ground points, which could be mistakenly identified as non-ground points. If that is the case, we can reset the marks of those mistakenly identified non-ground points to ground points' marks. By this means, we can leave these candidate ground points to be filtered in the later steps. If they would be filtered out in the rest of the steps, we can also correct the errors of mistakenly identified non-ground points in those steps.

The cluster analysis method can be introduced in any step of the filtering. Due to the problem of distinguishing large non-ground objects (such as buildings or constructions) and ground objects in undulating terrain, we can start involving the cluster analysis after the filtering window reaches a certain size, because it would not result in significant errors under small filtering window size.

Before we do the cluster analysis in the filtering steps, we have to create the cluster collection for each row or column. For each row or column of the grid data, we can scan from the first point to the end, and collect those ground points with mark "0" into different clusters. The interpolated points will be treated as ground points too, but they have a different mark (e.g. "-7777"), which would be used to distinguish between real ground points and interpolated points. The way to collect ground points into the same cluster is as follows. From the first ground point of a row or column, keep searching for the next ground point, which can be adjacent to the previous ground point or separated by some non-ground points identified in the previous filtering steps. After the next ground point is found, we can calculate the slope of the line connected by these two ground points or the elevation difference between these two ground points. We can then compare the slope or the elevation difference with the predefined threshold. If it satisfies the criteria, we can collect this ground point into the current cluster and keep searching the next ground point; otherwise, we start creating a new cluster and use the same method to collect new ground points into this cluster until no ground points can be found. By this means, we can generate a collection of clusters, which represents the candidate ground points of each row or column. Then, we can continue running the filtering process and get the filtered result as in the progressive morphological filtering method. The elevation

difference between the original value and the *open* operation value will be compared with the threshold, and then those qualified points will be marked as non-ground points with the mark value of the current window size (e.g. 8, 16, and etc.). If the non-ground points are interpolated points, they would be marked as negative window size (e.g. -8, -16 and etc.) in order to distinguish real points from interpolated points.

After each filtering step, the collection of clusters generated before filtering will be involved to check whether these non-ground points identified in this filtering step should be classified as non-ground points. The identified non-ground points will be scanned and checked in a certain procedure. If some criteria were satisfied, these identified non-ground points' marks would be reset to the ground objects' mark ("0") for the filtering later. The way to scan and check the identified non-ground points is as follows.

Each row or column's marks will be scanned in order to find one or more collections of continuing non-ground marks with the value of previous filtering window size, which means they are filtered out during previous filtering step. Only the previous step's mark will be checked. That would prevent identified non-ground points in prior steps from being involved in this recovery procedure. Accordingly, we can use the clusters generated before each filtering step to check whether they are completely falling into any ground points' cluster. The collection of the continuing marks is represented by the starting and ending index of these marks, which represents a range of a mark array. The ground objects' cluster created before each filtering step is also represented in the same way. Therefore, we can compare the range of the continuing marks with the range of the ground objects' cluster. It would show whether these marks are completely falling

into any ground objects' clusters. If the collection of continuing marks falls into any ground objects' cluster, we can reset these marks with the ground point's mark value ("0"), which means they will be treated as ground points in the following filtering step or in the final results. In this way, we can prevent those ground points from being removed by progressive morphological filtering, while utilizing progressive morphological filtering to filter out large size non-ground objects. Because it uses the ground points' cluster to gather similar terrain points together in a cluster, this will help avoid omission error from morphological filtering.

A sample cluster collection is shown in Figure 5-6. These clusters represent all the candidate ground points, which include points that could be identified as non-ground objects. The points of each cluster were connected by red lines. There is no line connected between each cluster; accordingly, we can distinguish the clusters by this means. As shown in Figure 5-6, if the points in the rectangle were identified as the non-ground objects' points in the morphological filtering procedure, we would check whether these points are completely falling into a cluster. As in the figure, they are completely located in a cluster; therefore they would be reset to ground objects for the next filtering step. In this way, we can reserve those candidate ground points on a ground surface effectively. If these reserved candidate ground points were on the non-ground objects' surface, they would be removed as the non-ground objects along with all the non-ground objects' points in the later filtering steps.
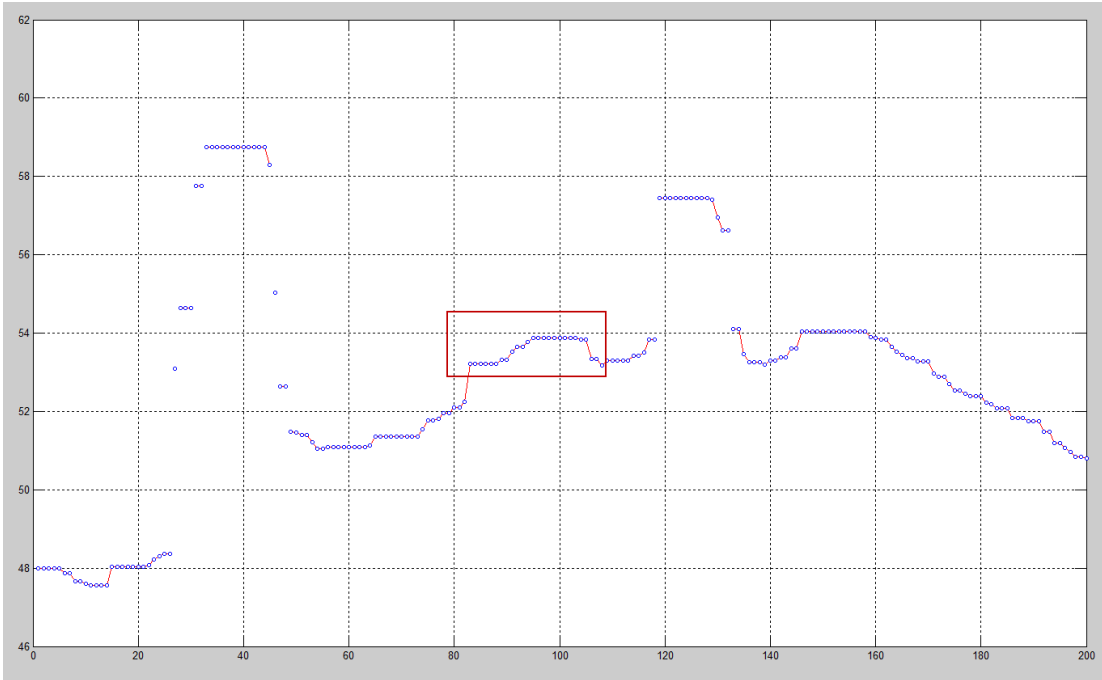
Figure 5-6 Clusters of points

After creating clusters for each row or column, we can use the progressive morphological filtering to filter the row or column, and set identified non-ground points' marks with the current window size. Once we complete the filtering, we can use the clusters created before the filtering to justify the identified non-ground objects and update their marks. If the consecutive non-ground points marked with the value of current filtering window size, which means they were identified in the current filtering step, are falling in any cluster of current row or column, their marks would be reset to ground objects' marks. By this means, we can prevent ground terrain objects from being mistakenly filtered out under large window size. As the tradeoff, this method would restore some identified non-ground points to ground points. Since we normally use the cluster analysis under a large window size, the error would happen to those large size non-ground objects, such as large buildings or constructions. However, the error would

93

not happen to every non-ground objects, it would happen on some of the large size objects. Therefore, after the cluster-based progressive morphological filtering, large non-ground object would be split into smaller pieces. We would use the second round progressive morphological filtering to remove those non-ground objects left in the first time filtering. In this second round filtering, a special interpolation method would be used.

A special interpolation method discussed in the previous chapter, called Grid-based Priority Interpolation [11], has to be used to interpolate the result from the previous cluster-based progressive morphological filtering. Since the previous filtering has removed a certain number of the non-ground objects, most non-ground objects' remains are from large size objects, such as buildings or constructions. These remains are normally surrounded by many empty grids, which were made by filtering large size non-ground objects. In order to successfully remove these remains and not remove terrain ground objects, we use a smaller window size, which would not be likely to remove terrain objects. Due to this reason, we have to interpolate these empty grids with as many surrounding ground points as possible. By this means, if the empty grids were interpolated by more ground points' values, the non-ground object remains would be more easily removed under small window size. The interpolation would be carried out briefly as follows:

1. We first use the expanding grid method to identify all the connected empty grid sets.

2. Second, we can identify the boundary of each empty grid set.

3. Finally, we can use these boundary grids to interpolate the empty grids.

In the first step, each empty grid's 4 or 8 neighbor grids will be checked. If the neighbor grids are empty grids, they would be collected into the same connecting set. The empty grids will be searched until no more connected empty grids are found. Finally, all the collected empty grids will form a set of connected empty grids. A unique identification number will be assigned to each connected empty grid set. After searching each of the whole data grids, one or more connected empty grid sets with a unique identification can be found. We can interpolate each connected empty grid set with some algorithms for different purposes.

From each collected empty grid set, we can easily achieve the boundary points of the set. The simplest interpolation method is to use the average elevation of all the boundary points of the set. For the post-processing of the cluster-based progressive morphological filtering, we can use this algorithm in order to filter high non-ground object remains. However, the average elevation of all the boundary points of the set does not represent the local terrain feature. There are two major reasons for this. One reason is that if the outside boundary grids cover an area with a large elevation difference, the average elevation value would not reflect the terrain variation very well. The other reason is that the non-ground object remains would normally be surrounded by the empty grid set. That would make these non-ground object remains be the inside boundary grids of the empty grid set, and thus they would be involved in calculating the average difference, which would make the average value contain more errors. Since the purpose of the interpolation on the empty grids is to filter out the non-ground object remains in the post-processing procedure, this simplest interpolation will do its job in many cases, because if

the non-ground object remains are high elevation points, they would be easily filtered out by the surrounding interpolated empty grids with low average values.

Despite the fact that the nearest neighbor interpolation can work well on many cases, some undulating terrains still need better interpolation for the post-processing. A more sophisticated method can be used to interpolate the empty grids. Since the elevation of the boundary points could vary greatly, we should interpolate the empty grids from the lowest elevation boundary points to the highest. We first start interpolation from the lowest boundary point, and then its adjacent empty grids would be found. These empty grids are interpolated first with the average elevation of adjacent neighbor grids. Then, they are used as the boundary points to interpolate their neighbor empty grids. The lowest elevation boundary point would be removed from the boundary after it is used to interpolate all its neighbor empty points, because the interpolated empty grids connected to it would replace it to form a new boundary. The procedure of the interpolation is illustrated in Chapter 4.2.4.

5.2.4   Filter Results and Analysis

Filtering experiments have been carried out on the sample data set in California. The testing data set is an undulating terrain with different sizes of non-ground objects, such as trees, vegetation, and buildings, etc. The data set covers a 200 x 200 square meter area. We used 1 meter as the cell size to grid the data set, which gives a 200 x 200 grid mesh. The original data set's 3-D mesh diagram is shown in Figure 5-7.
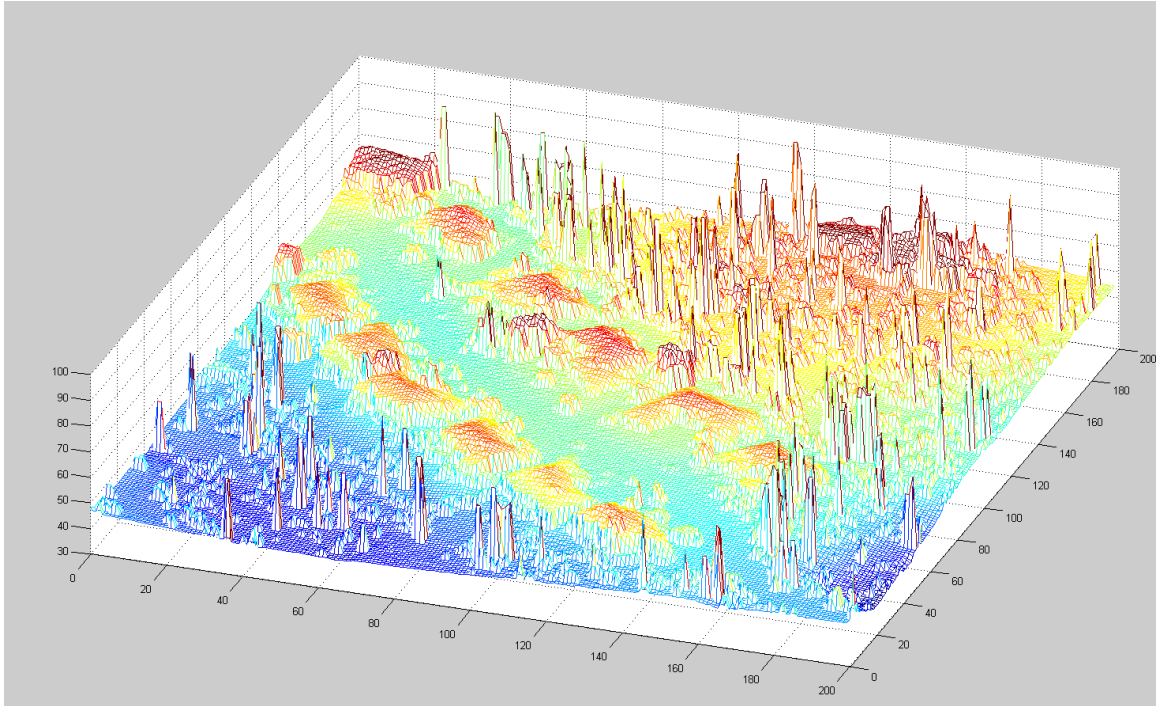
Figure 5-7 Original data set 3-D mesh

First, we used the progressive morphological filtering to filter the data set. The total filtering step is 7. We used a half window size as each step's window-size parameter, which is grid size extending to the left or right from the current point grid; thus, the real full window coverage would be twice of half the window size plus one. Figure 5-8 shows how to calculate the full window size by the half window size. The full window size is used in the filtering procedure.

97
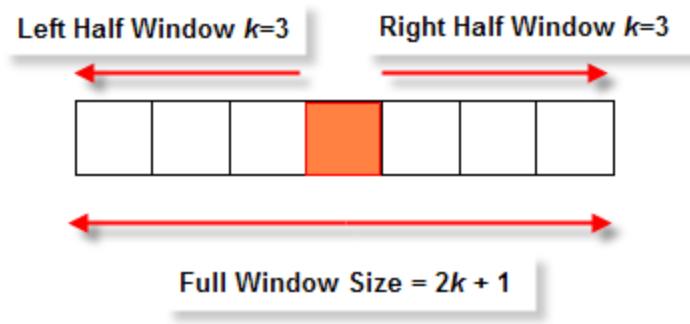
Figure 5-8 Half window size vs full window size

The half window size series of 7 filtering steps is 1, 2, 4, 8, 16, 20, and 32, and the full window size series is 3, 5, 9, 17, 33, 41, and 65, respectively. The results of the two filters are shown in 2-D figures. Figure 5-9 shows the results of the progressive morphological filter and the cluster-based progressive morphological filter. Without the cluster analysis procedure, the progressive morphological filter will remove some terrain objects, which are illustrated in the green rectangles when the filter window size increases to a certain size. Compared with two figures, it demonstrates that the proposed cluster analysis method can effectively improve the filtering results of the progressive morphological filter. This method would be very helpful to process a large area or undulating terrain data set, because it would be very sensitive for the filter to remove terrain objects when the filtering window size increases to a certain level.
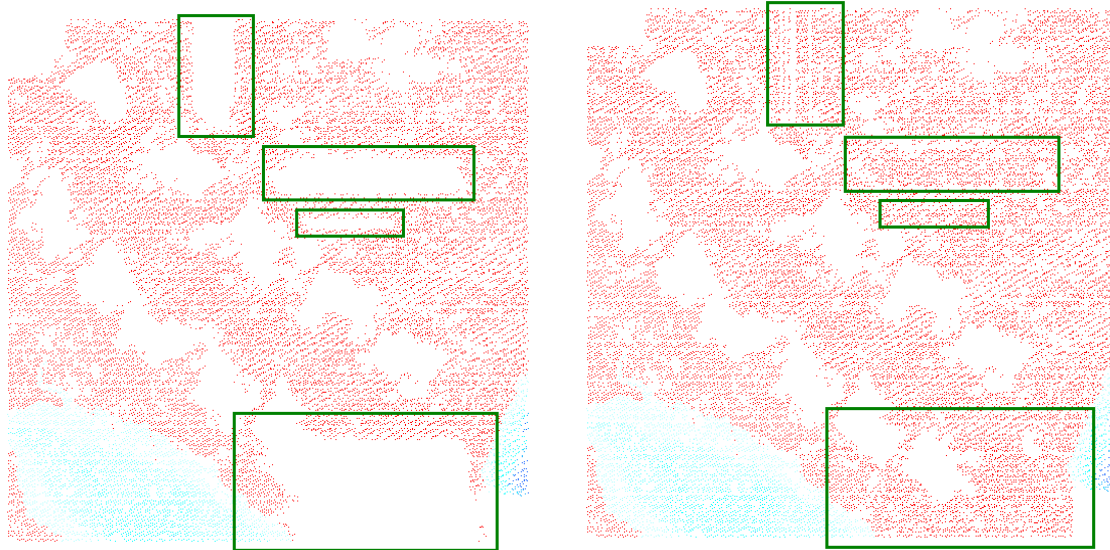
Progressive morphological filtering result      Cluster-based progressive morphological filtering result

Figure 5-9 PM filtering result vs cluster-based PM filtering result

The filtering results show that the progressive morphological filter will remove the ground surface points in several areas significantly. Those areas are shown in the green rectangles. The progressive morphological filter's results demonstrate the common problem of many morphological filters. When the filtering window size reaches a certain level, the ground points were inevitably removed by the filter. That is the intrinsic problem from the morphological filter itself. This problem would be enlarged on the undulating and complex terrain, such as a mountainous area. As in this testing data set, there are several buildings located on the mountain slope. It would be very easy to cause the cut-off problem after the filtering window reaches a certain level. However, if only smaller filtering window sizes were used in the filter, it would cause some buildings to not be removed. This would be a trade-off on the omission and commission errors of the filter. To avoid this kind of problem, some other mechanism, which can detect the connectivity of ground surface, has to be introduced. The proposed cluster analysis

method would assist in lowering the omission and commission errors at the same time. As shown in the filtering results, the cluster-based morphological filter can reserve the ground surface effectively. It can prevent the cut-off problem remarkably when the filtering window size reaches a certain level. The green rectangles indicate that the ground points on the mountain slope were well reserved, and the buildings were successfully filtered out as well.

Based on the algorithm description of the proposed cluster-based morphological filter, the time complexity of the algorithm should be the same as the progressive morphological filter. Since the proposed filter is based on the progressive morphological filter, the major computation time for the progressive morphology filter is the erosion and dilation operation. In addition to the interpolation, the time complexity for the open operation is O($wN$), where w is the window size of the morphological filter and N is the number of grids, which is the product of the number of rows and columns. For $M$ windows, the time complexity is equal to [51]

$$O(\sum_{k=1}^{M} w_k N)$$

Since in the cluster generation procedure *MakeCluster*, the whole data set just needs to be scanned once to generate all the clusters, and the time complexity of cluster generation is O($N$). In the cluster-analysis procedure, the time complexity of the cluster search for each reset range is O($logN$). Based on a binary search, the overall time complexity of the cluster-based morphological filter stays the same as the progressive filter. The space complexity of the proposed algorithm is O($N$), because the space

complexity of mark matrix and clusters are both O($N$), which turns out the same space complexity.

5.3 Conclusion and future work

Experiments results show that the proposed cluster-based morphological filter turns out to improve the progressive morphological filter on undulating and complex terrain significantly. Especially when the filtering window size increases to a relatively large size, it would prevent the ground terrain from being removed while it is still able to filter out large size non-ground objects.

The cluster analysis method can be extended to two-dimensional directions. The cluster recovery mechanism needs to be further refined to suit 2-D analysis.

The selections of the filtering window size and threshold parameters are very critical and sensitive to different terrain types. It would be very helpful to analyze the study area's terrain types and non-ground features for choosing all the appropriate parameters. Therefore, this method still needs some human interactive procedures to involve. Some adaptive parameter selection methods might be developed to automatically choose the appropriate parameters for different terrain types, which would make this method more efficient and automatic.

# Chapter 6

## ADAPTIVE TREND ANALYSIS MORPHOLOGICAL FILTER

In this chapter, a proposed terrain analysis and filtering method called Adaptive Trend Analysis Morphological Filter is discussed in detail [12]. As LIDAR has become a widely used technology in surveying and industrial measurement applications in recent years, many filtering methods have been developed. However, all the methods have their pros and cons, and all of them work well on certain types of terrain. To achieve good filtering results, experienced people are needed to process the data with appropriate methods and parameters. One of the major reasons why some methods are not working very well is there is no way to detect the terrain variation in the big survey area, which would make it very difficult to select the parameters for the filtering method. For example, the progressive morphological filter works very well on the flat terrain, but it does not work very well on undulating and mixed terrains. Since the filtering parameters on different terrains are variable, it would be very difficult to choose a general value for filtering on different terrain types. Therefore, effective and efficient methods to analyze the survey data and automatically calculate the parameters are very important to many filtering methods. In this chapter, a trend-based adaptive method is proposed to make the filtering process more automatic with fewer human interactions. We will demonstrate how it would be used to automatically estimate the filtering parameters for morphological filtering. This method can be utilized in any filtering methods which need the terrain information for estimating filtering parameters.

