FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A FRAMEWORK FOR AUTOMATIC FEATURE EXTRACTION FROM LIDAR DATA

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Jianhua Yan

2007

To: Dean Vish Prasad
College of Engineering and Computing

This dissertation, written by Jianhua Yan, and entitled A FRAMEWORK FOR AUTOMATIC FEATURE EXTRACTION FROM 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.

Keqi Zhang

Xudong He

Nagarajan Prabakar

Shu-Ching Chen, Major Professor

Date of Defense: March 15, 2007

The dissertation of Jianhua Yan is approved.

Dean Vish Prasad
College of Engineering and Computing

Interim Dean Stephan L. Mintz
University Graduate School

Florida International University, 2007

ABSTRACT OF THE DISSERTATION

A FRAMEWORK FOR AUTOMATIC FEATURE EXTRACTION FROM LIDAR DATA

by

Jianhua Yan

Florida International University, March 15, 2007

Miami, Florida

Professor Shu-Ching Chen, Major Professor

0.1 Abstract

Recent advances in airborne Light Detection and Ranging (LIDAR) technology allow rapid and inexpensive measurements of topography over large areas. Airborne LIDAR systems usually return a 3-dimensional cloud of point measurements from reflective objects scanned by the laser beneath the flight path. This technology is becoming a primary method for extracting information of different kinds of geometrical objects, such as high-resolution digital terrain models (DTMs), buildings and trees, etc. In the past decade, LIDAR gets more and more interest from researchers in the field of remote sensing and GIS. Compared to the traditional data sources, such as aerial photography and satellite images, LIDAR measurements are not influenced by sun shadow and relief displacement. However, voluminous data pose a new challenge for automated extraction the geometrical information from LIDAR measurements because many raster image processing techniques cannot be directly applied to irregularly spaced LIDAR points.

In this dissertation, a framework is proposed to filter out information about different kinds of geometrical objects, such as terrain and buildings from LIDAR automatically. They are essential to numerous applications such as flood modeling, landslide prediction and hurricane animation. The framework consists of several intuitive algorithms. Firstly, a progressive morphological filter was developed to detect non-ground LIDAR measurements. By gradually increasing the window size and elevation difference threshold of the filter, the measurements of vehicles, vegetation, and buildings are removed, while ground data are preserved. Then, building measurements are identified from no-ground measurements using a region growing algorithm based on the plane-fitting technique. Raw footprints for segmented building measurements are derived by connecting boundary points and are further simplified and adjusted by several proposed operations to remove noise, which is caused by irregularly spaced LIDAR measurements. To reconstruct 3D building models, the raw 2D topology of each building is first extracted and then further adjusted. Since the adjusting operations for simple building models do not work well on 2D topology, 2D snake algorithm is proposed to adjust 2D topology. The 2D snake algorithm consists of newly defined energy functions for topology adjusting and a linear algorithm to find the minimal energy value of 2D snake problems. Data sets from urbanized areas

including large institutional, commercial, and small residential buildings were employed to test the proposed framework. The results demonstrated that the proposed framework achieves a very good performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

LIDAR (LIght Detection And Ranging) is a new and versatile technology for the highly automated generation of a digital terrain model (DTM)/digital elevation model (DEM). It is also referred to as laser ranging, laser altimetry, laser scanning, and LADAR (LAser Detection And Ranging). Its development can be traced back to 1970s and 1980s, with an early NASA system and other attempts in USA and Canada. While airborne laser ranging systems are not new, development of a system consisting of a laser scanner, global positioning system (GPS), and an inertial measuring unit (IMU) that can meet map accuracy has been a relatively recent occurrence. This technology is in its infancy and it is experiencing rapid evolution as the market for this technology grows. Not only can it be used for DTM/DEM data collection, but it also has been used for a myriad of other studies including determining vegetation volume and atmospheric studies to feature extraction. In the following section, the basic information and major features of LIDAR System are introduced.