Since most morphological filtering methods need experienced user to select filtering parameters for different kinds of terrain types, there is no general way to choose

parameters based on the terrain types. This trend-based adaptive method provides a mechanism to adaptively select filtering parameters according to the terrain variations, because it can automatically analyze and detect the variations of the terrain. By this means, it would reduce a lot of human efforts of selecting filter parameters, and make the filter method more automatic and less dependent on human interactions, thereby making it much easier for inexperienced users.

6.1 Morphological Filter Review

Since the morphological filter normally uses different sizes of filtering windows to filter the data, for each window size, different threshold values have to be selected as the criteria for separating ground and non-ground objects. Most of the morphological filtering methods use the constant value for each window size, which would make the filter not suitable for complex terrain types. Because the constant threshold value for the same morphological filter window implies uniform ground slope in the data set, it would work well in the simple terrain area, such as flat ground surface, while it would not work very well in the complex terrain area, especially undulating terrain, such as mountainous areas.

Zhang et al. [51] proposed a method to remove non-ground objects using a progressive morphological filter. In their method, an input LIDAR data set was gridded into mesh for 1-dimension or 2-dimension morphological filtering. An increasing window size was used during each morphological filtering step. The threshold of elevation difference in each filtering step was variable, which is suitable for removing different sizes of non-ground objects. It is a very effective and efficient method. As long

as the window size is greater than the non-ground objects, and there are enough ground objects within the filtering window as the reference points, those non-ground objects will be filtered out with appropriate elevation difference threshold. Therefore, the selection of elevation difference threshold is very critical and sensitive for the morphological filter under certain window sizes. A constant value was normally selected as the threshold under each filtering window size. This would be the same threshold for all the points during the filtering at a certain window size. This has some intrinsic defects when filtering complex terrains, because different parts of a large survey area could have different terrain types: the ground surface would have different slopes; therefore it would not be ideal to have the same threshold to filter terrains with variant slopes. Furthermore, any surveyed area could have arbitrary non-ground objects with different sizes. It would not make sense to use the same threshold for different surveyed areas with various terrain types under the same window size. These problems would require the user to have more experience and knowledge on how to choose the filtering threshold under various window sizes. In order to solve these problems and make the filtering process rely less on human interactions, an adaptive trend analysis method was proposed in this chapter, which was combined with cluster method proposed in the previous chapter, to make the filter automatically select the filtering threshold under each filtering window according to the local terrain variations.

6.2  Algorithm and Implementation

A new mechanism of estimating local terrain variations is proposed to help filtering methods select parameters in this chapter. This mechanism, called the trend

analysis method, was introduced in the filtering procedure. We will demonstrate this method by combining it with the progressive morphological filter. It can also be embedded in any filter methods which need terrain slope information. By working together with the cluster analysis method, this method will help the filtering procedure automatically choose the filtering parameters based on the analysis of local terrain characteristics during the progressive morphological filtering and make the filtering adaptive to different kinds of terrain types.

The essential aspects of this adaptive filtering method include two major analysis means to estimate the filtering threshold for each point. One is the cluster analysis method, and the other is the trend analysis method. The cluster analysis method is used to build up some collections of point clusters for each row or column of the grid data; the trend analysis method is used to analyze the local surface trend for the data set. If these two analysis methods are combined, some important filtering parameters can be estimated automatically. For the progressive morphological filter, one of the important filtering parameters, the filtering threshold, is normally given by the user. It is a constant value under each filtering window. The selection of this parameter would affect the results significantly, and it relies on the users' processing experience and knowledge of the surveyed area. Through this estimation process, the filter method would be more adaptive and have fewer human interactions involved, which means it is not required for users to have much experience on the filtering method or much familiarity to the surveyed area.

6.2.1    Cluster Analysis Method

As we discussed in the previous chapter, the cluster analysis is based on a grid data set, therefore the data set needs to be gridded before processing. After that, it can be carried out on the rows or columns of the grid data. For each row or column of the grid data, the data points are scanned from the first to the last, and unfiltered points with zero marks are collected into different clusters. The method for collecting unfiltered points into the same cluster is as follows. We will show how it works on the row direction. It would work for the column in the same way.  From the first unfiltered points of a row, we will keep searching for the next unfiltered point and collect it into clusters. There are two scenarios for the next unfiltered point in terms of the position relationship. One is that the next unfiltered point is adjacent to the previous unfiltered point, and the other is that the next unfiltered point is separated by some filtered points identified in the previous filtering steps. If the two unfiltered points are next to each other, we can use the predefined cluster threshold to check whether they should be separated into different clusters. If the elevation difference of these two points is less than the predefined threshold for separating clusters, they are collected into the same cluster, otherwise a new cluster is created and the following unfiltered point is collected into the new cluster. While in the second scenario, the next unfiltered point is separated by some filtered points, the criteria of separating clusters would be different. The slope of the two unfiltered points' elevation would be compared with the threshold. By this means, we can generate a collection of clusters, which represents the unfiltered points of each row. Figure 6-1 shows the result of the clusters of a row. Points within the same cluster were connected by the red lines. There is no connecting line between different clusters.
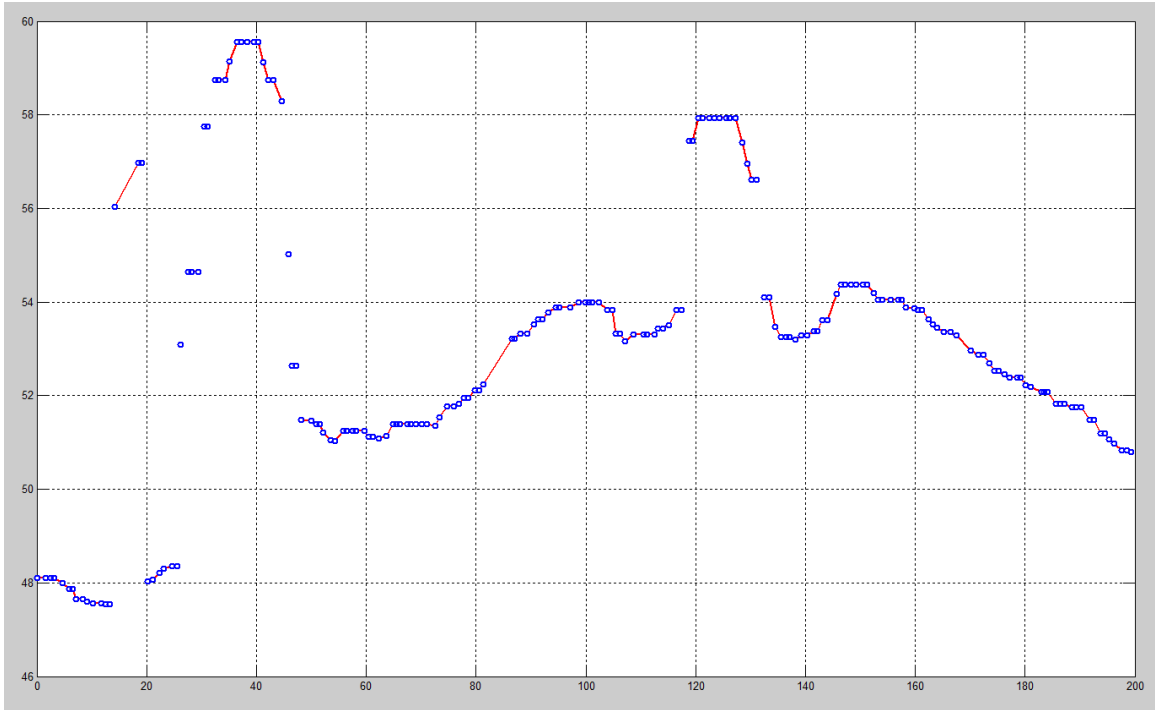
Figure 6-1 Clusters of data points

For each row or column data, there are some unfiltered points. The cluster analysis method will generate a collection of clusters based on those unfiltered points in each row or column. These clusters are treated as candidate ground point clusters, because some of them represent the ground surfaces, while others represent non-ground objects surfaces. These clusters will help the trend analysis method estimate filtering parameters, which will be discussed later. The algorithm of identifying clusters (*MakeCluster*) was described in Chapter 5.2.3.

The *MakeCluster* algorithm shows that all the points in a row or column are scanned one after another, and checked as to whether they can be collected in the same cluster. If the elevation difference between two unfiltered points is greater than the cluster threshold, a new cluster is separated and created. All the unfiltered points that satisfied the criterion are collected in this cluster until a new cluster is found. After scanning the

107

whole row or column data, one or multiple clusters are found. Each cluster stores all the unfiltered points that belong to it, and some other cluster information, such as elevation difference between neighboring clusters. Each grid point's filtering status is represented and stored in a matrix with the same dimension of the partitioned grid.

6.2.2   Trend Analysis Method

The trend analysis method is also based on the rows or columns of the grid data. For each row or column of the grid data, the data points are scanned from the first to the last, and the local minima and maxima of the unfiltered points with zero marks are identified and collected into minima and maxima arrays. So-called local minimum is the point that has a lower elevation than its neighboring points, while local maximum is the point that has a higher elevation than its neighboring points. The way to identify local minima and maxima points is to compare each unfiltered point's elevation with the elevations of its left and right unfiltered neighboring points. If the point's elevation is less than its neighboring points, it would be identified as a minimum; if the point's elevation is greater than its neighboring points, it would be identified as a maximum. In our analysis method, we used a loose criterion to check the minima and maxima. If the point's elevation is less than one neighboring point and less than or equal to the other neighboring point, it would be identified as a minimum; if the point's elevation is greater than one neighboring point and greater than or equal to the other neighboring point, it would be identified as a maximum. The criteria of minima and maxima are not strictly less than or greater than both neighboring points. In other words, the minima and maxima points' elevation can be equal to one of the neighbor points' elevation but not both. These

criteria would prevent the points with the same elevation as its two neighboring points from being identified as minima or maxima points, while keeping points which have equal elevation with one side neighboring point and different elevation on the other side as minima or maxima. This would be very useful to reveal the terrain shape for analysis. By this means, we can identify all the minima and maxima points in a row or column of the grid data. We can further draw the maxima and minima trend lines by connecting maxima and minima points correspondingly.

Figure 6-2 shows the trend lines made by the local minima and maxima. Each point was shown as a blue circle in the figure, and they were connected by red lines. The minima were marked with a black dot inside the circle and connect by black lines, while the maxima were marked with a green cross inside the circle and connected by green lines. From the figure, it was shown that the connected lines made by maxima and minima form two trend lines, which roughly reflect the terrain and non-terrain variations. On the terrain without an abrupt elevation change, it was shown that the maxima trend lines and minima trend lines make a narrow strip, which can describe the slope of the local terrain variation. This is a very useful result achieved from the trend analysis method, because it reveals the terrain characteristics. By combining the trend analysis results with the cluster analysis results, more information can be further concluded for the filtering method. One of the utilizations of this method is to estimate the filtering threshold under a certain window size.
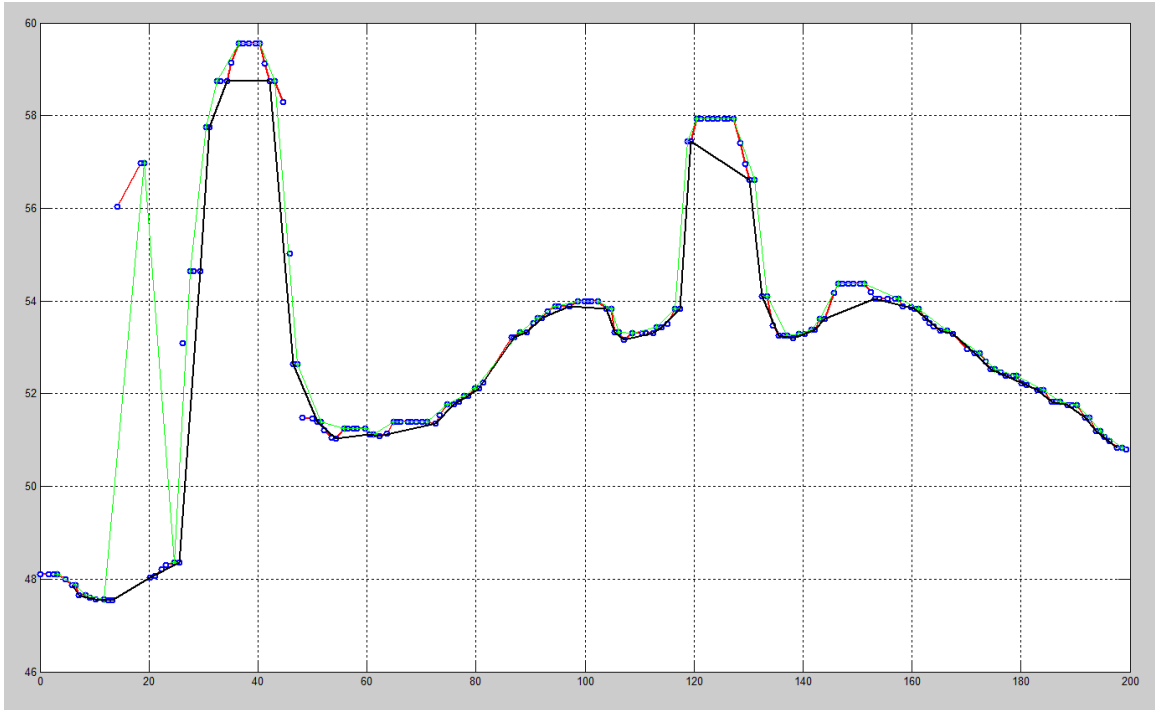
Figure 6-2 Minima and maxima trend lines

6.2.3    Filtering Threshold Estimation

The combination of the cluster and trend analysis methods provides a mechanism to estimate the morphological filtering threshold under a certain window size. By introducing this estimation method into the morphological filter, we can fulfill the adaptive filtering, which means the filtering parameters would be automatically selected according to the local terrain variation. This would not only significantly help the inexperienced filter users process the LIDAR data, but make it easy to process large volumes of data for all users as well. In the experiments, this estimation method was combined with the progressive morphological filter. The cluster and trend analysis procedure is embedded into the progressive morphological filtering loop. Since each morphological filtering loop has its filtering window size, the threshold estimation

procedure is based on the filtering window size and the result of cluster and trend analysis.

In each step of progressive morphological filtering, a certain filtering window will be used to filter each point. Window size is normally represented by the number of grids it can cover. Figure 6-3 shows the full filtering window size and the half window size. The full window size consists of two half-window size and one grid size. If the half-window size is $k$ grids, the full window size would be *2k+1* grids. When executing morphological filtering, each point is filtered by erosion and dilation operation in the full window size coverage. As the input parameter, the filtering window size normally uses half window size, because it is easier to calculate the full window size value with any half window size value. The full windows size cannot use arbitrary value because it would cause the left and right half window size to be not equal if the window size provided is not appropriate. Therefore, the terrain variation in the full window coverage would determine the filtering parameters such as threshold. Our adaptive filtering method is based on analyzing the cluster and trend properties of each point and its coverage window, so that it can automatically estimate the filtering threshold.

Figure 6-3 Half window size vs full window size

There are two scenarios when estimating the filtering threshold of a point under a certain filtering window size. The half window will extend the coverage from the current point to both the left and right directions. Accordingly, each direction will reach a grid point, which can be called the left and right bounding grid points. The left and right bounding grid points' cluster and trend properties are involved in the threshold estimation. The current filtering point and left/right bounding grid points' cluster properties make two special scenarios. One is that these three points are in the same cluster, and the other is that they are in the different clusters. When estimating the filtering threshold, these two scenarios are handled differently.

The first scenario is that these three points are not in the same cluster. If the current point's cluster is below its two neighboring clusters, the current point would be more likely on a lower surface, which would probably not be filtered out during this filtering loop. If the current point and the bounding points are in different clusters, and its cluster is above one or two bounding points' cluster, the current point would be more likely to be a non-ground point that needs to be filtered. This scenario was illustrated in

Figure 6-4. The data points are in three different clusters, which were separated by the dash lines. When estimating the threshold of the point shown in the figure, its left and right half window size coverage have to be considered. From the current point, the filtering window's leftmost and rightmost grids can be found by extending the half window size from the current point to the left and right directions. These two points can be called left and right window bounding points. Further, these two bounding points' cluster and trend properties can be found by searching their corresponding cluster and trend. From the cluster's coverage property, it reveals which cluster the current point belongs to, and the elevation difference between that cluster and its neighboring clusters. The elevation difference of the neighboring clusters is the elevation difference between the two adjacent ending points in the two consecutive clusters.
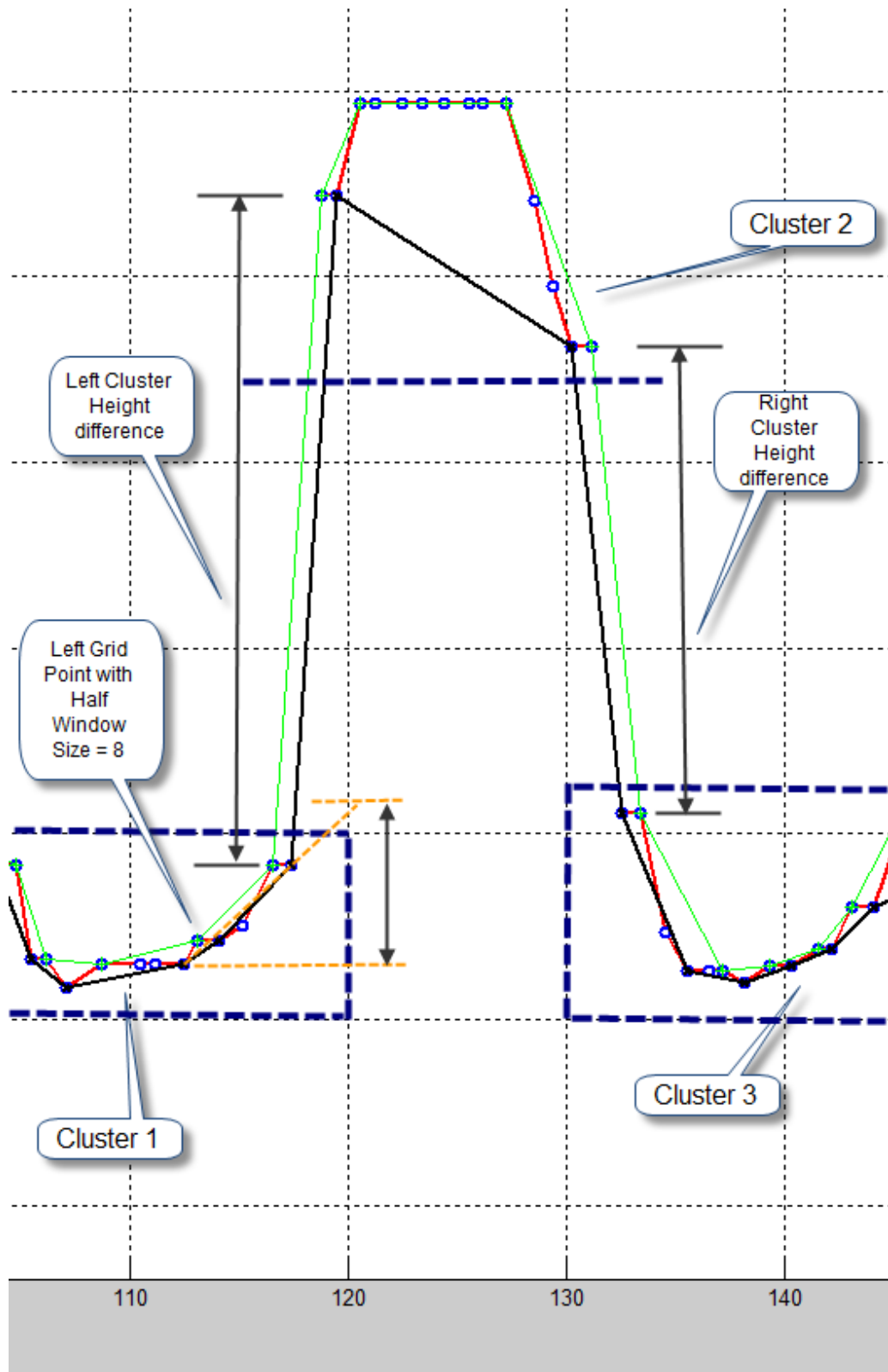
Figure 6-4 Threshold estimation of different clusters coverage

From the trend property, it shows which minima or maxima range the point falls into, because all the minima and maxima points were collected in the trend analysis. The highest and lowest minima points are used to estimate the filter threshold under a certain window size. The reasons to choose the highest and lowest minima points are as follows. First, minima points are more likely representing ground points than are maxima points. Second, the lowest minima point in a certain range is more likely representing the start of a trend. Two scenarios of the minima points' spatial relationship are shown in Figure 6-5. One scenario (1) shows that the minimum point is lower than its neighboring minima points, which means it is the minimum of three minima points. These kinds of points show a trend reverse. It is a pivot point of a surface. The second scenario (2) is that the lowest minima point in a certain range is only lower than one of its neighboring minima. This implies this minimum point is on an uptrend or downtrend surface. The third scenario is that the three consecutive minima points have the same elevation, which might imply this range of terrain has no significant trend change. Since the criteria of identifying minima in the proposed method exclude the minima points with the same elevation as that of its neighboring minima, this scenario does not need to be considered. All of these properties will be used to estimate the surface slope variation under different filtering window sizes.
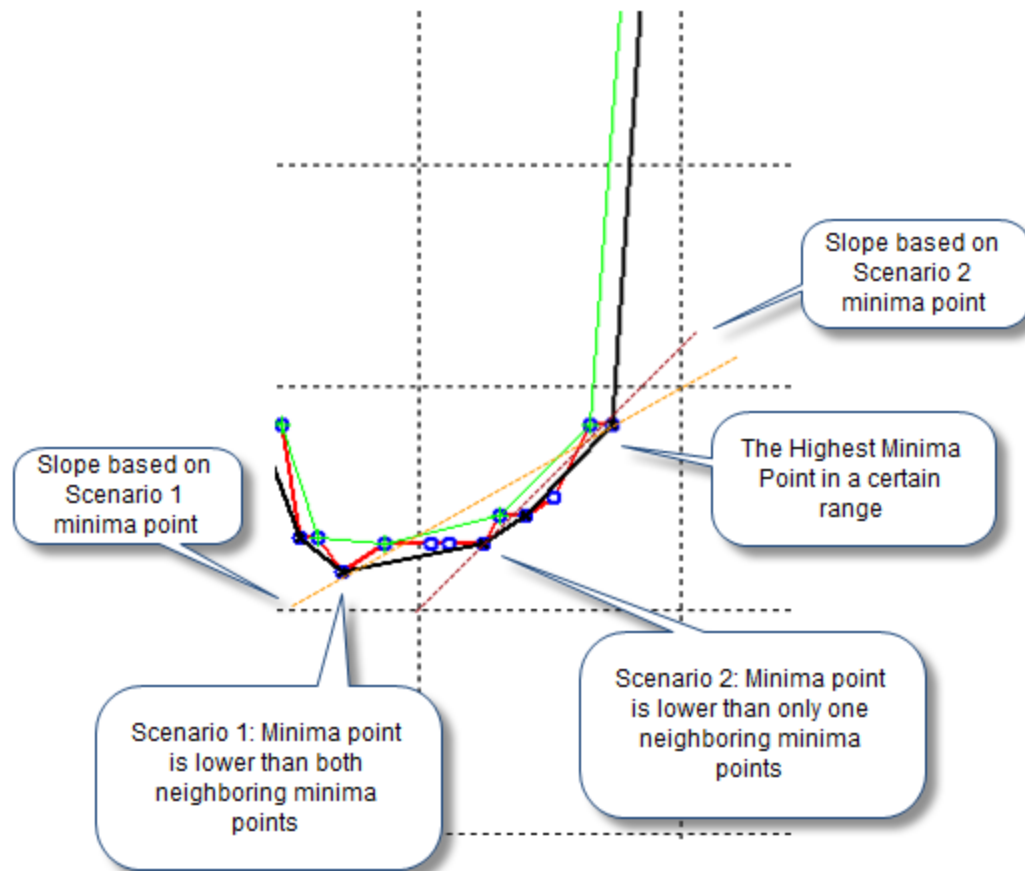
Figure 6-5 Different minima and slope of terrain surface

From the current point's cluster, two elevation difference values can be acquired; one is the elevation difference over the previous cluster, which was represented by *preDiff*, and the other is the elevation difference over the next cluster, which was represented by *postDiff*. The threshold estimation procedure is as follows:

1. Set the cluster difference value *clusterDiff* as the smaller absolute value of *preDiff* and *postDiff* of the current point's cluster. If the cluster is the first or the last cluster in the collection, there would be only one side cluster elevation difference with a non-

zero value. The other one is defined as zero. For example, the first cluster's *preDiff* would be zero, because there is no cluster before it, and the last cluster's *postDiff* would be zero, because there is no cluster after it.

2. Find the tight minima ranges between the left window bounding point and right window bounding point (shown in Figure 6-6). The tight minima range of a point refers to the range made by two consecutive minima points that covers the point. If the point resides between one minima point and one ending point (the first or last point) of the row or column, the range would be made by one ending point (the first or last point) of the row or column, and one minima point. The minima ranges are shown in Figure 6-6.

3. By finding left and right bounding points' tight minima range, the full window minima range can be detected, which is formed by the left minima of the left bounding point tight minima range and the right minima of the right bounding point tight minima range.

4. After finding the full window minima range, the lowest and highest minima points in the full coverage range can be found. Further, the elevation difference of these two points can be calculated, and so can the slope of between these two points. By multiplying the half window size with this slope, a trend elevation increase value *trendLift* can be calculated.

5. By comparing *trendLift* and *clusterDiff*, the smaller value can be used as the filtering threshold value *thresh*.

6. If the filtering threshold *thresh* calculated in step 5 is less than *clusterThresh,* which is the threshold for separating clusters, let the threshold value be *clusterThresh*.
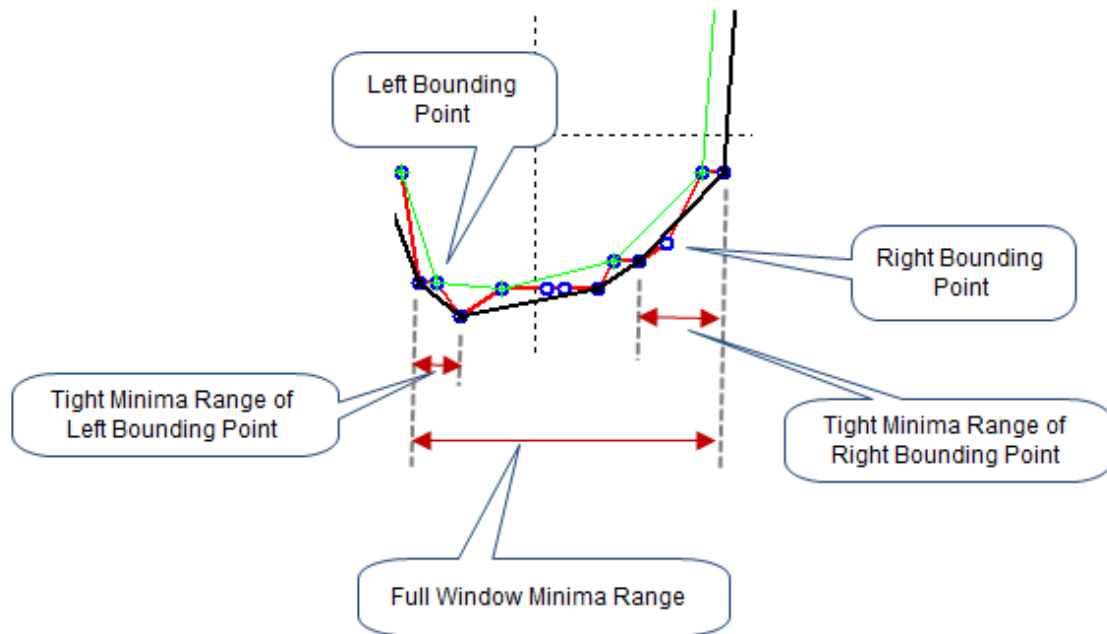
Figure 6-6 Tight minima range and full window minima range

The second scenario of the current point and bounding points is that they are in the same cluster. The threshold estimation procedure is similar to steps 2 and 3 in the previous scenario:

1. Find the tight minima range between the left window bounding point and right window bounding point.

2. By finding left and right bounding points' minima range, the full window minima range can be determined.

3. After finding the full window minima range, the lowest and highest minima points in the full coverage range can be found. Further, the elevation difference *minimaDiff* of these two points can be calculated.

4. If the *minimaDiff* is less than *clusterThresh,* which is the threshold for separating clusters, let the filtering threshold value *thresh* be *clusterThresh*.

In the two scenarios, the minima points were both used to estimate the terrain slope, because minima points more likely represent the terrain surface than the maxima points; and most of the local minima points are from the ground surface or the lower surface of non-ground objects. If using minima to estimate slope of a certain range of data, it would be more accurate to reflect the terrain surface characteristics than to use maxima points. In the threshold estimation, the lowest and highest minima points in a certain range were used to calculate the slope. This slope value reflects the local terrain characteristics and can be used for the filtering threshold estimation.

Based on the algorithm description of the proposed filter, the time complexity of the algorithm should be the same as that of the progressive morphological filter. Since the proposed filter is based on the progressive morphological filter, the major computation time for the progressive morphological filter is the *erosion* and *dilation* operations, in addition to the interpolation. The time complexity for the *open* operation is O($wN$), where $w$ is the window size of the morphological filter and $N$ is the number of grids, which is the product of the number of rows and columns. For $M$ windows, the time complexity is equal to [51].

$$O(\sum_{k=1}^{M} w_k N)$$

Since in the cluster generation procedure *MakeCluster*, the whole data set just needs to be scanned once to generate all the clusters, the time complexity of cluster generation is O($N$) [10]. In the cluster analysis procedure, the time complexity of the cluster search for each point is O($\log N$), based on a binary search. The threshold estimation has the same time complexity O($wN$) as that of the *open* operation. Therefore,

the overall time complexity of the proposed filter is the same as that of the progressive morphological filter.

The space complexity of the proposed filter is O($N$). Because the space complexities of the mark matrix for the filtering status, the threshold matrix and the clusters are all O($N$).

### 6.2.4 Adaptive Morphological Filtering Algorithm

In this section, we will demonstrate how the cluster and trend analysis works for automatically and adaptively estimating filtering parameters by combining this method with the progressive morphological filter. It can also be embedded into any other filters that need to estimate parameters based on the local terrain characteristics. The proposed adaptive morphological filtering method consists of three major components. One is the cluster analysis method, the second is the trend analysis method, and the third is the progressive morphological filtering method. Each of these three components has its own functionality. By combining them, it would make this method a more automatic and adaptive way to filter LIDAR data. From the functionality perspective, each method would help the adaptive filtering from its own feature. The cluster analysis method provides a mechanism to separate points into different groups with a discrete elevation level. Normally non-ground object points must have some elevation jump from their nearby ground surface points, thus the clusters identified by a certain elevation difference provide not only a potential separation between ground points and non-ground points, but a potential separation between different non-ground point groups as well. The non-ground object points would normally fall into different clusters from the nearby ground

surface point clusters. This feature of a cluster would help the threshold estimation by providing the trend analysis method an appropriate range to estimate the slope of the terrain.

The trend analysis method offers a way to detect the local terrain variations. By identifying the local minima and maxima, we can know the terrain trend in a certain range. We can gain much more useful information from the collections of minima and maxima, such as the slope in a certain range. By combining this with clusters information, we can obtain the non-ground object surface trend lines, because the trend lines made by the maxima and minima in the non-ground object surface clusters would form a tight boundary range, which would be very helpful to estimate the surface trend. This is a very important result from the trend analysis. This feature can be extended for multiple applications.

Finally, the progressive morphological filter is the filtering method we selected to work together with cluster and trend analysis components. It can be replaced by any other filters that also need the estimation of parameters based on the terrain surface trend. The reason why we choose the progressive morphological filter is that it uses increasing window sizes to filter the non-ground objects, and it will retain the terrain shape for larger objects until the filtering window size reaches the same level of these objects in size. This feature collaborates with cluster and trend analysis methods to make the clusters and trends obtained in each filtering step reflect the terrain characteristics under different filtering window sizes.

The structure of the adaptive morphological filtering algorithm is based on the progressive morphological filtering. The input data has to be gridded with a certain grid

size, and the empty grids has to be interpolated with some interpolation method, such as the nearest neighboring point. During each step of progressive morphological filtering, a moving window with a certain size will be used in the *open* operation, which includes *erosion* and *dilation* operations. After that operation, the elevation difference will be calculated between the elevation of pre-operation and post-operation. In the proposed method, the filter embedded the cluster and trend analysis mechanism, which will help estimate the window filtering threshold adaptively in each filtering step.

Algorithm description

AdaptiveMorphFilterFile Procedure

INPUT:

Original LIDAR data file.

OUTPUT:

Ground points file.

1. Grid Input file
2. Interpolate empty grid
3. Use AdaptiveMorphFilter procedure to process grid data
4. Output ground points to file

AdaptiveMorphFilter Procedure

INPUT:

1. Gridded and interpolated LIDAR data: *Z*
2. Filtering steps: *steps*
3. Initial window size to start Adaptive Morphological Filtering: *initThreshBin*
4. Cluster Threshold for separating different clusters: *clusterThresh*

OUTPUT:

Mark Matrix for Ground points: *mark*.

1.  Create and initialize *mark* matrix for each grid of point
2.  for k = 1 to *steps*
3.   for each direction (by row and by column)
4.    for each *i*th row/column in the grid
5.     zcurr ← *Zi*; // extract the *i*th row/column's z values
6.     if *window_size*(k) >= *initThreshBin* then
7.      clusters ← *MakeCluster*(zcurr, *clusterThreshold*); // Create
                    cluster array for current row/column
8.      trendArr ← *MakeTrendArray*(zcurr); // Create trend array for
                    current row/column
9.      thresholdArr ← *CalThreshArray*(clusters, *window_size*(k), trendArr);
             // Calculate the filtering threshold for each point in the row/column
10.     end if
11.     zopen ← *Morphopen*(zcurr, *window_size*(k)); // Open operation
12.     zdiff ← zcurr – zopen;
13.     if *window_size*(k) < *initThreshBin* then
14.      thresholdArr ← threshold(k); // Set each threshold value in
                    thresholdArr as threshold(k)
15.     else
16.     if zdiff(i, j) > thresholdArr(j) then
17.      *mark*(i, j) ← *window_size*(k); // Update mark
18.     end if
19.     *Zi* ← zopen; // Set the *i*th row/column's z values
20.    end of each row/column
21.   end of for loop of each direction
22.  end of for loop of all steps


In each filtering loop, when the filtering window size is greater than or equal to the predefined initial adaptive filtering window size *initThreshBin* (in Line 6), the cluster and trend analysis methods is involved in the filtering loop. The adaptive threshold estimation results are used in the classification of points (in Line 16). While if the filtering window size is less than *initThreshBin* (in Line 13), the constant filtering threshold would be used. This is good for the smaller window size, such as 1 or 2, which is not necessary to carry out the adaptive threshold estimation, while users can still involve the adaptive threshold estimation in for all the window sizes by simply set *initThreshBin* as 0.

The *MakeClusters* procedure (in Line 7) described in [10] creates a cluster array for each row or column based on the predefined cluster threshold *clusterThresh*. In this way, the data points in a row or column can be separated into a collection of clusters.

The *MakeTrendArray* procedure (in Line 8) is used to collect all the minima and maxima points from the unfiltered points in a row or column, which reflects the local trend variation information. This trend information is combined with clusters information and helps the *CalThreshArray* procedure estimate the filtering threshold. Therefore, in the *MakeTrendArray* procedure, the entire row or column would be scanned to find the minima and maxima points from all the unfiltered points.

After getting clusters and minima information (in Line 7 and Line 8), the *CalThreshArray* procedure was called to estimate the filtering threshold for each point in a row or column. The essential idea of filtering threshold estimation is by checking the relationship between the clusters of the current point and the clusters of its bounding points in the filtering window, and analyzing their local trend information from local minima, to estimate the local slope of the terrain surface where the current point is located. The local terrain slope of the current point would be used to estimate the filtering threshold.

CalThreshArray Procedure

INPUT:

1. A row or column data points: *Z*
2. Half filtering windows size: *binSize*
3. Cluster threshold for current filtering window size: *clusterThresh*
4. Current row or column clusters collection: *clusters*
5. Current row or column minima collection: *trendArray*

OUTPUT:

An array of the filtering threshold for each unfiltered point in the row or column:

*thresholdArr*

1.  foreach unfiltered point $Z_i$ in current row or column
2.      binLeft ← $i$ – *binSize*;
3.      if binLeft < 1 then binLeft ← 1;
4.      binRight ← $i$ + *binSize*;
5.      if binRight > n then binRight ← n; // n is the number of grids in current row or column
6.      clusterNum ← *findCluster*($i$, *clusters*); // Find current point cluster number in clusters
7.      binLeftCluster ← *findCluster*(binLeft, *clusters*); // Find the cluster number of left bounding point within filtering window size
8.      binRightCluster ← *findCluster*(binRight, *clusters*); // Find the cluster number of right bounding point within filtering window size
9.      (binLMiniLeft, binRMiniRight) ← *findTrendRange*(*trendArray*, i, n); // Find the left and right bounding points minima range (binLMiniLeft, binRMiniRight)
10.     (lowMinima, highMinima) ← *findLowHighPt*(Z, trendArray, binLMiniLeft, binRMiniRight); // Find the lowest and highest minima points values lowMinima and highMinima in the range (binLMiniLeft, binRMiniRight)
11.     if clusterNum, binLeftCluster, binRightCluster are the same cluster number then
12.         *thresholdArr*($i$) ← highMinima – lowMinima; // Set current point's threshold
13.     else
14.         *thresholdArr*($i$) ← Slope(highMinima, lowMinima)**binSize*;
15.     end if
16.     if *thresholdArr*($i$)< *clusterThresh*
17.         *thresholdArr*($i$) ← *clusterThresh*;
18.     end if
19.     if *thresholdArr*($i$) > maxThreshold // Compare with maximum threshold, e.g. *clusterThresh*binSize*/4
20.         *thresholdArr*($i$) ← maxThreshold;
21.     end if
22. end of for loop of current row or column
23. Output thresh array for current row or column

The *CalThreshArray* Procedure is the core part of estimating the filtering threshold. After determining the current point's bounding points (in Line 2~5), the

clusters of the current and its bounding points can be found (in Line 6~8). The corresponding minima range, which covers the left and right bounding points, can be detected (in Line 9). In Line 10, the lowest and highest minima points in the minima range are found. In Line 11, the cluster numbers of the current point and its bounding points were compared. If they are in the same cluster, the estimated threshold is set as the elevation difference between the highest and lowest minima points (in Line 12). Otherwise, the estimated threshold would be set as in Line 14. If the estimated threshold is less than the *clusterThresh,* which is used to separate the cluster, the estimated threshold is set as *clusterThresh* in Line 17. If the estimated threshold is greater than the maximum threshold, which can be calculated by the *clusterThresh*, the estimated threshold is set as the maximum threshold in Line 20. The idea of determining the maximum threshold is based on how many times *clusterThresh* elevation increases in a half filtering window size. In Line 19, an example value was given as *clusterThresh\*binSize/4*, which means in a half window size *binSize*, there is a *clusterThresh* elevation increase every 4 grids. The total elevation increases would be *clusterThresh\*binSize/4*, which can be treated as a maximum threshold. If the number of grids uses 2 or 3, it will give the maximum threshold as *clusterThresh\*binSize/2* or *clusterThresh\*binSize/3* accordingly.

The key of this adaptive filtering method is the filtering threshold estimation. The more accurately the threshold is estimated, the more accurate the filtering results will be. Although there could be many ways to calculate the threshold, it would be very challenging to find a generalized way to estimate perfectly in all the parts of a complex data set. More experiments on different kinds of terrains would help achieve a more

126

general rule of adaptive threshold estimation for all the terrain types, which can further improve the correctness of this method.

6.2.5  Discussion of Filtering Parameters

Since all the filtering methods have one or more parameters that are given by the user, the selection of parameters is very important to any filtering method. Some parameters might be very sensitive for some methods and get remarkably different results. The parameters related to our adaptive filtering method focus on the following parameters:

- Grid size,
- Filtering window sizes,
- Cluster threshold,
- Filtering directions.

These parameters are related to each other. Therefore, their selection might need to consider other parameters. We will discuss their relationships and how to choose the values under different circumstances in detail.

From the essentials of our adaptive filtering method, the cluster thresholds under various filtering window sizes provide the mechanism that can help our adaptive filter estimate the critical filtering thresholds according to the local terrain. However, the selection of cluster thresholds is not necessary to be as accurate as that of filtering thresholds in many filtering methods. Since the cluster threshold, which is for the cluster separation procedure, only assists the filtering threshold estimation, it would not be very sensitive as the filtering threshold on different terrain types. It might have some minor

influences to the filtering results, but it would not alter the results significantly. Also, we can find some general rule to select cluster threshold on certain terrain types, which would make our method more general. We will discuss the details of how to choose cluster threshold in the later paragraphs.