1.1 What is LIDAR

LIDAR, as it has been mentioned, is actually a system that integrates three basic data collection tools: laser scanner, GPS, and IMU. As shown in Figure 1, the LIDAR system is mounted on an aircraft just like an aerial camera. It sends out an infrared laser signal to the ground that is then reflected back to the instrument. The time it takes the laser signal to complete this trip is recorded. Since the signal makes the trip from the aircraft to the ground twice (once to the ground and then back to the aircraft), the total distance is divided by two and this is then multiplied by the speed of light to obtain the distance. IMU is used to record the angle α at which the laser signal is sent out. GPS is used to determine the coordinate value (X, Y, H) of the LIDAR system. Then, using simple trigonometry, the horizontal coordinates and elevation values (x', y', z') of the detected ground point can be derived.

One of the advantages of a laser signal is that the beam is very narrow but geta larger the farther away the system is from the target ground source. Therefore, we should recognize that the LIDAR signal is not a point but rather is an area with a certain size. Hence, when a laser signal is sent to the earth, it can easily hit more than one object. For example, Figure 2 shows a laser pulse heading towards the ground. A part of the signal first encounters a part of the foliage while the rest of the signal hits the ground. Depending on how the system

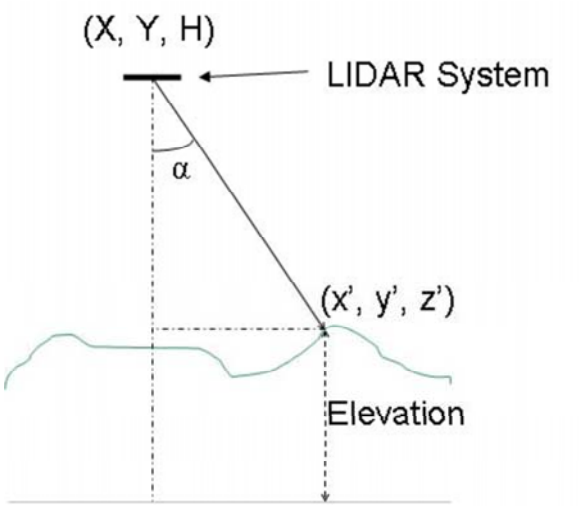


Figure 1: Detection of coordinate values of target points by the LIDAR system

is set up, the LIDAR scanner can, as an example, collect both of these data pulses. They are commonly called the first pulse or first return (the portion of the signal striking the foliage) and the last pulse or last return (the portion of the signal hitting the ground). In some systems, it is possible to collect more than two different returns. The measurement elevations of the first and second returns have very little difference on the building areas. On the areas covered by vegetations, the laser signal can partly penetrate through the vegetation cover and form a large difference on the elevation from the first and second returns.

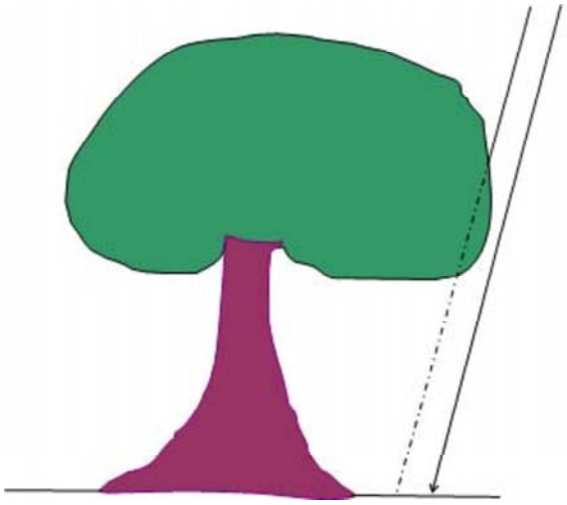


Figure 2: Lasersignal hitting multiple objects

The number of pulses sent out by the scanner is referred to as the pulse repetition rate (PRF) and is measured in kHz. Thus, 10 kHz means that 10,000 pulses are being emitted per second. PRF is very important to DTM generation. Present measuring rates lie between 2 kHz and 25 kHz. Accordingly, from 1000m flying height, the sampling densities on the ground range from 1 point per 20 m^2 up to 20 points per m^2. The actual

sampling density, which is one major concern for the post processing of LIDAR measurements, depends on PRF together with LIDAR scanner, the flying speed, scan angle and flying height.