The grid size is a basic parameter among all the parameters because it shows the resolution of the data set, and how precisely the user would like to filter the data set as well. The grid size is often chosen as the resolution of the data set, because the data set information would be utilized completely in this way. For example, we can choose the appropriate grid size to let each grid contain one or two points. Sometimes, the grid size is chosen with a larger value because of computation time and computer hardware limitations. However, when processing a very large data set, the data set is normally split into an appropriate smaller size for processing, so that the smallest grid size can be chosen to fully use the data. In our experiment data, we normally used 1 meter as the grid size according to the data set resolution.

After the grid size was decided, the filtering window sizes can be determined by the terrain types and various non-ground objects. In our filtering method, we used increasing window sizes to remove non-ground objects iteratively. The reasons why we need to choose increasing window sizes are as follows. First, we have to remove different sizes of non-ground objects under various sizes of filtering windows. Second, the best way to remove non-ground objects is from small non-ground objects to large ones. Based on these two reasons, we can select an appropriate window sizes' series for removing various non-ground object features. In the real window size parameter, we used the number of grids to represent window size; therefore the real window size would be the

number of grids multiplied by the grid size. Also, the window size we used is actually a half window size, which represents the number of grids extended from the current grid to either the left or right sides of current grid. Therefore, the full window coverage of the current grid would be $2N+1$, where $N$ is the window size value. These values are very important to be considered on different terrain types and for various sizes of non-ground features. For example, if we choose 1 meter as the grid size, and 1 for the half filtering window size, the full coverage of filtering window would be 3 grids in size. Accordingly, the real size of the full window coverage is equal to 3 meters. This is very important for different sizes of non-ground features to select window size. For example, if we have a 10x10 square meters non-ground object, if the grid size is 1 meter, we need to choose 5 grids as the half window size to cover the whole object, because the full coverage of the filtering window size would be 11 grids, which is equal to 11 meters in this case. At the same time, if the grid size is 0.5 meters, we need to choose 10 grids as the half window size to cover the whole object, because the full coverage of the filtering window size would be 21 grids, which is 10.5 meters. Since we used the half window size as a mark value to identify when the non-ground objects were removed in each filtering step, these values are very useful and important to reflect when the non-ground objects were removed. We can further analyze these marks to acquire better filtering results. The typical window size series we used in the experiment data sets is 1, 2, 4, 8, 16, 32, and 40. The grid size of our experiment data set is normally 1 meter. The full coverage of window sizes for this series is 3, 5, 9, 17, 33, 64, and 81 meters. This window series works for various dimensions of non-ground objects. The half window size with 1, 2, and 4 (full window coverage with 3, 5, and 9 meters) focuses on the small non-ground objects,

such as small vegetation, cars, and small constructions. The half window size with 8 and 16 (full window coverage with 17, and 33 meters) deals with mid-size buildings, and parts of complex constructions, such as parts of bridges. The half window size with 40 grids (full window coverage with 81 meters) works for large buildings and complex constructions.

The filtering window series is tightly related to the sizes of non-ground object features in a data set. Small window size, such as 1, 2, and 4, is very common for any data sets. This is the case because it is necessary for small, non-ground objects, such as trees, cars, and small-size noise points. These small-size features normally exist in most of the data sets. Therefore, the small windows normally have the values of 1, 2 and 4.

The selection of the mid-size and large filtering windows is slightly tricky, because they need to comply with the sizes of non-ground features, which might be significantly different in size and elevation on various terrains. For example, the buildings on mountainous areas are relatively smaller and lower than on flat terrains. In the flat urban areas, there exist many large and high buildings, while the buildings on the mountainous areas are rarely such sizes. This influences the selection of mid-size to large filtering window size significantly. If the sizes of non-ground object features in a data set can be categorized in certain levels, it would be very helpful to choose the filtering window size under each level. For example, if the sizes of buildings in a data set are in the levels of 10, 20, and 30 meters long, we can select 5, 10 and 15 as half filtering window size if the grid size is 1 meter. Because the coverage of full window size would be 11, 21 and 31 meters in length, which can filter out these three levels of buildings, correspondingly. In addition, if we know the limit of the largest non-ground object

feature's size, it would help the filter set the maximum window size, which would reduce the filtering errors and computation time. For example, if we know the maximum non-ground object feature's size in a data set is less than 40 meters in length, we can select 20 as the maximum half filtering window size if the grid size is 1 meter. Because the coverage of full window size would be 41 meters in length, it can filter out the largest non-ground object features, and stop the filtering process.

According to the window size series, we can define the cluster threshold based on the non-ground features of any sizes in the data sets. The major purpose of the cluster threshold is to separate potential non-ground object features from local ground features based on the elevation difference under certain window sizes. Therefore, the cluster threshold is not necessarily very accurate. As long as the cluster threshold value can separate different levels of data points, it would help the filtering threshold estimation procedure to calculate the filtering threshold more precisely according to the local terrain variations. The selection of cluster threshold should comply with the elevation difference of each window size level. For example, if the data set contains relatively small and low vegetation, we should choose a relatively smaller cluster threshold, which can separate low elevation non-ground features from terrain surfaces into more clusters. It would help filter out more non-ground features. If the data set contains relatively high constructions in a large size, we can adjust the cluster threshold higher for large filtering window sizes, because it would more effectively separate non-ground objects from terrain surfaces.

The Table 6-1 shows two typical sets of window size and cluster threshold parameter series. In the second row of parameter series, the cluster thresholds under large window sizes (16, 32, and 40) are greater than the first series. This is better parameters

131

setting for the terrain, which has large non-ground features with a greater elevation gap. Additionally, the cluster threshold under a small window size (e.g. 1) is smaller than the first series. This is good for filtering terrain that has small non-ground objects with relatively low elevation.

Table 6-1 Filtering half window size series and cluster threshold series

| Half Window Size Series | 1 | 2 | 4 | 8 | 16 | 32 | 40 |
|---|---|---|---|---|---|---|---|
| Cluster Threshold Series 1 | 0.5 | 0.5 | 1.0 | 1.0 | 1.0 | 2.0 | 2.0 |
| Cluster Threshold Series 2 | 0.3 | 0.5 | 1.0 | 1.0 | 1.0 | 3.0 | 4.0 |

Another parameter which would affect the filtering results is the filtering direction. Our filtering method is based on grids of the data, which are stored in a matrix-like structure. The filtering directions refer to the moving direction of the filtering windows. The typical directions are row and column directions. When choosing different filtering directions, it would make the selections of other parameters slightly different. A simple example is to filter a rectangular building. If we choose the filtering window moving along the shorter edge direction, we can use a smaller filtering window size to filter out the object, while if we filter it along the longer edge direction, we need a larger filtering window size. This is the major reason for us to choose the appropriate directions. Figure 6-7 shows the basic idea of how the filtering direction would influence the selection of the filtering window size.
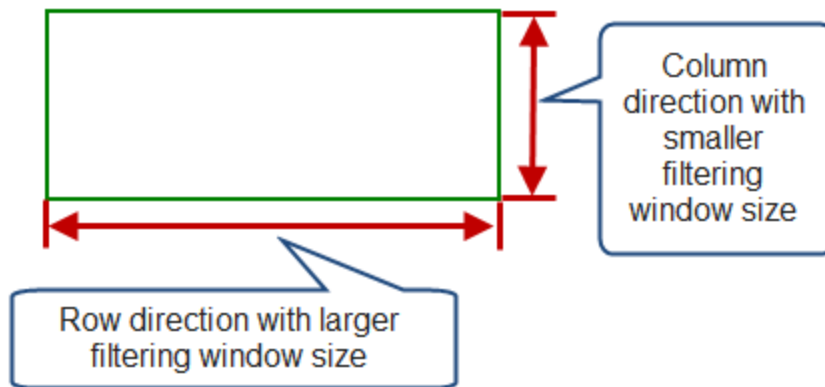
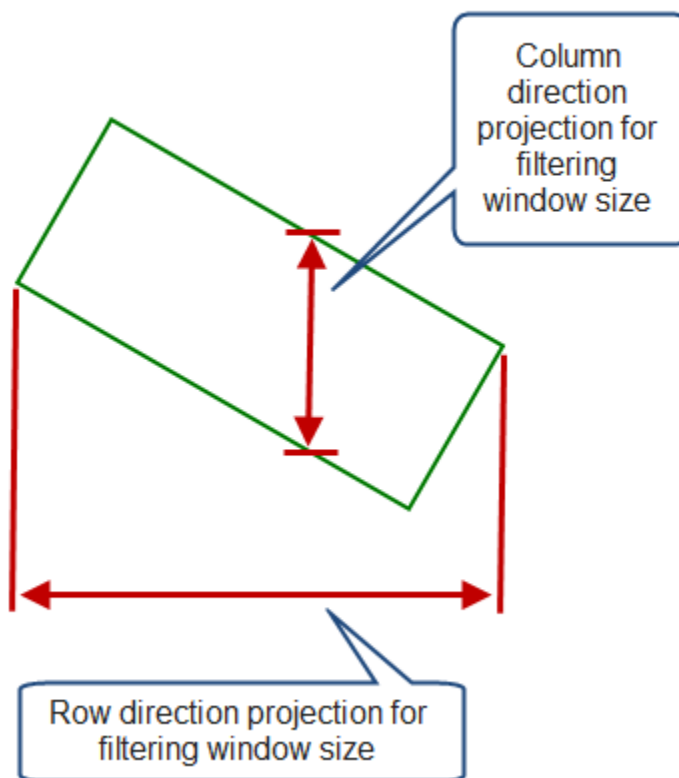Figure 6-7 Filtering directions of rectangular shape objects



Figure 6-8 Non-orthogonal object shape filtering window size

However, if the rectangular shape object is not orthogonal with the grid matrix, which means the shape has some angle with the grid matrix's row or column, it would influence the selection of filtering window sizes as well. The maximum filtering window size to filter out the rectangular shape object relies on the projection size on the row and column directions. Figure 6-8 illustrates how the non-ground object's position influences the selection of filtering window sizes on the row and column directions. Small and mid-size non-ground object features would not bring so much trouble to the selection of filtering window sizes, while large size non-ground object features might bring trouble to it. The complex bridge is an example of this scenario. Because it normally has very large extension on a certain direction, also it is connecting with the terrain surface and rising smoothly. Therefore, the selection of filtering direction is determined by the purposes of filtering jobs, or the requirements of the filtering results. For instance, when we process a complex bridge, we have to make decision on what we need to filter out based on the purposes. Because the middle part of a complex bridge is normally above the ground and the ends of it are normally connected to ground surfaces and rise smoothly, if we need to remove the middle parts of the bridge, which are over the ground surface, but not the ending parts, which are connected to the ground surfaces, we had better choose the filtering direction with the shorter extent. However, if we need to keep the whole bridge intact as much as possible, we should choose a filtering direction with the longer extent. As long as the filtering window size is large enough to filter out the whole bridge, it would be possible to keep more points on the bridge.

For some complex non-ground object features, we might need more than one direction of filtering. The results from different filtering directions would provide more

information to help decide what should be filtered out according to different purposes. One solution is that we can process the whole data in row direction and then in column direction, or vice versa. Since we used a mark matrix to represent the filtering statuses of all the grids, we would get two separate mark matrices for two directions' filtering results. We can further analyze these two matrices to achieve final results. By filtering a data set from row or column direction, we can choose different sets of parameters, such as window series. The filtering window sizes' series are determined by the position and shape of non-ground features. The major factor is the projection sizes of the objects on the horizontal (row) and vertical (column) directions. There are many cases that the major terrain or non-ground features are not orthogonal with the row or column direction, which means the major directions of the terrain or non-ground features have some angle with the horizontal and vertical directions. This would affect the accuracy of filtering results in some cases. Normally, if the terrain or non-ground features are very large and there are many objects on them, it might affect the filtering results.

To utilize the advantage of multiple directions filtering, we can carry out the filtering on each direction and analyze the mark matrix of each filtering direction to get the final result. We can combine the results on two directions. The typical way to combine the results is to compare the mark matrices on different directions and set those unfiltered grids in one matrix as filtered grids according to other directions filtering results' matrices. For example, we can filter the row direction with the maximum window size of 40 and then the column direction with the maximum window size of 16. Finally, we can set those unfiltered grids in the row direction but filtered in the column direction

as filtered grids. We can also reset some filtered grids in one direction but unfiltered in other directions back to the unfiltered status according to our filtering needs.

## 6.2.6   Result Analysis

Filtering experiments have been carried out in many data sets from different terrain types. The first data set is from California. The testing data set is on a mountainous area, which has undulating terrain with different sizes of non-ground objects, such as trees, vegetation, and buildings, etc. The data set covers a 200 x 200 square meter area. One meter was used as the cell size to grid the data set, which would give a 200 x 200 grid mesh.
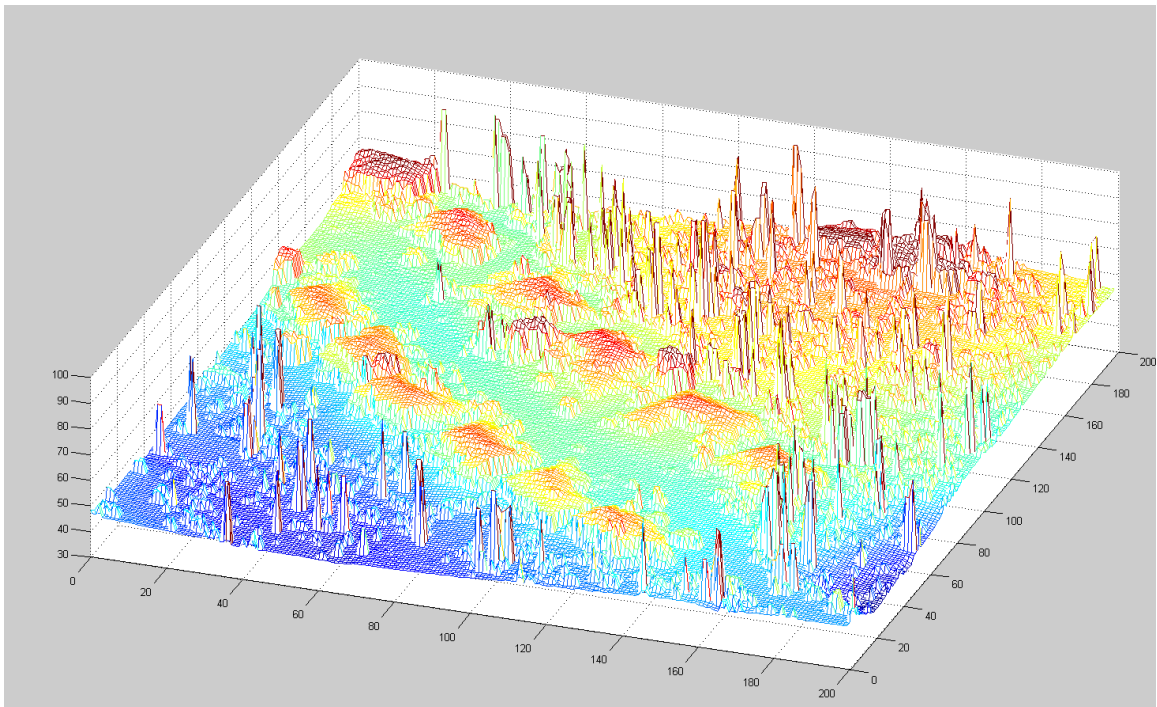


Figure 6-9 3-D mesh of original data set 01

Progressive Morphological Filter Result 01          Progressive Morphological Filter Result 02
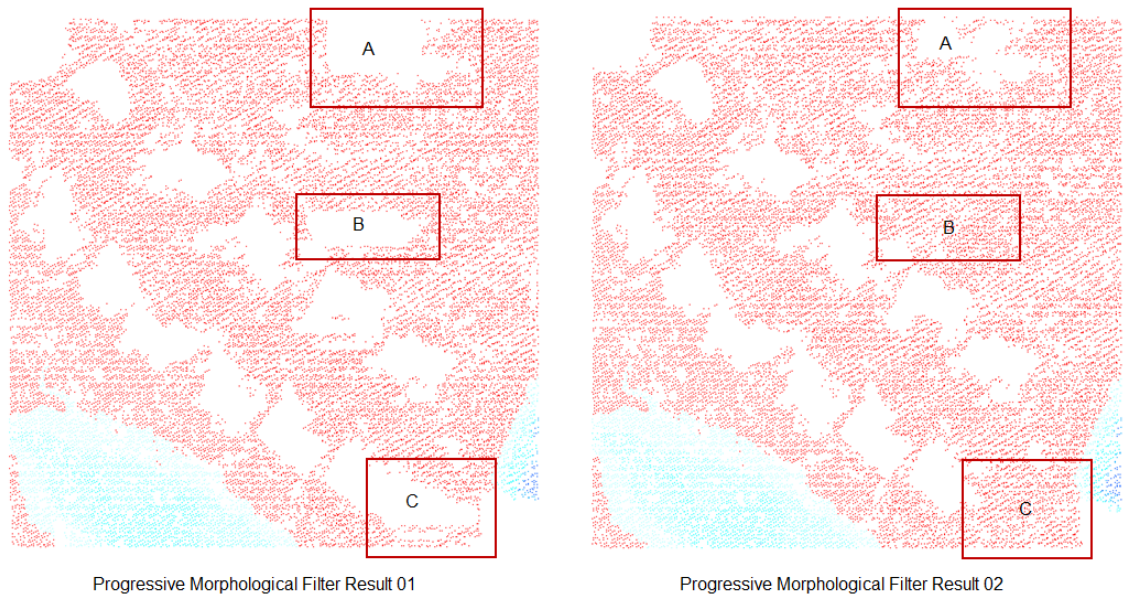
Figure 6-10 PM filter results of different threshold parameters for data set 1

The original data set's 3-D mesh diagram was shown in Figure 6-9. Figure 6-10 shows two results of progressive morphological filter with different filtering threshold parameters. The same series of half filtering window size was used for these two results. The series of half window size is 1, 2, 4, 8, 16, and 32. The thresholds of each window size for the two results are listed in Table 6-2.

Table 6-2 Thresholds of filtering results of data set 01

| Window Size | 1 | 2 | 4 | 8 | 16 | 32 |
| --- | --- | --- | --- | --- | --- | --- |
| Threshold of Result 01 | 0.60 | 0.64 | 0.68 | 0.76 | 0.92 | 1.24 |
| Threshold of Result 02 | 0.40 | 0.48 | 0.56 | 1.00 | 1.60 | 2.00 |

From the filtering results, it was known that the selection of threshold would affect the filtering results significantly. The rectangle areas marked with A, B, and C in

137

the two results show remarkable difference because of the filtering threshold. Since the constant filtering threshold was used under each window size in the progressive morphological filter, it would not suit all the parts of the data set. Better selection of filtering threshold would achieve more ideal results. The constant threshold value under each window size is the intrinsic cause of poor filtering results. The proposed adaptive method's goal is to prevent this problem and reduce the human efforts of choosing the critical threshold parameters. Figure 6-11 shows the adaptive morphological filter results. The rectangle areas marked with A, B and C demonstrate that the adaptive filtering method performs better than progressive morphological filter, because it would adaptively select the filtering threshold for each point according to the local terrain variation. The cluster thresholds used in the adaptive morphological filter were a more generalized set of values under each window size for all the terrain types. Since the cluster threshold is just used to distinguish potential continuous surface, it does not need to be very accurate, as long as it can separate terrain surfaces into different layers. Furthermore, the critical filtering threshold is calculated more accurately during filtering, which makes the cluster threshold series general for many data sets with different terrain types. The selection of a cluster threshold is generally based on the filtering window sizes and non-ground features' sizes. Under small half window sizes, such as 1, 2, 4 grids, the full window size is less than 10 grids. A smaller cluster threshold can be used. As window size gets larger, a greater cluster threshold is used to separate large-scale surfaces. The commonly used rule of cluster threshold selection in our experiments is to choose a slope between the cluster threshold and full window size less than 0.15. The local terrain trend is not very clear under small half window size, such as 1 and 2 grids, because there

138

are few grid points to form the trend. Therefore, it is not very critical to use adaptive threshold estimation under small filtering window sizes. Constant value is good enough for small window sizes. In our experiments, adaptive threshold estimation is normally involved when the half window size is greater than 2. When the half window size is 1 or 2, constant filtering threshold values are normally used. As in Table 6-3, when the half window sizes are 1 and 2 meters based on 1 meter grid size, constant filtering threshold values are 0.3 and 0.5 meters respectively, which means the adaptive filtering threshold estimation is involved after the half window size increases to 4 meters. These constant threshold values can be used in other experiments as well.

Table 6-3 Adaptive filtering threshold of data set 01

| Window Size | 1 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| Filtering Threshold | 0.3 | 0.5 | | | | |
| Cluster Threshold | | | 1 | 1 | 2 | 2 |

The second data set is from Florida International University's campus, in Miami, Florida. The testing data set is a flat terrain with different size non-ground objects, such as trees, vegetation, buildings, etc. The data set covers a 540 x 540 square meter area. We used 1 meter as the cell size to grid the data set, which would give a 540 x 540 grid mesh.
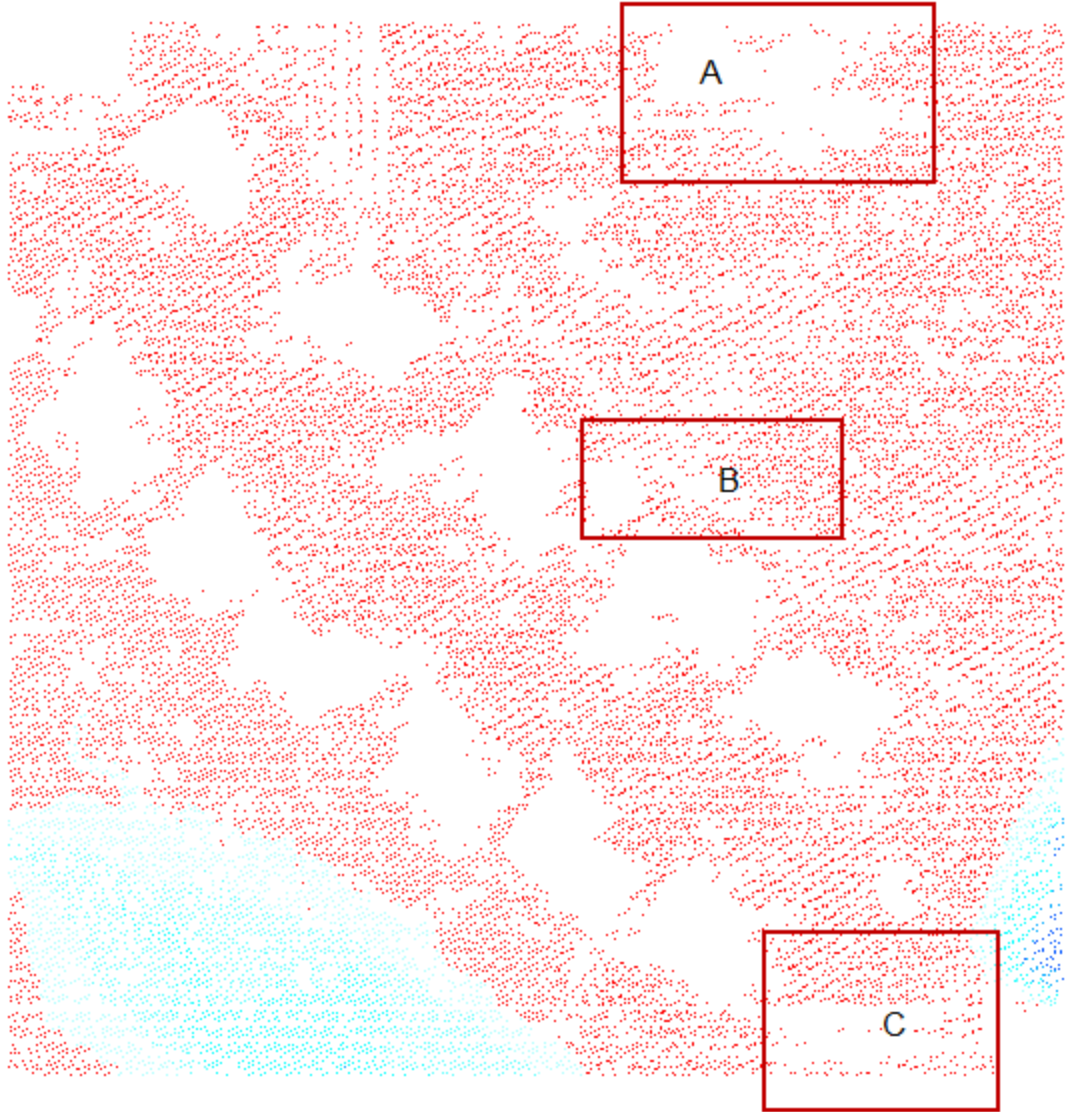
Figure 6-11 Adaptive morphological filter result of data set 1

The original data set's 3-D mesh diagram was shown in Figure 6-12. Figure 6-13 shows two results of progressive morphological filter with different filtering threshold parameters. We used the same series of filtering half window size for these two results. The series of half window size are 1, 2, 4, 8, 16, 32 and 64. The thresholds of each window size for the two results are listed in Table 6-4.
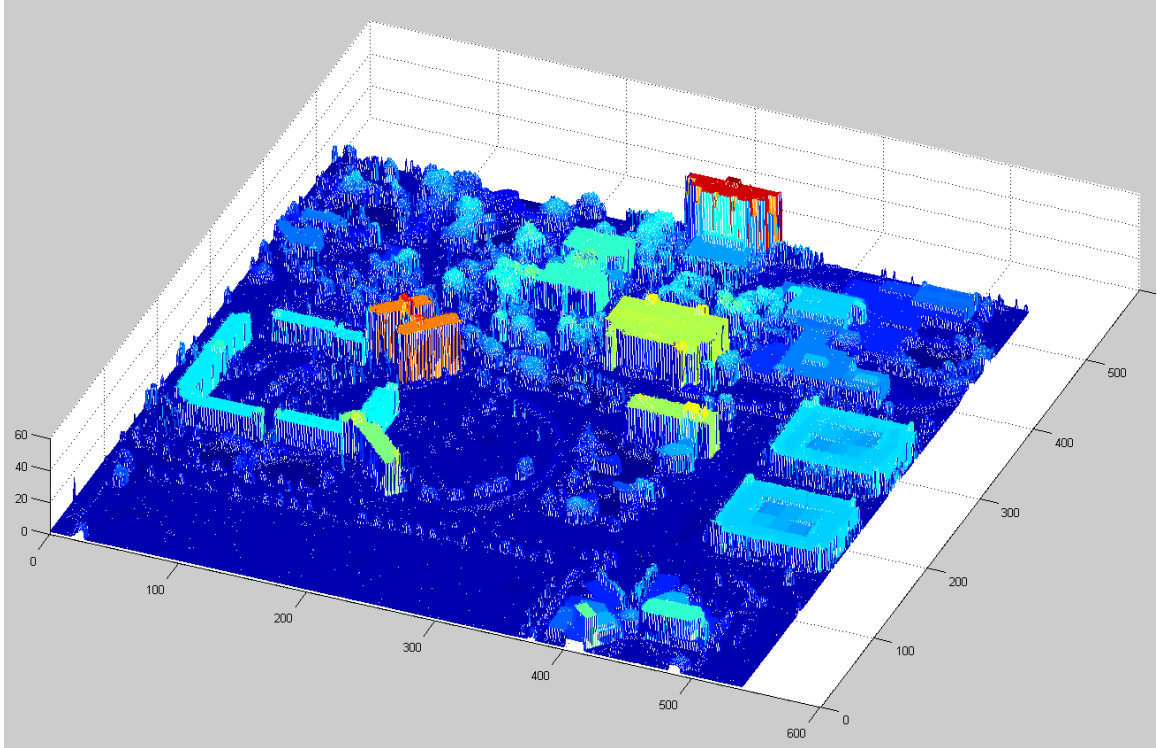
Figure 6-12 3-D mesh of original data set 02

Table 6-4 Thresholds of filtering results of data set 02

| Window Size | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|---|
| Threshold of Result 01 | 0.40 | 0.48 | 0.56 | 0.72 | 1.04 | 1.68 | 2.96 |
| Threshold of Result 02 | 0.40 | 0.48 | 0.56 | 1.00 | 2.00 | 4.00 | 8.00 |

Progressive Morphological Filter Result 01        Progressive Morphological Filter Result 02
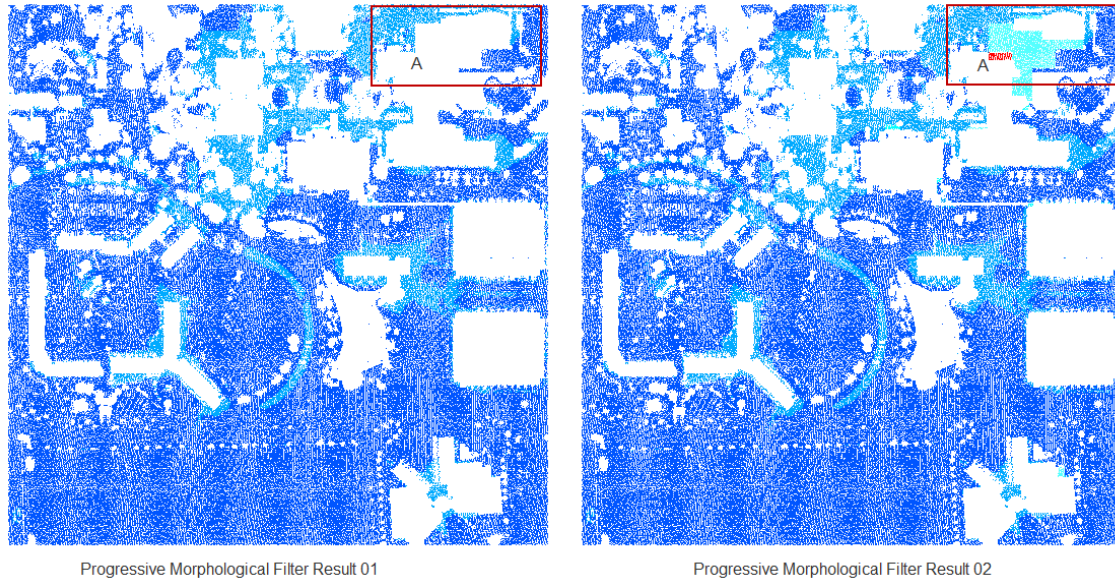
Figure 6-13 PM filter results of different threshold parameters for data set 2

From the filtering results, we learned that the selection of thresholds would affect the filtering results moderately. Only the rectangle marked with A in the two results shows a remarkable difference because of the filtering threshold. Since the progressive morphological filter was very effective in the flat terrain, it would not make a significant difference on filtering major non-ground objects. The only scenario in which the filter would make ambiguous results is when the non-ground object's height is very close to the ground surface and its size is relatively large. In that case, it would be more difficult for the filter to separate them. This kind of result difference was caused by the intrinsic terrain shape. Therefore, it would be very hard for users to use any filtering methods without additional information, except three-dimensional data. Figure 6-14 shows the adaptive morphological filter results; the rectangle areas marked with A, B show the difference from the progressive morphological filter. In the area A, the result was between the two progressive morphological filtering results. In area B, the adaptive

142

method removed more points than the two progressive morphological filtering results. In the other area, the three results have no significant difference. This testing results show our adaptive filter works effectively as the progressive morphological filter in the flat terrain.
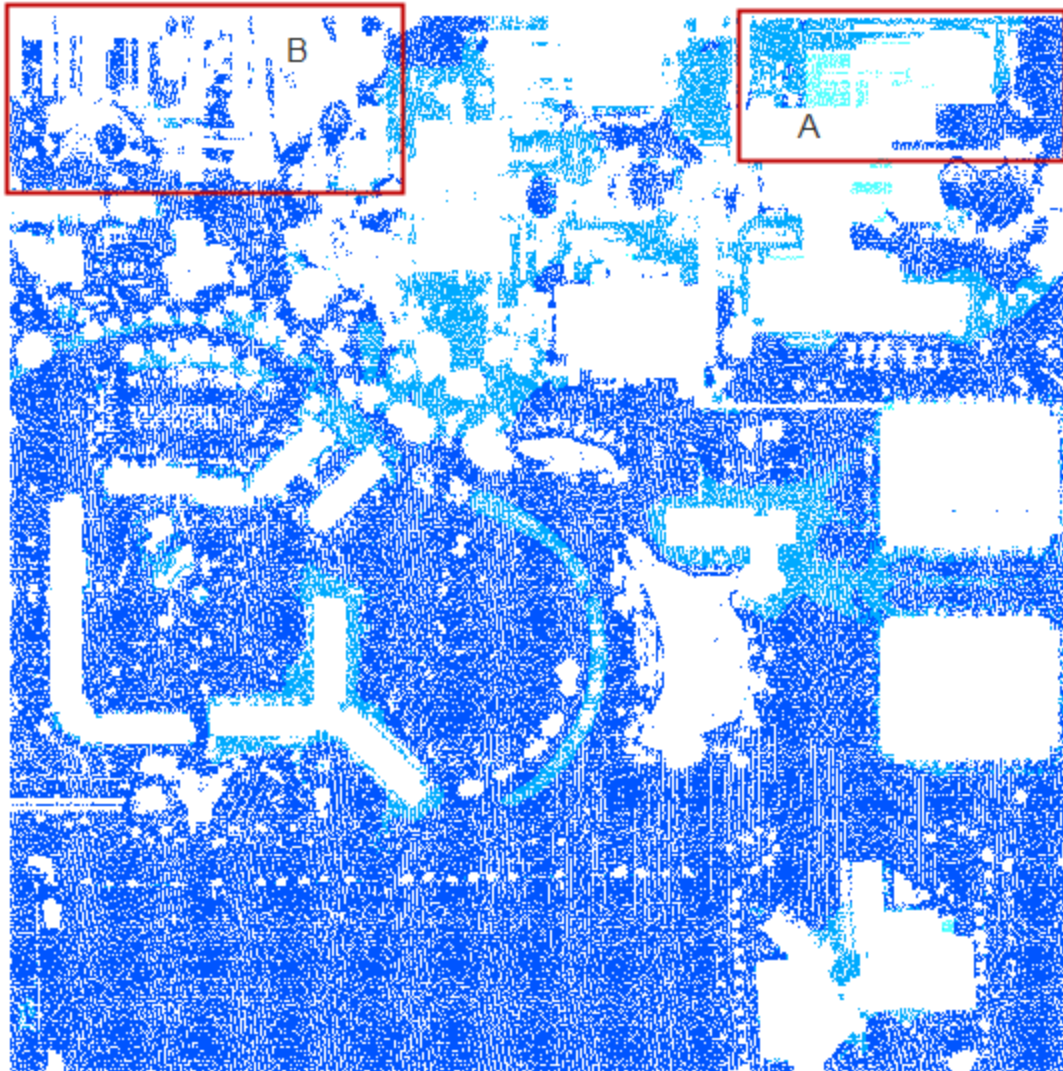


Figure 6-14 Adaptive morphological filter result of data set 2
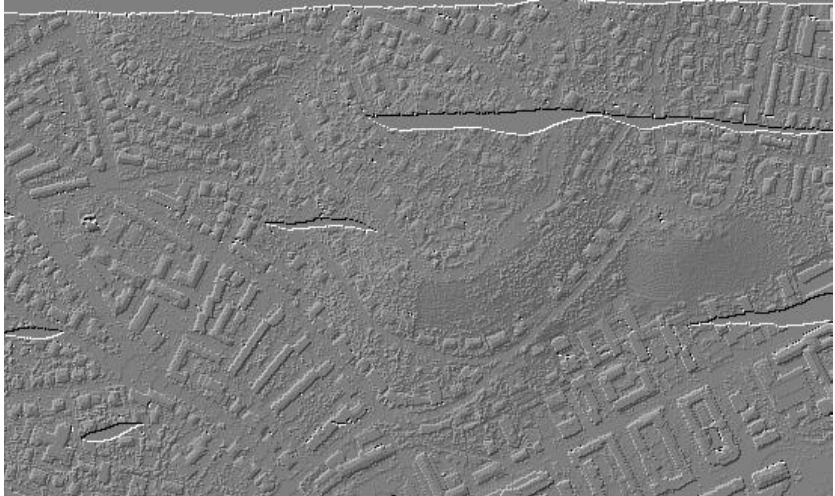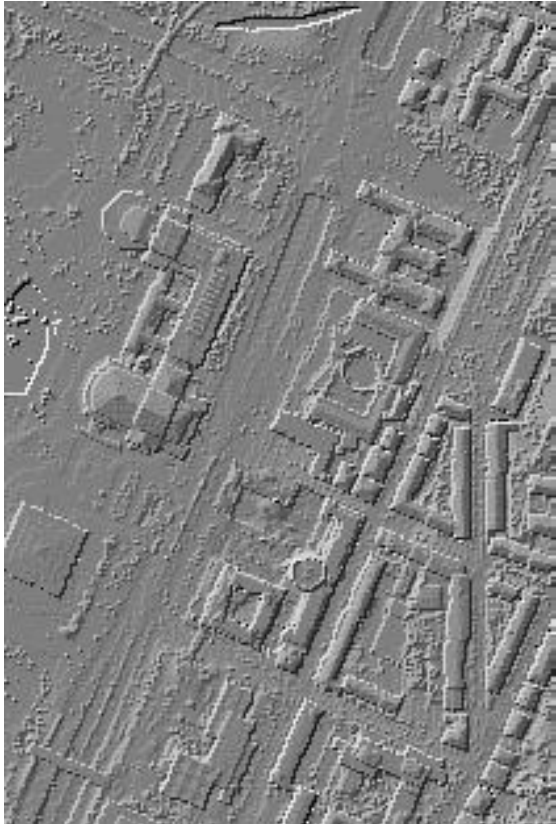
6.3 Experiments on ISPRS Testing Data Set

Since many ground filters nowadays are developed for particular terrain types, and are tested on relatively small study sites, it is very difficult to compare all the methods on various terrain types. Among many study data sets, the International Society for Photogrammetry and Remote Sensing (ISPRS) provided a collection of standardized LIDAR and ground reference data sets. These LIDAR data sets include many different terrain types and various non-ground object features. Seven testing sites and their reference ground data sets are commonly used for filtering methods to test their performances by comparing the filtering results with reference data.

Sithole and Vosselman [41] did a complete test with eight ground filtering algorithms on twelve ISPRS datasets. The results showed that most filters performed well on relatively flat and smooth terrains, but that all have noticeable errors in certain terrain types. These results demonstrated that it is very difficult to find a method with a single means to filter all the terrain types. To make a method have good performance on all the terrains, the filter has to utilize the advantage of some methods and combine them to achieve ultimate results. To test the proposed algorithm thoroughly, complete processing experiments on the ISPRS data sets have been accomplished.

The shade relief maps of seven testing sites are listed below [41]:

Table 6-5 ISPRS site 1-7 shade relief map

| Site 1 |  |
| Site 2 |  |

| Site 3 |  |
| --- | --- |
| Site 4 |  |

| Site 5 |  |
| Site 6 |  |

147

| Site 7 |  |

From the seven data sets, 15 representative samples where extracted. The reason for the choice of the samples is given in the table below [41]:

Table 6-6 ISPRS site 1-7 reference sample data set terrain features

| Site 1 | Sample 11 | Vegetation and buildings on steep slopes |
| | Sample 12 | Small objects (cars) |
| Site 2 | Sample 21 | Narrow bridge |
| | Sample 22 | Bridge (South west)/ Gangway (North East) |
| | Sample 23 | Complex buildings, Large buildings, Disconnected terrain |
| | Sample 24 | Ramp |
| Site 3 | Sample 31 | Disconnected terrain, Low point, Low point influence |
| Site 4 | Sample 41 | Clump of low points (Multi-path error) |
| | Sample 42 | Elongated objects, Low (objects) and high-frequency variation in the landscape |
| Site 5 | Sample 51 | Vegetation on slope |
| | Sample 52 | Low vegetation, Discontinuity – sharp ridge |
| | Sample 53 | Discontinuity preservation |
| | Sample 54 | Low resolution buildings |
| Site 6 | Sample 61 | Discontinuity – sharp ridge, ditches |
| Site 7 | Sample 71 | Bridge, Discontinuity – preservation |

The complete processing experiments have been carried out on the fifteen data sets, and the error analysis has been done in terms of type I (commission error), type II (omission error) and total errors as in Sithole and Vosselman [41]. The filtering process results would separate bare-earth points and non-ground object points. The error analysis

focuses on the accuracy of classification of these two types of points. The error types were categorized into type I, type II and total errors. Type I error is also called omission error, which is caused by the bare earth points that have been incorrectly identified as non-ground object points. If we use $a$ to represent the number of bare earth points that have been correctly identified, and use $b$ to represent the number of bare earth points that have been incorrectly identified as non-ground object points, the type I error can be calculated as follows:

Type I error $= b / (a + b)$

Type II error is also called commission error, which is caused by the non-ground object points that have been incorrectly identified as bare earth points. If we use $c$ to represent the number of non-ground object points that have been incorrectly identified as bare earth points, and use $d$ to represent the number of non-ground object points that have been correctly identified, the type II error can be calculated as follows:

Type II error $= c / (c + d)$

The total error can be calculated as follows:

The total error $= (b + c) / (a + b + c + d)$

Mongus and Žalik [30] proposed a parameter-free filtering algorithm and processed the fifteen data sets. The results show it can achieve less average error when compared with TerraScan's processing results.

Table 6-7 Accuracy comparison between the parameter-free algorithm and TerraScan® on ISPRS benchmark datasets

| Dataset | TerraScan® | | | The Parameter-free method | | |
|---|---|---|---|---|---|---|
| | Total error (%) | Type I error (%) | Type II error (%) | Total error (%) | Type I error (%) | Type II error (%) |
| samp11 | 16.14 | 26.66 | 2.00 | 11.01 | 7.32 | 15.98 |
| samp12 | 11.55 | 21.49 | 1.12 | 5.17 | 4.23 | 6.15 |
| samp21 | 11.56 | 14.30 | 1.95 | 1.98 | 0.01 | 8.87 |
| samp22 | 10.78 | 14.51 | 2.56 | 6.56 | 4.97 | 10.09 |
| samp23 | 8.01 | 12.92 | 2.54 | 5.83 | 4.38 | 7.45 |
| samp24 | 12.97 | 16.38 | 3.98 | 7.98 | 5.69 | 14.04 |
| samp31 | 4.85 | 8.36 | 8.97 | 3.34 | 0.21 | 7.00 |
| samp41 | 13.15 | 25.10 | 0.74 | 3.71 | 3.39 | 4.03 |
| samp42 | 2.55 | 8.00 | 1.39 | 5.72 | 0.06 | 8.06 |
| samp51 | 1.31 | 0.41 | 0.29 | 2.59 | 0.35 | 10.60 |
| samp52 | 5.38 | 4.72 | 4.52 | 7.11 | 6.61 | 11.43 |
| samp53 | 4.02 | 3.62 | 11.01 | 8.52 | 8.39 | 11.59 |
| samp54 | 2.30 | 2.49 | 13.68 | 6.73 | 1.18 | 11.50 |
| samp61 | 1.71 | 1.60 | 4.81 | 4.85 | 3.23 | 4.85 |
| samp71 | 1.90 | 1.69 | 3.56 | 3.14 | 2.37 | 9.15 |
| **Avg.** | **7.20** | **10.82** | **3.75** | **5.62** | **3.49** | **9.39** |

We used the proposed algorithm to process the fifteen data sets with multiple sets of parameters. The results show the algorithm can achieve less overall average error than TerraScan's and parameter-free algorithm's results.

The first set of filtering parameters is shown in the following table:

Table 6-8 Adaptive filtering parameter set 01

| Window Size | 1 | 2 | 4 | 8 | 12 | 16 | 20 |
|---|---|---|---|---|---|---|---|
| Cluster Threshold | 0.3 | 0.5 | 1 | 1 | 1 | 2 | 2 |
| Row Direction | x | x | x | x | x | x | x |
| Column Direction | x | x | x | x | x | x | |

Table 6-9 The comparison of terrain filtering accuracy between the parameter-free algorithm and the proposed algorithm on ISPRS benchmark datasets.

| Dataset | The Parameter-free method | | | The proposed method | | |
|---|---|---|---|---|---|---|
| | Total error (%) | Type I error (%) | Type II error (%) | Total error (%) | Type I error (%) | Type II error (%) |
| samp11 | 11.01 | 7.32 | 15.98 | 9.98 | 11.03 | 8.47 |
| samp12 | 5.17 | 4.23 | 6.15 | 3.28 | 2.65 | 4.01 |
| samp21 | 1.98 | 0.01 | 8.87 | 1.96 | 1.17 | 4.84 |
| samp22 | 6.56 | 4.97 | 10.09 | 6.7 | 4.14 | 13.37 |
| samp23 | 5.83 | 4.38 | 7.45 | 7.41 | 7.52 | 7.27 |
| samp24 | 7.98 | 5.69 | 14.04 | 5.41 | 4.71 | 7.21 |
| samp31 | 3.34 | 0.21 | 7.00 | 2.31 | 0.21 | 5.52 |
| samp41 | 3.71 | 3.39 | 4.03 | 6.46 | 5.39 | 7.47 |
| samp42 | 5.72 | 0.06 | 8.06 | 5.61 | 0.86 | 7.93 |
| samp51 | 2.59 | 0.35 | 10.60 | 2.7 | 0.89 | 9.26 |
| samp52 | 7.11 | 6.61 | 11.43 | 7.63 | 7.27 | 10.75 |
| samp53 | 8.52 | 8.39 | 11.59 | 8.73 | 8.81 | 6.92 |
| samp54 | 6.73 | 1.18 | 11.50 | 4.04 | 1.13 | 6.55 |
| samp61 | 4.85 | 3.23 | 4.85 | 3.26 | 3.25 | 3.63 |
| samp71 | 3.14 | 2.37 | 9.15 | 4.71 | 4.15 | 9.17 |
| **Avg.** | **5.62** | **3.49** | **9.39** | **5.35** | **4.21** | **7.49** |

In the parameter set 01, the adaptive threshold estimation procedure starts from window size 4. We used row and column directions filtering. The maximum window size on row and column direction was 20 and 16 meters. The unfiltered point in the row direction filtering result would be removed if it was filtered in the column direction.

The second set of filtering parameters is shown in the following table:

Table 6-10 Adaptive filtering parameter set 02

| Window Size | 1 | 2 | 4 | 8 | 16 | 20 | 30 |
|---|---|---|---|---|---|---|---|
| Cluster Threshold | 0.3 | 0.5 | 1 | 1 | 1 | 2 | 2 |
| Row Direction | x | x | x | x | x | x | x |
| Column Direction | x | x | x | x | x | | |

In the parameter set 02, we increased the windows size on the row direction filtering. The maximum window size on row and column direction was 30 and 16 meters. The unfiltered point in the row direction filtering results was removed if it was filtered in the column direction.

Table 6-11 Filtering results accuracy comparison between parameter sets 01 and 02

| Dataset | *The Parameter Set 01* | | | *The Parameter Set 02* | | |
|---|---|---|---|---|---|---|
| | Total error (%) | Type I error (%) | Type II error (%) | Total error (%) | Type I error (%) | Type II error (%) |
| samp11 | 9.98 | 11.03 | 8.47 | 9.94 | 10.93 | 8.50 |
| samp12 | 3.28 | 2.65 | 4.01 | 3.27 | 2.66 | 3.99 |
| samp21 | 1.96 | 1.17 | 4.84 | 1.96 | 1.17 | 4.84 |
| samp22 | 6.7 | 4.14 | 13.37 | 5.02 | 4.31 | 6.88 |
| samp23 | 7.41 | 7.52 | 7.27 | 6.38 | 7.53 | 4.94 |
| samp24 | 5.41 | 4.71 | 7.21 | 8.56 | 9.22 | 6.86 |
| samp31 | 2.31 | 0.21 | 5.52 | 2.05 | 0.21 | 4.86 |
| samp41 | 6.46 | 5.39 | 7.47 | 4.70 | 6.02 | 3.47 |
| samp42 | 5.61 | 0.86 | 7.93 | 4.74 | 0.86 | 6.63 |
| samp51 | 2.7 | 0.89 | 9.26 | 2.85 | 1.05 | 9.37 |
| samp52 | 7.63 | 7.27 | 10.75 | 8.51 | 8.29 | 10.41 |
| samp53 | 8.73 | 8.81 | 6.92 | 9.59 | 9.69 | 7.00 |
| samp54 | 4.04 | 1.13 | 6.55 | 4.21 | 1.11 | 6.90 |
| samp61 | 3.26 | 3.25 | 3.63 | 3.63 | 3.62 | 3.80 |
| samp71 | 4.71 | 4.15 | 9.17 | 6.63 | 6.29 | 9.29 |
| **Avg.** | **5.35** | **4.21** | **7.49** | **5.47** | **4.86** | **6.52** |

The accuracy comparison of filtering results between parameter sets 01 and 02 is shown in Table 6-11. The results show that when the maximum filtering window size approaches the maximum size of non-ground object features, more accurate results can be achieved. The results on samp41 and samp42 are the typical cases. The omission errors (Type I error) have little difference, while the commission errors (Type II error) are much different. Because in the samp41 and samp42 data sets, there are some non-

ground object features with relatively large size, the maximum filtering window size in parameter set 01 is 20 meters, which is not good enough to filter out large size non-ground features. The parameter set 02's results show the maximum window size (30 meters) is more appropriate for samp41 and samp42, because it can lower the commission error and keep the omission error unchanged as well. Therefore, it is very important for the processing accuracy to choose the appropriate maximum filtering window size. If we have the knowledge on the maximum size of non-ground object features in the data set, it would be very helpful to select the appropriate maximum filtering window size. However, in reality, it is difficult to know the exact maximum size of non-ground object features; therefore, we have to choose a rough maximum filtering window size based on the terrain types of a data set or some other map tools, such as Google Map. Normally, we can decide the maximum size of non-ground objects by the terrain types. For example, there are rarely huge buildings in the mountainous areas. Thus, we can choose a relatively smaller maximum window size. Since the huge buildings with large coverage on the ground are normally located in the relatively flat terrain, we can select relatively larger maximum window size accordingly. Based on this observation, we can use a relatively larger maximum window size (30 meters) to process site 01, 03 and 04, and use a relatively smaller maximum window size (20 meters) to process site 02, 05, 06 and 07. In Table 6-12, we can achieve better overall accuracy of site 01, 03, and 04 under larger maximum window size (30 meters) in parameter set 02, and better overall accuracy of site 02, 05, 06, and 07 under smaller maximum window size (20 meters) in parameter set 01. By combining the two parameter sets' best results, we improve the overall accuracy, which is shown in Table 6-12.

Table 6-12 Accuracy comparison between parameter set 01 and the best results of parameter set 01 & 02

| Dataset | *The Parameter Set 01* | | | *The Best result from Set 01 & 02* | | |
|---------|----------------|----------------|-----------------|----------------|----------------|-----------------|
| | Total error (%) | Type I error (%) | Type II error (%) | Total error (%) | Type I error (%) | Type II error (%) |
| samp11 | 9.98 | 11.03 | 8.47 | 9.94 | 10.93 | 8.50 |
| samp12 | 3.28 | 2.65 | 4.01 | 3.27 | 2.66 | 3.99 |
| samp21 | 1.96 | 1.17 | 4.84 | 1.96 | 1.17 | 4.84 |
| samp22 | 6.7 | 4.14 | 13.37 | 6.7 | 4.14 | 13.37 |
| samp23 | 7.41 | 7.52 | 7.27 | 7.41 | 7.52 | 7.27 |
| samp24 | 5.41 | 4.71 | 7.21 | 5.41 | 4.71 | 7.21 |
| samp31 | 2.31 | 0.21 | 5.52 | 2.05 | 0.21 | 4.86 |
| samp41 | 6.46 | 5.39 | 7.47 | 4.70 | 6.02 | 3.47 |
| samp42 | 5.61 | 0.86 | 7.93 | 4.74 | 0.86 | 6.63 |
| samp51 | 2.7 | 0.89 | 9.26 | 2.7 | 0.89 | 9.26 |
| samp52 | 7.63 | 7.27 | 10.75 | 7.63 | 7.27 | 10.75 |
| samp53 | 8.73 | 8.81 | 6.92 | 8.73 | 8.81 | 6.92 |
| samp54 | 4.04 | 1.13 | 6.55 | 4.04 | 1.13 | 6.55 |
| samp61 | 3.26 | 3.25 | 3.63 | 3.26 | 3.25 | 3.63 |
| samp71 | 4.71 | 4.15 | 9.17 | 4.71 | 4.15 | 9.17 |
| **Avg.** | **5.35** | **4.21** | **7.49** | **5.15** | **4.25** | **7.09** |

If we select the maximum window size to be greater than the necessary size which covers the maximum non-ground objects, the overall accuracy would get worse. We processed the fifteen data sets with parameter set 03 (shown in Table 6-13), which has a larger maximum filtering window size, 40 meters. The results (Table 6-14) show the overall accuracy is getting worse.

Table 6-13 Adaptive filtering parameter set 03

| Window Size | 1 | 2 | 4 | 8 | 16 | 32 | 40 |
|---|---|---|---|---|---|---|---|
| Cluster Threshold | 0.3 | 0.5 | 1 | 1 | 1 | 2 | 2 |
| Row Direction | x | x | x | x | x | x | x |
| Column Direction | x | x | x | x | x | | |

Table 6-14 Accuracy comparison between parameter sets 01 and 03

| Dataset | *The Parameter Set 01* | | | *The Parameter Set 03* | | |
|---|---|---|---|---|---|---|
| | Total error (%) | Type I error (%) | Type II error (%) | Total error (%) | Type I error (%) | Type II error (%) |
| samp11 | 9.98 | 11.03 | 8.47 | 9.97 | 10.90 | 8.64 |
| samp12 | 3.28 | 2.65 | 4.01 | 3.33 | 2.80 | 3.96 |
| samp21 | 1.96 | 1.17 | 4.84 | 1.98 | 1.17 | 4.95 |
| samp22 | 6.7 | 4.14 | 13.37 | 5.45 | 4.64 | 7.57 |
| samp23 | 7.41 | 7.52 | 7.27 | 6.93 | 7.85 | 5.77 |
| samp24 | 5.41 | 4.71 | 7.21 | 13.19 | 15.80 | 6.38 |
| samp31 | 2.31 | 0.21 | 5.52 | 2.26 | 0.21 | 5.38 |
| samp41 | 6.46 | 5.39 | 7.47 | 4.57 | 5.45 | 3.74 |
| samp42 | 5.61 | 0.86 | 7.93 | 4.78 | 0.86 | 6.70 |
| samp51 | 2.7 | 0.89 | 9.26 | 2.84 | 1.05 | 9.34 |
| samp52 | 7.63 | 7.27 | 10.75 | 8.91 | 8.76 | 10.20 |
| samp53 | 8.73 | 8.81 | 6.92 | 9.98 | 10.10 | 7.00 |
| samp54 | 4.04 | 1.13 | 6.55 | 4.18 | 1.06 | 6.88 |
| samp61 | 3.26 | 3.25 | 3.63 | 4.21 | 4.23 | 3.71 |
| samp71 | 4.71 | 4.15 | 9.17 | 6.88 | 6.59 | 9.17 |
| **Avg.** | **5.35** | **4.21** | **7.49** | **5.96** | **5.43** | **6.63** |

6.4 Conclusion and Future Work

Experiments results show that the proposed adaptive morphological filtering provides a very effective and efficient way to adaptively filter the LIDAR data. Since the selection of the filter parameters is very critical and sensitive to different terrain types, it would be very helpful to filter the study area without too many human interactions. Since this method can automatically analyze the terrain shape and obtain plenty of useful terrain information, it also offers a framework to dynamically select different filtering methods for different parts of the data set. Based on the cluster and trend analysis results, the filter can automatically choose a filtering method according to the terrain type. Also, we can acquire more terrain shape information by analyzing the cluster and trend information, which can be used for different kinds of filters.

# Chapter 7

## GUI-BASED LIDAR DATA PROCESSING SYSTEM FOR MODEL

## GENERATION AND MAPPING

Several LIDAR processing methods have been proposed in this dissertation, which can benefit model generation and mapping. High-quality digital elevation model (DEM) and mapping can be generated from good processing results according to different applications. These DEMs and mapping results are indispensable for many applications, such as urban management, emergency event planning, and mapping tools. Since LIDAR data contains a huge volume of data points even in a small surveyed area, it is necessary to have a practical GUI system for the users to process the data according to their application requirements.

Developing the tools to process LIDAR data is a challenging task because of the unique characteristics of LIDAR data [9][54][55]. A huge volume of irregularly spaced points in the LIDAR data is difficult to display, edit and analyze in many commercial GIS and remote sensing software types, which are designed to handle vector and raster images. Therefore, the dedicated LIDAR data processing system for varied application needs is necessary. The high-quality processing results would provide the commercial software with appropriate input for viewing, editing and analyzing. In this chapter, a GUI-based LIDAR data processing system is proposed and discussed in detail.

## 7.1  System Architecture

The proposed GUI system provides a whole workflow to process the LIDAR data. The general processing of LIDAR data consists of several major components, which

include interpolation of data, filtering of data, classification of objects, and generation of object models. The proposed GUI system offers a generalized batch processing interface for users to select different methods to process input data files. In the meantime, it offers a GUI to let users display, edit and process the data on the fly interactively.



Figure 7-1 Schematic procedures for LIDAR data processing

The schematic procedure for LIDAR data processing is shown in Figure 7-1. High-volume LIDAR data in a surveyed area is partitioned into tiles with a predefined size. Some project data needs to be split into strips in the *Strip Boundary & Merge* procedure before being partitioned into tiles. This *Data Partition* procedure would split a huge data set into small pieces for better storage and processing. In the *Data Preprocessing* procedure, the outlier points, such as extremely low points, would be removed. Also, the interpolation of empty parts in the data set could be carried out in this procedure. The *Filtering* procedure will separate the ground and non-ground features according to the application needs. Multiple filtering methods have been implemented in the *Filtering* module. The separated ground and non-ground points will be processed in the *Features Classification* procedure to further extract the desired terrain or non-terrain features according to the application purposes. Finally, the extracted features would be used to generate a digital terrain model (DTM) or digital elevation model (DEM). These processing modules have been implemented in a generalized multi-task batch processing

160

GUI. Users can use the same interface to select the processing input/output files, methods, and parameters.

The proposed system offers GUI for users to interactively view the three-dimensional LIDAR data, which is displayed in different colors according to their elevations. The legend can be modified to satisfy different display purposes. The data set's information, such as 3-D boundary information, is shown in the GUI. Users can also retrieve a single LIDAR point's location information by pointing the cursor to it. The view can be zoomed in and out. Multiple views of display and processing results are cached for undo and redo operations. Users can switch the data views back and forth easily in this way. These viewing functionalities are very helpful for users to study the spatial detail of the data set and compare the processing results.

The GUI offers the editing and processing operations through polygon shape selection in the data view. These functionalities are necessary for users to edit and process a complex data set, especially when the users need to visually select various areas of interest in a big data set to carry out different operations. These operations include exporting data, removing data, filtering data, objects detection, and objects classification in the selected regions. By using these functionalities, users can visually choose the study area in the data set and carry out their processing methods repeatedly. The processing results will be displayed on the fly and can be retrieved from cache by undo/redo operations.

These features of the system are very useful for processing and analyzing large complex data sets, which is difficult for a single filtering method to separate ground and non-ground features. With the interactive features, users are able to select any part of the

data set by using a polygon, and process it with different methods and parameters according to the applications. It offers users much flexibility to handle different kinds of terrain with various configurations.

## 7.2    System Demonstration

The proposed system is demonstrated by a personal computer program on the Windows platform. A sample LIDAR data set is used for the viewing, editing and processing functionalities. The functionalities presented to the users are described as follows [9].
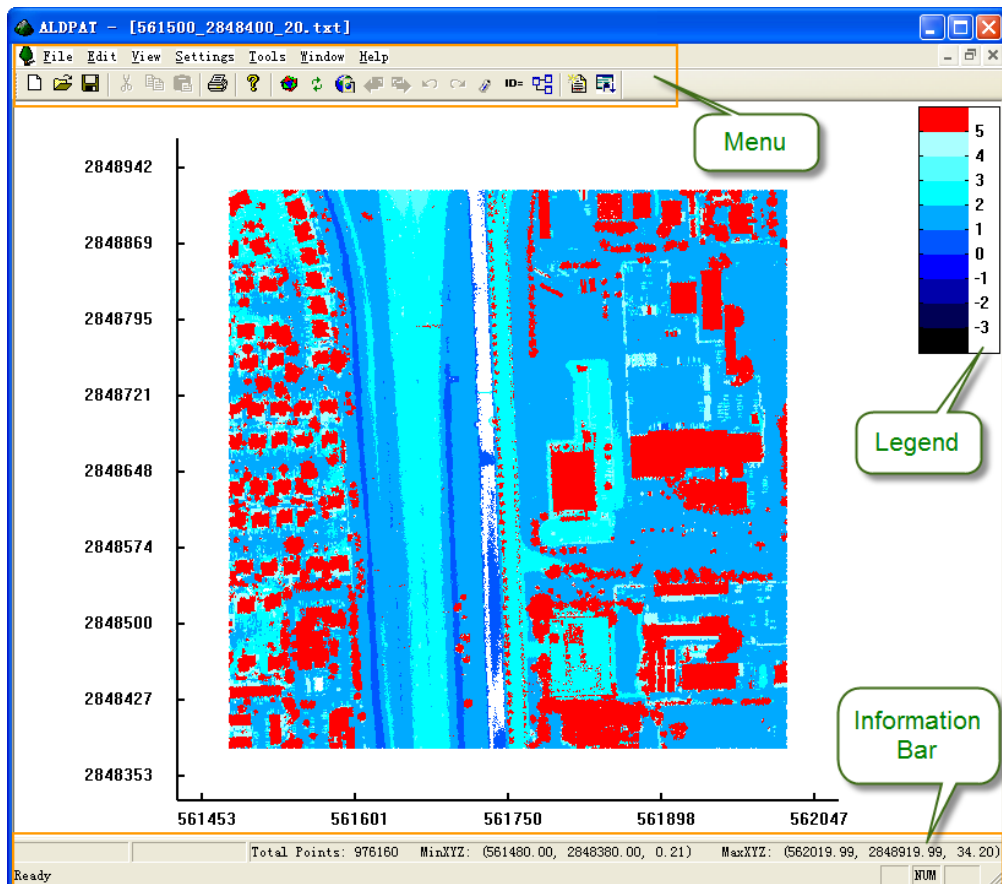


Figure 7-2 GUI view of LIDAR data

The main GUI is shown in Figure 7-2, which uses Document/View architecture for the Visual C++ project. The data points can be shown in the view with different colors defined in the legend bar. The legend bar's settings dialog is shown in Figure 7-3. Multiple parameters can be modified by users. The elevation display settings use relative elevation values to the benchmark value. Users can set the benchmark value to define the origin of vertical axis. All the elevation levels can be defined relative to the benchmark value in a list. Different elevation ranges are displayed by user-defined colors. Users can change the color as they need, and the interval between each elevation level can be modified by setting *Interval* value. The legend settings offer users a good way to display the data. Users can use mouse to select any rectangle region to zoom in on the selected area. Multiple views are cached for users to retrieve.
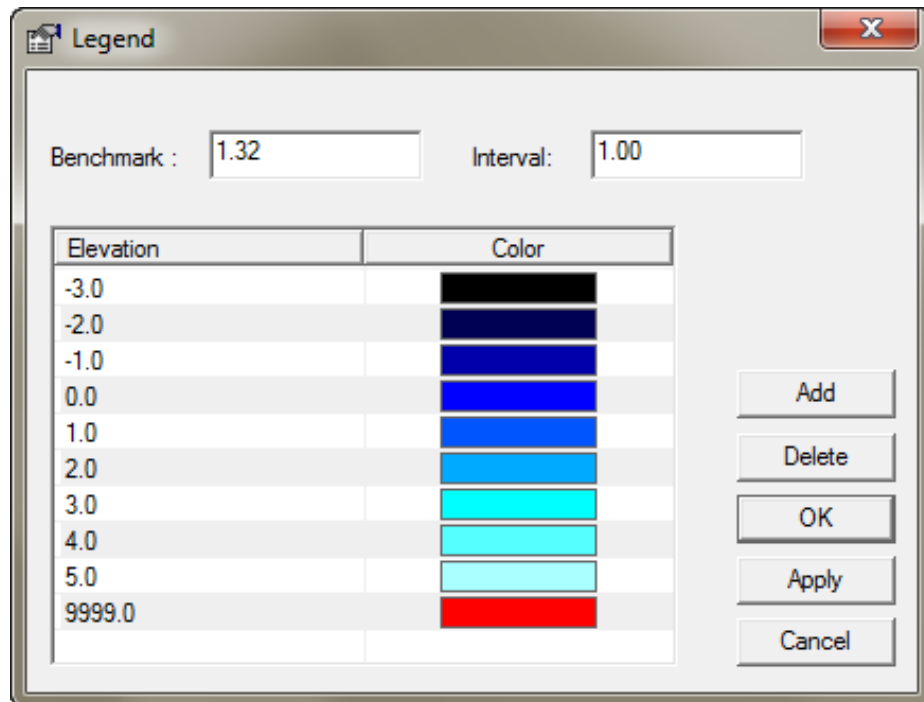


Figure 7-3 Legend bar settings

The interactive processing interface is shown in Figure 7-4. The figure shows that users can select any polygon-shaped area to carry out related processing operations, such as export, deletion, and other filtering-related operations. The polygon shape is adjustable; users can resize the polygon shape by dragging its vertices. After the polygon region is selected, users can do all the related operations by right-clicking the polygon. The operation menu will be prompted for users to choose the operations. For example, if the user chooses a filtering operation such as *Morph Filter*, a parameters form will pop up. The user can fill in the parameters of the method shown in Figure 7-5, and then the processing will start after the user clicks the OK button. The operation result is shown in Figure 7-6. To process the same polygon size of data in a different area of the data set, the user can drag the polygon shape to a different part of the data set and choose the appropriate operation for that area. Figure 7-7 shows the results of another operation on a different part of data set with the same polygon shape. This operation extracts the non-ground features of the data set, while the non-ground features were removed in the previous polygon-shaped area. Users can select different operation methods to satisfy their own data retrieval needs. The processing results in each polygon shape can be exported into file for further analysis and processing. This GUI-based interface provides a framework for testing and comparing different methods. All the related LIDAR processing methods can be embedded into this GUI interface. It would benefit researchers to test and compare various methods on different areas of each data set, because users can easily switch between different processing results' view through *Undo* and *Redo* buttons. Another operation result is shown in Figure 7-8. It extracts the points of buildings in the data set. This can be further used to retrieve the boundaries of these

buildings. Figure 7-9 shows the results of the boundaries identified from the extracted buildings. With this information, the buildings' boundaries can be easily displayed in commercial software, such as ArcGIS, and overlapped on the web map services, such as Google Map. The results of the boundaries also provide important information for 3-D model generation.



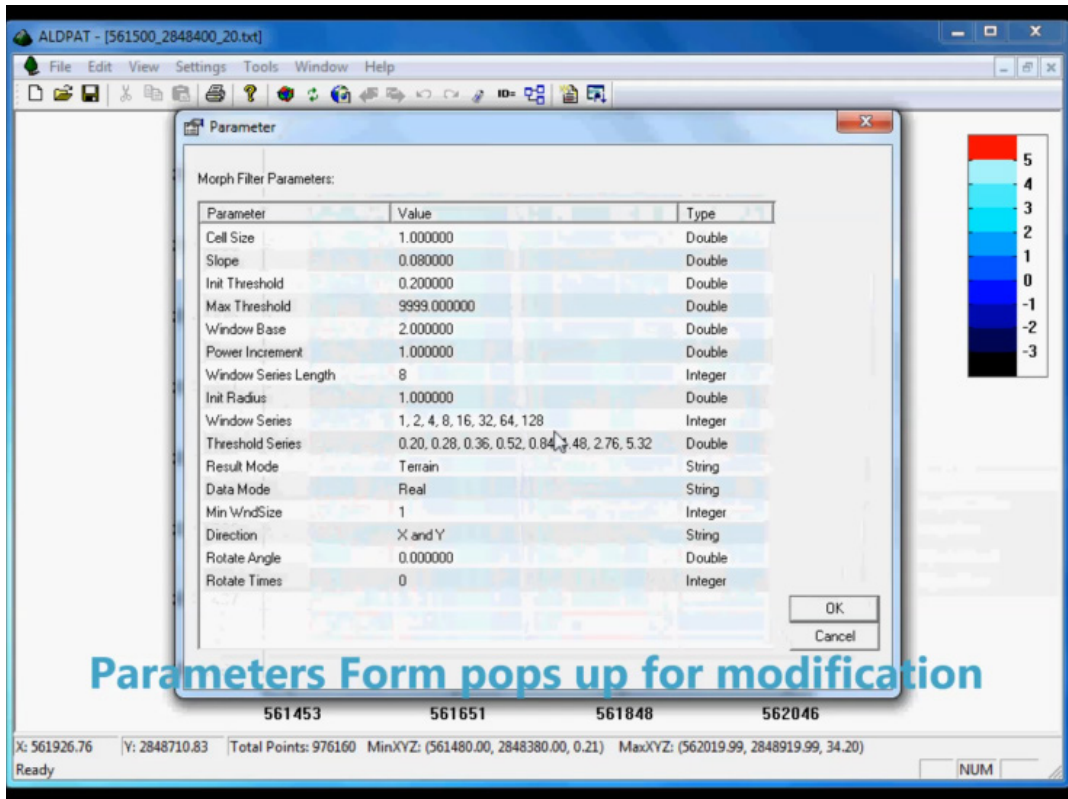Figure 7-4 Operations on selected polygon region
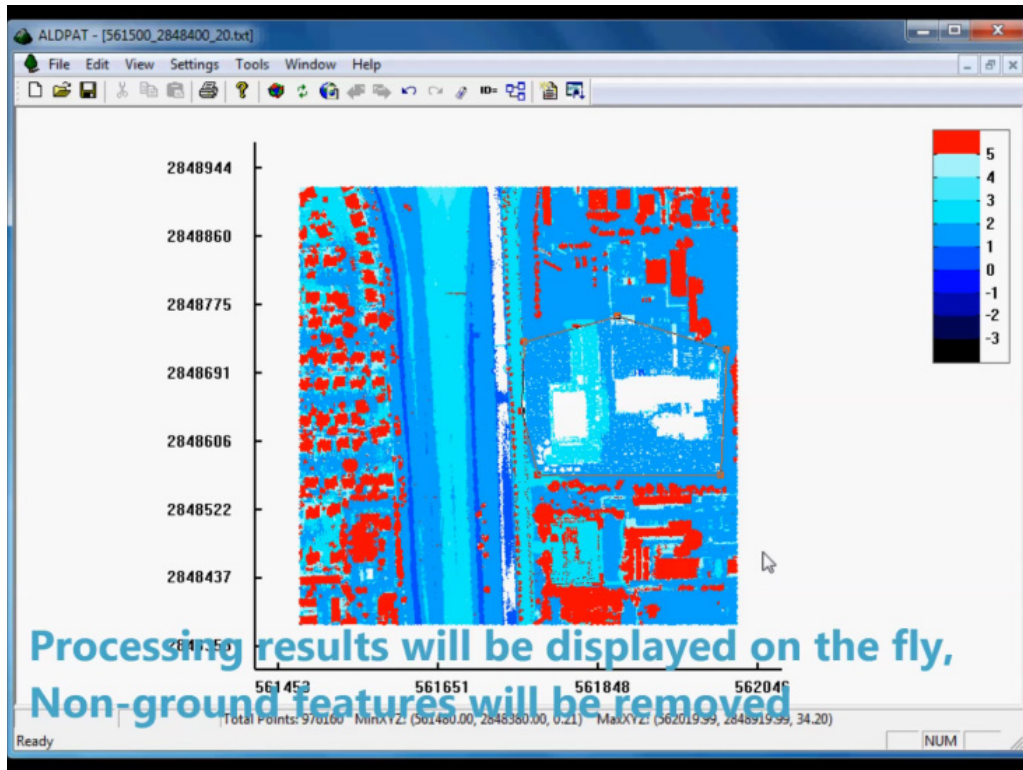
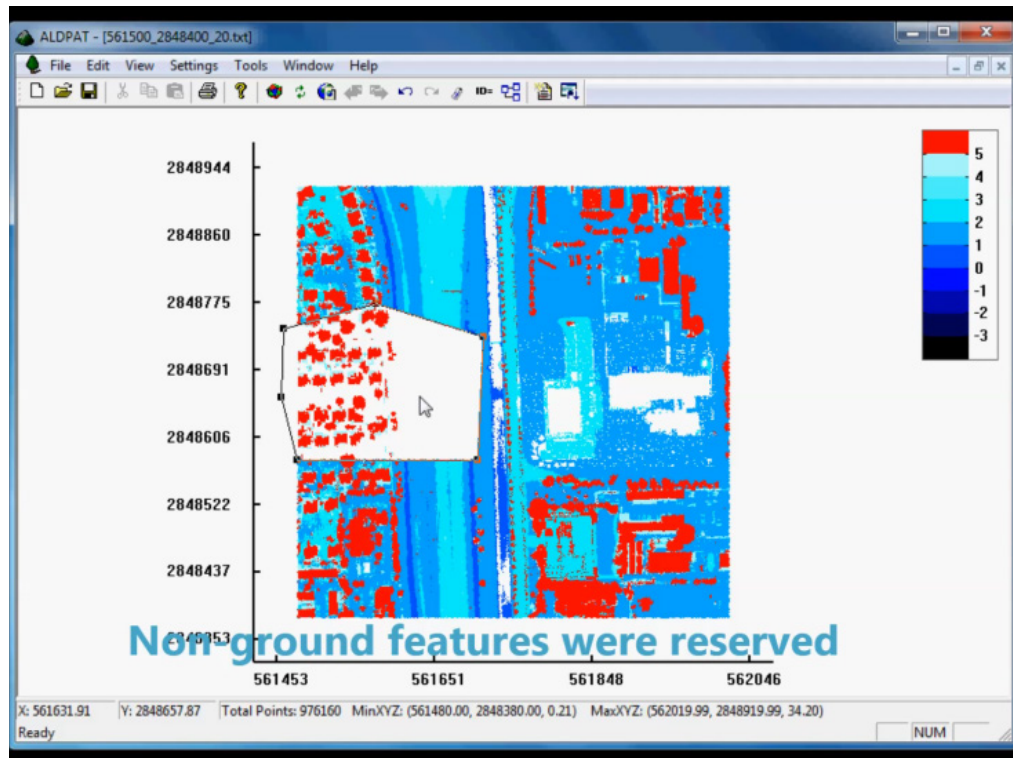Figure 7-5 Parameters form of the operation

Figure 7-6 GUI operation result



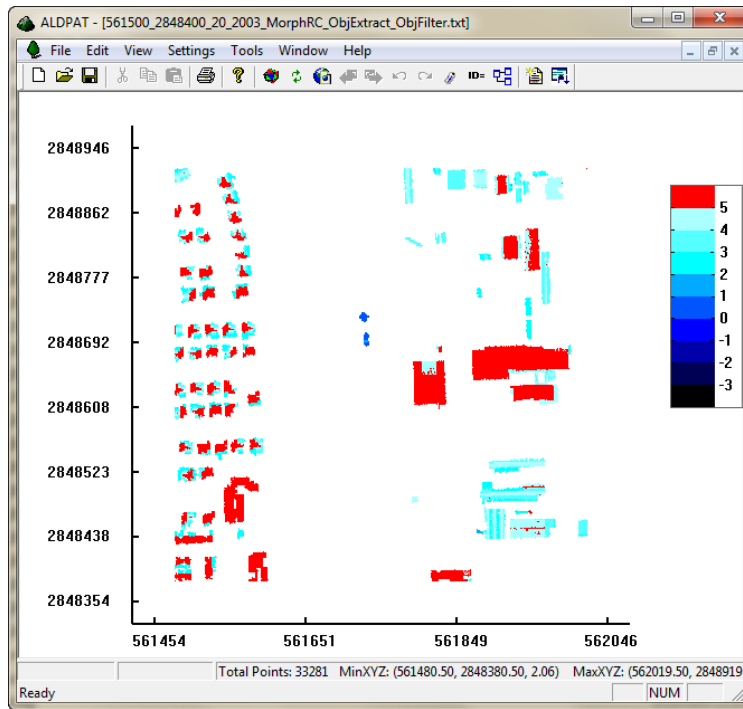Figure 7-7 Another operation result with the same polygon shape
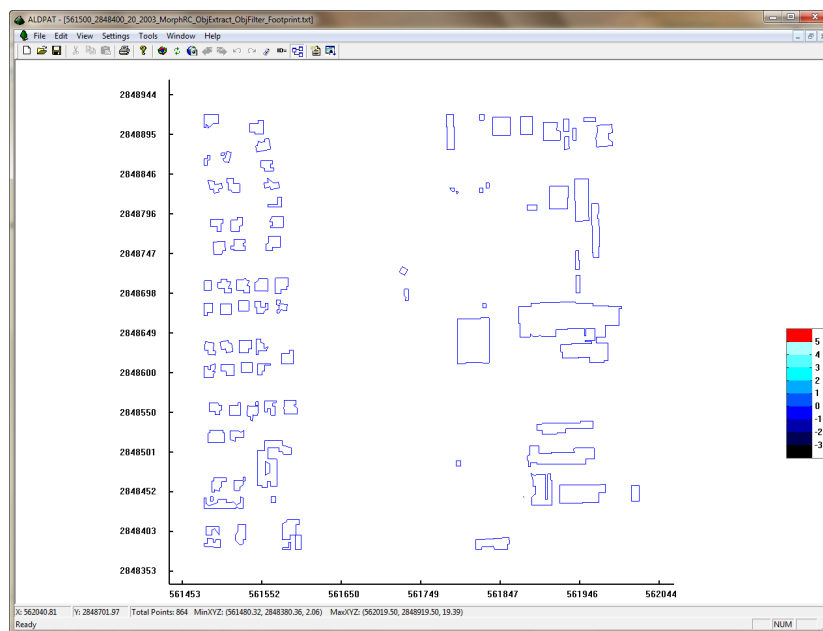
167

Figure 7-8 Building extraction



Figure 7-9 Non-ground objects' boundaries detection

The batch processing interface is shown in Figure 7-10. Users can choose from a list of implemented methods, set the parameters for each method, and add it into the processing task list. The processing input and output file can be selected and set through the general batch processing form. The processing method can be selected from the methods list. Before the processing job is added to the task list, a parameters form will pop up for users to fill in all the parameters of the method. Multiple jobs can be added into the task list; they will be processed sequentially. The processing time and job status will be displayed for each job. Users can stop the processing job anytime. These features are very helpful for comparing methods' performance, especially for large data sets, which require long processing time. The detailed batch processing features are explained in [55]:
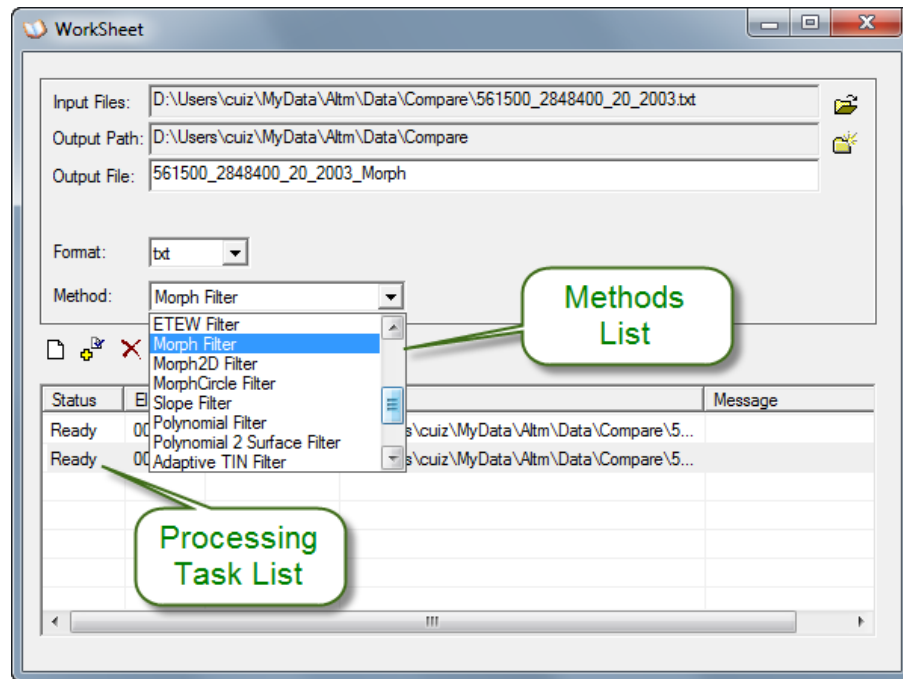

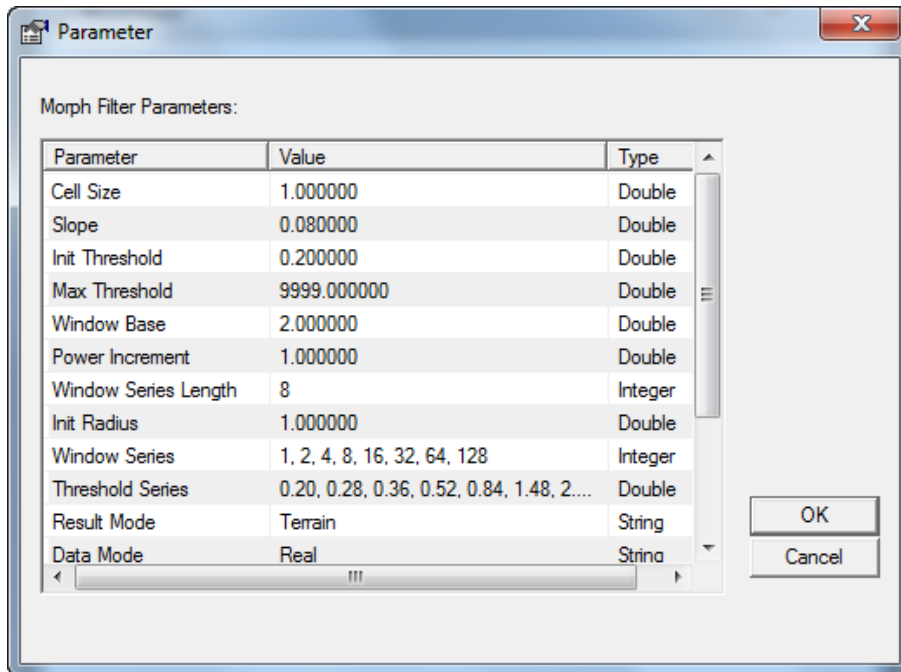
Figure 7-10 Batch processing interface

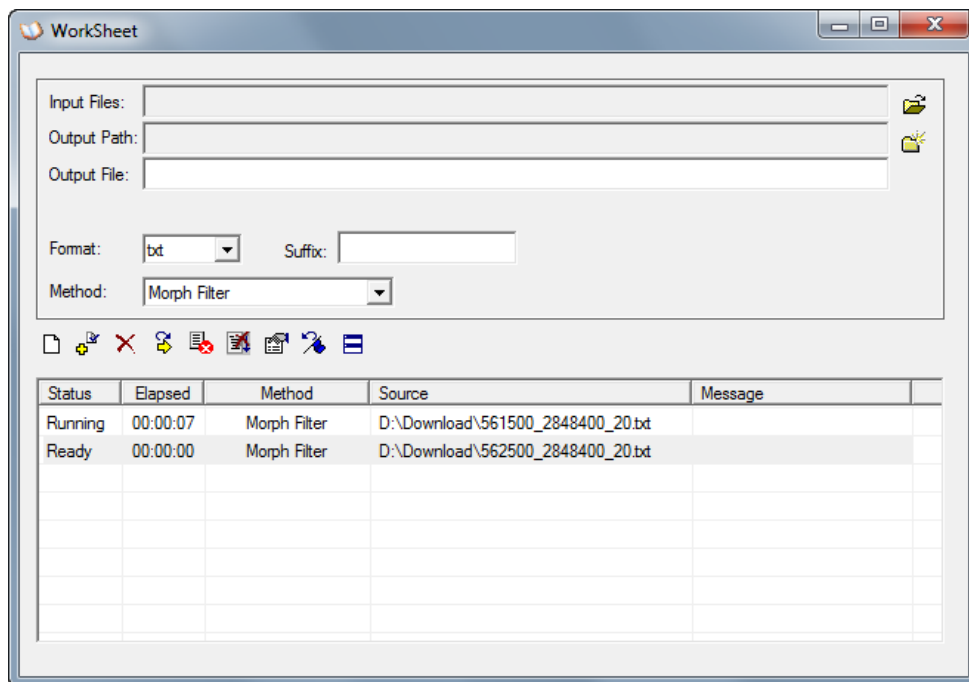Figure 7-11 Parameter form of the selected method



Figure 7-12 Processing jobs' status are shown in task list pane

# Chapter 8

## CONCLUSIONS AND FUTURE WORK

The major motivation behind the research presented in this dissertation is to develop a LIDAR data processing framework, which would provide an effective and efficient way to process LIDAR data on varied terrain types. Due to the characters of LIDAR data, it normally contains a huge number of data points. It is a tremendous challenge for the storage and processing. An effective LIDAR processing framework should include all the necessary components, such as data storage, pre-processing, filtering and model generation. All the methods for each component should be able to work effectively and efficiently, so that it can satisfy the practical processing purpose, rather than ideas and theories. Thus, the performance of the method is critical for practical use, because the practical data sets are much larger than the testing set for research purposes. Among all the components, the filtering component is the most critical part for the LIDAR data processing. Therefore, it is the most important part in this research.

This dissertation focuses on the development of a generalized adaptive LIDAR filter, which can work on varied terrain types. A cluster-based morphological filter was proposed to solve the cut-off problem for the morphological filter. An adaptive trend analysis morphological filter was proposed to achieve a generalized filter on varied terrain. Since so many filtering methods have been developed, and have different performance on various terrain types, it creates a hassle for practical use, because it is very difficult for an inexperienced user to choose the appropriate filtering methods to process a large and complex terrain. Almost all the filters that have been developed can

be categorized into the following types: surface-based filters, segmentation-based filters, region-based filters, TIN-based filters and slope-based filters. They all have their own advantages and disadvantages, which cause them to perform differently on varied terrains. In practical use, a large data set normally contains complex and multiple terrain types, which makes it difficult for a single method to achieve ideal results. Current filtering research shows a trend to combine multiple filter methods' features to suit complex and various terrain types. In this dissertation, mathematical morphological filtering, cluster, trend analysis and slope filtering are included in our proposed methods that can make the filter more generalized and adaptive to varied terrains for practical use. Also, the related interpolation method has been discussed in this dissertation, and a grid-based priority interpolation method was proposed to interpolate empty holes around filtering residuals, and to assist the filter in removing them.

## 8.1   Conclusions

The results and conclusions from the work described above are summarized as follows:

1.  Literature review

A comprehensive literature search and review was performed to investigate the LIDAR processing components and methods. Also, the advantages and disadvantages of different methods have been compared. The challenges of methods have been presented. The literature review demonstrates all of the LIDAR processing procedures and the role of each component. The difficulties and challenges of varied LIDAR filtering methods

have been discussed. The advantages and disadvantages of each method have been shown as well in the literature review.

2. Data interpolation

Common interpolation methods in our framework have been discussed. A grid-based priority interpolation method was proposed to interpolate empty grids. This interpolation method can be combined with a multi-pass morphological filter to remove large non-ground objects' filtering remains for some complex terrains.

3. Filtering methods development

A cluster-based morphological filter was proposed to solve the cut-off problem for the morphological filter. The cut-off problem is a common problem for morphological filters, because there is no mechanism involved in most of the morphological filters to detect the terrain shapes and the spatial connectivity of points. Especially in the undulating terrain and steep terrain, morphological filters turn out more errors. The cluster-based method proposed provides an iterative way to check the connectivity of a complete spatial terrain feature, which can recover the cut-off features during morphological filtering.

An adaptive trend analysis morphological filter was proposed to achieve adaptive filtering threshold estimation according to the terrain changes. One of the most challenging problems for morphological filters is how to provide the filtering thresholds adaptively based on the terrain shape. Some morphological filters use the filtering thresholds from users' input according to their experience, and some use simple formula to estimate the thresholds. These methods result in either a constant threshold under each filtering window or the same threshold on different terrain types. They would not work

very well on complex terrains, and on a large data set for practical processing as well. Thus, a generalized adaptive method is necessary for practical processing. The adaptive trend analysis morphological filter proposed in this dissertation combines the features of cluster analysis, adaptive trend analysis and progressive morphological filter. It can achieve the goal of a generalized LIDAR filtering method for practical use.

4. GUI-based LIDAR data processing system

A GUI-based LIDAR data processing system was proposed to aim for the LIDAR processing related methods design and development. It implemented many LIDAR processing related methods in our research work, which makes it a useful LIDAR data processing tool for practical use.

## 8.2 Future Work

The proposed LIDAR filtering methods can be improved in the following directions to make them more generalized for practical use.

More generalized adaptive filtering methods can be studied in the following issues:

1. Optimization of trend line analysis

The trend line analysis method can be studied and tested on more data sets to optimize the threshold estimation. The trend estimation method can be studied on more sophisticated geometry analysis to improve the estimation algorithm.

2. Three-Dimensional trend surface analysis

The trend line analysis method proposed in this dissertation is based on two-dimensional data (x or y coordinates with an elevation dimension). It can be extended to

three dimensions, so that the trend surface can be analyzed for threshold estimation. It would be more sufficient for the analysis of terrain or objects' spatial shapes.

3. Adaptive selection of filtering methods

Since different filtering methods have their own advantages on varied terrain types, it would be very helpful to use each method's advantages to filter each part of a large data set with different terrain types. Thus, an adaptive filter selection mechanism would improve the filtering results on a data set which contains complex terrains. This method relies on the terrain type analysis and partition. First, the data set has to be analyzed to find varied terrain types. Then, the data set needs to be partitioned according to the terrain types. Thus, how to partition the data set according to the terrain types would be the critical part of this study. After the data set is successfully partitioned according to the terrain types, the appropriate filtering method can be automatically selected.

REFERENCES

[1]  Arefi, H.; and Hahn, M. (September 2005). A morphological reconstruction algorithm for separating off-terrain points from terrain points in laser scanning data. Proceedings of the ISPRS Workshop Laser Scanning, Enschede, The Netherlands.

[2]  Axelsson, P., (2000). DEM Generation from Laser Scanner Data Using Adaptive TIN Models. Int. Arch. Photogramm. Remote Sens., vol. 33, pp. 110-117.

[3]  Axelsson, P., (1999). Processing of laser scanner data-algorithms and applications. ISPRS J. Photogramm. Remote Sens., 54, pp. 138-147.

[4]  Bourke, P., (1989). Efficient triangulation algorithm suitable for terrain modeling. Pan Pacific Computer Conference, Beijing, China.

[5]  Brovelli, M. A., and Cannata, M. (2004). Digital terrain model reconstruction in urban areas from airborne laser scanning data: The method and the example of the town of Pavia (Northern Italy). Computers & Geosciences, vol. 30.4, pp. 325-331.

[6]  Brovelli, M. A., Cannata, M., and Longoni, U.M. (2004). LIDAR data filtering and DTM interpolation within GRASS. Trans. GIS, 8, 155-174.

[7]  Brugelmann, R., (2000). Automatic breakline detection from airborne laser range data. in Int. Arch. Photogramm. Remote Sens., Amsterdam, Netherlands, vol. 33, B3, pp. 109–115.

[8]  Crosilla, F., Visintini, D., and Prearo, G. (2004). A robust method for filtering non-ground measurements from airborne LIDAR data. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 35, pp. 196-201.

[9]  Cui, Z., Zhang, K., Zhang, C., Yan, J., and Chen, S.-C. (November 5-8, 2013). A GUI Based LIDAR Data Processing System for Model Generation and Mapping. Accepted for publication, 1st ACM SIGSPATIAL International Workshop on Interacting with Maps (MapInteract) 2013, in conjunction with ACM SIGSPATIAL 2013, Orlando, Florida, USA. (Demo Paper)

[10] Cui, Z., Zhang, K., Zhang, C., and Chen, S.-C. (November 5-8, 2013). A Cluster-based Morphological Filter for Geospatial Data Analysis. Accepted for publication, 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, in conjunction with 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013), Orlando, Florida, USA.

[11]  Cui, Z., Zhang, K., Zhang, C., and Chen, S.-C. (December 16-19, 2013). A Multi-Pass Morphological Filter for Geospatial Data Analysis. Accepted for publication, 2013 IEEE International Workshop on Big Data Analytics in Mobile Social Networks (BigMSN), in conjunction with CloudCom-Asia 2013, Fuzhou, China.

[12]  Cui, Z., Zhang, K., Zhang, C., Yan, J., and Chen, S.-C. A Generalized Adaptive Mathematical Morphological Filter for LIDAR Data. will be submitted to IEEE Transactions on Geoscience and Remote Sensing.

[13]  Douglas, D. H., and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization 10.2: pp. 112-122.

[14]  Filin, S. (2002). Surface clustering from airborne laser scanning data. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 34, pp. 119-124.

[15]  Ghosh, S. and Lohani, B. (2007). Near-realistic and 3D visualization of LIDAR data. in Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 36.4/W45, Joint Workshop "Visualization and Exploration of Geospatial Data".

[16]  Haines, E. (1994). Point in polygon strategies. Graphics gems IV 994: 24-t6.

[17]  Jacobsen, K., and Lohmann, P. (2003). Segmented filtering of laser scanner DSMs. in Int. Arch. Photogramm. Remote Sens., vol. 34.3/W13.

[18]  Kilian, J., Haala, N., and Englich, M. (1996). Capture and Evaluation of Airborne Laser Scanner Data. Int. Arch. Photogramm. Remote Sens., Vienna, vol. 31, Part B3, pp. 383–388.

[19]  Kraus, K., and Otepka, J. (2005). DTM modeling and visualization—the SCOP approach. Proceedings of Photogrammetric Week 05, Heidelberg, Germany, pp. 241-252.

[20]  Kraus, K., and Pfeifer, N. (2001). Advanced DTM generation from LIDAR data. Proceedings of the ISPRS Workshop on Land Surface Mapping and Characterization Using Laser Altimetry, Hofton, M.A., Ed., Annapolis, MD, USA.

[21]  Kraus, K. and Pfeifer, N. (1998). Determination of terrain models in wooded areas with aerial laser scanner data. ISPRS J. Photogramm. Remote Sens., 53, pp. 193-203.

[22]  Kraus, K., and Rieger, W. (1999). Processing of laser scanning data for wooded areas. Photogrammetric Week99, Fritsch, D., Spiller, R., Eds., Wichmann Verlag: Stuttgart, Germany, pp. 221-231.

[23] Lee, H.S., and Younan, N.H. (2003). DTM extraction of LIDAR returns *via* adaptive processing. IEEE Trans. Geosci. Remote Sens., *41*, 2063-2069.

[24] Liu, X. (February, 2008). Airborne LIDAR for DEM generation: some critical issues. Progress in Physical Geography, 32, 1, pp. 31-49.

[25] Lloyd, C.D., and Atkinson, P.M. (2006). Deriving ground surface digital elevation models from LIDAR data with geostatistics. International Journal of Geographical Information Science, 20, pp. 535-563.

[26] Lohmann, P., Koch, A., and Schaeffer, M. (2000). Approaches to the filtering of laser scanner data. Int. Arch. Photogramm. Remote Sens., vol. 33, pp. 540-547.

[27] Masaharu, H., and Ohtsubo, K. (2002). A Filtering Method of Airborne Laser Scanner Data for Complex Terrain. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 34.3/B, pp. 165–169.

[28] Meng, X., (2005). A slope- and elevation-based filter to remove non-ground measurements from airborne LIDAR data. Proceedings of ISPRS WG III/3, III/4, V/3 Workshop "Laser scanning 2005", The Netherlands, September 2005, pp. 23.

[29] Meng, X., Currit, N., and Zhao, K. (2010). Ground filtering algorithms for airborne LIDAR data: a review of critical issues. Remote Sensing, vol. 2, pp. 833–860.

[30] Mongus, D., and Žalik, B. (2012). Parameter-free Ground Filtering of LIDAR Data for Automatic DTM Generation. ISPRS J. Photogramm. Remote Sens., 67, pp. 1–12.

[31] Morgan, M. and Tempfli, K. (2000). Automatic Building Extraction from Airborne Laser Scanning Data. Int. Arch. Photogramm. Remote Sens., vol. 33.B3/2, pp. 616-623.

[32] Petzold, B., Reiss, P., and Stossel, W. (1999). Laser scanning - surveying and mapping agencies are using a new technique for the derivation of digital terrain models. ISPRS J. Photogramm. Remote Sens., 54.2, pp. 95-104.

[33] Pfeifer, N., Gorte, B., and Elberink, S. O. (2004). Influences of vegetation on laser altimetry–analysis and correction approaches. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol 36, part 8.

[34] Pfeifer, N., Reiter, T., Briese, C., and Rieger, W. (1999). Interpolation of high quality ground models from laser scanner data in forested areas. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 32.3/W14, pp. 31-36.

[35] Pfeifer, N., Stadler, P., and Briese, C. (2001). Derivation of digital terrain models in the SCOP++ environment. Proceedings of OEEPE Workshop on Airborne Laser Scanning and Interferometric SAR for Digital Elevation Models, Stockholm, Sweden, vol. 3612.

[36] Rabbania, T., Heuvel, F. A. van den and Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 36.5, pp. 248-253.

[37] Roggero, M., (2001). Airborne laser scanning: clustering in raw data. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 34.3/W4, pp. 227-232.

[38] Schickler, W., and Thorpe, A. (April 2001). Surface estimation based on LIDAR. Proceedings of ASPRS Annual Conference, St. Louis, MO, USA.

[39] Shi, W.Z., and Tian, Y. (2006). A hybrid interpolation method for the refinement of a regular grid digital elevation model. International Journal of Geographical Information Science, vol. 20, pp. 53-67.

[40] Sithole, G., (2001). Filtering of laser altimetry data using a slope adaptive filter. Int. Arch. Photogramm. Remote Sens., vol. 34-3/W4, pp. 203-210.

[41] Sithole, G., and Vosselman, G. (2004). Experimental comparison of filter algorithms for bare earth extraction from airborne laser scanning point clouds. ISPRS J. Photogramm. Remote Sens., 59.1, pp. 85-101.

[42] Sithole, G., and Vosselman, G. (2005). Filtering of airborne laser scanner data based on segmented point clouds. Proceedings of ISPRS Workshop Laser Scanning 2005, Enschede, the Netherlands, pp. 66-71.

[43] Smith, S.L. Holland, D.A. and Longley, P.A. (2005). Quantifying interpolation errors in urban airborne laser scanning models. Geographical Analysis, 37.2, pp. 200-224.

[44] Sohn, G., and Dowman, I. (2002). Terrain surface reconstruction by the use of tetrahedron model with the MDL Criterion. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 34.3/A, pp. 336–344.

[45] Tóvári, D., and Pfeifer, N. (2005). Segmentation based robust interpolation–a new approach to laser filtering. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 36.3/W19, pp. 79-84.

[46] Vosselman, G., (2000). Slope based filtering of Laser altimetry data. Int. Arch. Photogramm. Remote Sens., vol. 33.B3/2, pp. 935-942.

[47] Wack, R., and Wimmer, A. (2002). Digital terrain models from airborne laserscanner data – A grid based approach. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. 34.3/B, pp. 293–296.

[48] Wang, C., Menenti, M., Stoll, M.P., Alessandra, F., Enrica, B., and Marco, M. (2009). Separation of Ground and Low Vegetation Signatures in LIDAR Measurements of Salt-Marsh Environments. IEEE Trans. Geosci. Remote Sens., 47.7, pp. 2014-2023.

[49] Whitman, D., Zhang, K., Leatherman, S.P., and Robertson, W. (2003). Airborne laser topographic mapping: application to hurricane storm surge hazards. Earth Science in the Cities: A Reader, Heiken, G., Fakundiny, R., Sutter, J., Eds., American Geophysical Union: Washington, DC, USA, pp. 363-376.

[50] Yang, C.-S., Kao, S.-P., Lee, F.-B., and Hung, P.-S. (2004). Twelve different interpolation methods: A case study of Surfer 8.0. Proceedings of the XXth ISPRS Congress, 35.

[51] Zhang, K., Chen, S.-C., Whitman, D., Shyu, M.-L., Yan, J. and Zhang, C. (April, 2003). A progressive morphological filter for removing nonground measurements from airborne LIDAR data. IEEE Transactions on Geoscience and Remote Sensing, 41.4, pp. 872-882.

[52] Zhang, K., and Whitman, D. (2005). Comparison of three algorithms for filtering airborne LIDAR data. Photogrammetric Engineering and Remote Sensing, vol. 71, pp. 313-324.

[53] Zhang, K., Yan, J., and Chen, S.-C. (2006). Automatic construction of building footprints from airborne LIDAR data. IEEE Transactions on Geoscience and Remote Sensing, 44. 9, pp. 2523-2533.

[54] Zhang, K., Cui, Z., and Houle, P. (2012). Airborne LIDAR Remote Sensing and Its Application. in Advances in Mapping from Remote Sensor Imagery Techniques and Applications, edited by Xiaojun Yang and Jonathan Li. CRC Press, in press.

[55] Zhang, K., and Cui, Z. (2007). Airborne LIDAR Data Processing and Analysis Tools ALDPAT 1.0. National Center for Airborne Laser Mapping, International Hurricane Research Center, Department of Environmental Studies, Florida International University, ALDPAT user manual 81 pages.

ZHENG CUI


Born, Beijing, China

EDUCATION

| 1992 - 1997 | B.S. in Automation Department |
| | Tsinghua University, |
| | Beijing, China |

1999 - 2001        M.S. in Computer Science
School of Computing and Information Sciences
Florida International University
Miami, Florida

2005 - Present     Ph.D. candidate in Computer Science
School of Computing and Information Sciences
Florida International University
Miami, Florida

EMPLOYMENT

1997 - 1998        Software Engineer
Greatwall Computer Software Company
Beijing, China

2001 - 2001        Software Engineer
Asoki Software Company
Miami, Florida

2002 - Present     Software Engineer, Research Associate, IT Administrator
International Hurricane Research Center
Florida International University
Miami, Florida


PUBLICATIONS AND PRESENTATIONS

Cui, Z., Zhang, K., Zhang, C., Yan, J., and Chen, S.-C. A Generalized Adaptive Mathematical Morphological Filter for LIDAR Data. will be submitted to IEEE Transactions on Geoscience and Remote Sensing.

Cui, Z., Zhang, K., Zhang, C., and Chen, S.-C. (December 16-19, 2013). A Multi-Pass Morphological Filter for Geospatial Data Analysis. Accepted for publication, 2013 IEEE International Workshop on Big Data Analytics in Mobile Social Networks (BigMSN), in conjunction with CloudCom-Asia 2013, Fuzhou, China.

Cui, Z., Zhang, K., Zhang, C., Yan, J., and Chen, S.-C. (November 5-8, 2013). A GUI Based LIDAR Data Processing System for Model Generation and Mapping. Accepted for publication, 1st ACM SIGSPATIAL International Workshop on Interacting with Maps (MapInteract) 2013, in conjunction with ACM SIGSPATIAL 2013, Orlando, Florida, USA. (Demo Paper)

Cui, Z., Zhang, K., Zhang, C., and Chen, S.-C. (November 5-8, 2013). A Cluster-based Morphological Filter for Geospatial Data Analysis. Accepted for publication, 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, in conjunction with 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013), Orlando, Florida, USA.

Zhang, K., Cui, Z., and Houle, P. (2012). Airborne LIDAR Remote Sensing and Its Application. in Advances in Mapping from Remote Sensor Imagery Techniques and Applications, edited by Xiaojun Yang and Jonathan Li. CRC Press, in press.

Zhang, K., and Cui, Z. (2007). Airborne LIDAR Data Processing and Analysis Tools ALDPAT 1.0. National Center for Airborne Laser Mapping, International Hurricane Research Center, Department of Environmental Studies, Florida International University, ALDPAT user manual 81 pages.