Like other data sources used to generate DTM, LIDAR data also contains some errors which have a great impact on the post processing of LIDAR data. The conservative estimation of the accuracy of LIDAR is about 15 centimeters in elevation. It is very difficult to assess horizontal accuracy with LIDAR, but the general thought within the industry is that it is about 1.5 times the vertical accuracy. The error information will be mentioned frequently later and its impact on the proposed algorithms will be analyzed.

A new LIDAR system has emerged recently by incorporating the LIDAR system and photography together. The spatial information provided by LIDAR and the image provided by photography are combined and some disadvantages of both methods are overcome. It makes the tasks like the virtual city reconstruction become much more promising in the near future.

Currently, many applications have been developed based on LIDAR. Some of them are listed as follows:

- DTM and Digital Surface Model (DSM) generation in urban areas, automated building extraction, and generation of 3-D virtual city models for movie, video and computer game production, computer animation for natural disasters
- Derivation of vegetation parameters, e.g., tree height, crown diameter, tree density, determination of forest borders
- Improve the mapping accuracy by comparing and verifying the geometrical objects filtered from LIDAR, such as roads, buildings, trees, lakes, pipelines and terrain with existing ones
- Rapid damage assessment after natural disasters, such as hurricanes, earthquakes, landslides, floods, etc
- Measurement of coastal areas, including dunes and tidal flats, determination of coastal change and erosion

1.2 Comparison of LIDAR with other techniques

Photography and LIDAR are competitive in many ways since both of them are used to reconstruct buildings, generate DTM and a host of other tasks. But, neither of them can beat the other in all situations and both of them confront some common difficulties. For example, some buildings may be partly hidden by the surrounding trees which causes incomplete buildings. The advantages and disadvantages of them in DTM generation and building reconstruction are discussed as below.

As far as DTM generation is concerned, LIDAR has big advantages over photography because LIDAR can determine the height of a surface point through a single range measurement and then filter out DTM. However,

photography requires the surface point be identifiable on at least two photographs to determine the height of surface point. In other words, stereoscopy is needed. Even the requirement is satisfied, the performance is still not good when vegetation or tall buildings are present within the imagery. Without accurate height information, it is very difficult to derive the DTM. As [55] points out, surfaces like sand, snow, ice and water bodies can create formidable obstacles for surface measurements using photography. LIDAR works very well under these types of conditions.

As far as buildings are concerned, both methods have supplementary properties. LIDAR provides accurate height information which helps separate different kinds of objects, such as buildings and trees. However, the LIDAR measurements are not dense enough and irregularly spaced. Voluminous data pose a new challenge for automated extraction of building information from LIDAR measurements because many raster image processing techniques cannot be directly applied to irregularly spaced points. Features like break lines and roof ridges are not perfectly captured. Other features like roads have no unique characteristic in LIDAR and are difficult to identify. On the contrary, photography provides less accurate height information, color and texture information, from which it is very difficult to separate different objects from them. For example, the shadows in photography will hide other features, such as buildings, trees and terrain. Colors of certain features may not be distinguishable from the background colors. The poor feature identification result from photography makes it very difficult to create 3D relationship between objects. The advantage of photography over LIDAR is that its denser sampling rate and color information makes small features more obvious. For example, human beings can easily locate roads and small features, such as break lines, roof ridges, pipelines, chimneys on the building roofs from photography.

1.3 Motivation

In 2000-2002, the International Hurricane Research Center (IHRC) at Florida International University (FIU) collected LIDAR measurements in eastern Broward and Palm Beach Counties to assist emergency management personnel in revising their hurricane evacuation maps. Elevations were collected with an Optech ALTM 1210 LIDAR mapping system jointly owned and operated by FIU and the University of Florida. Data were collected as a series 600-meter-wide swaths consisting of points spaced approximately every 2.5 m (8 ft) as shown in Figure 3. Beneath the flight path, flight lines were spaced 500 m apart to allow sufficient overlap in order to avoid data gaps and to assess measurement repeatability. Each deployment typically took 4-5 hours during which GPS data were continuously recorded on both the aircraft and on the ground. In total 160 separate swaths consisting of over 700 million measurements were collected.

Our objective is to overcome some disadvantages of LIDAR and develop a system to filter out different geometrical features automatically. These features will be utilized by the hurricane animation project developed by international hurricane research center (IHRC). The animation result will be broadcast to give the

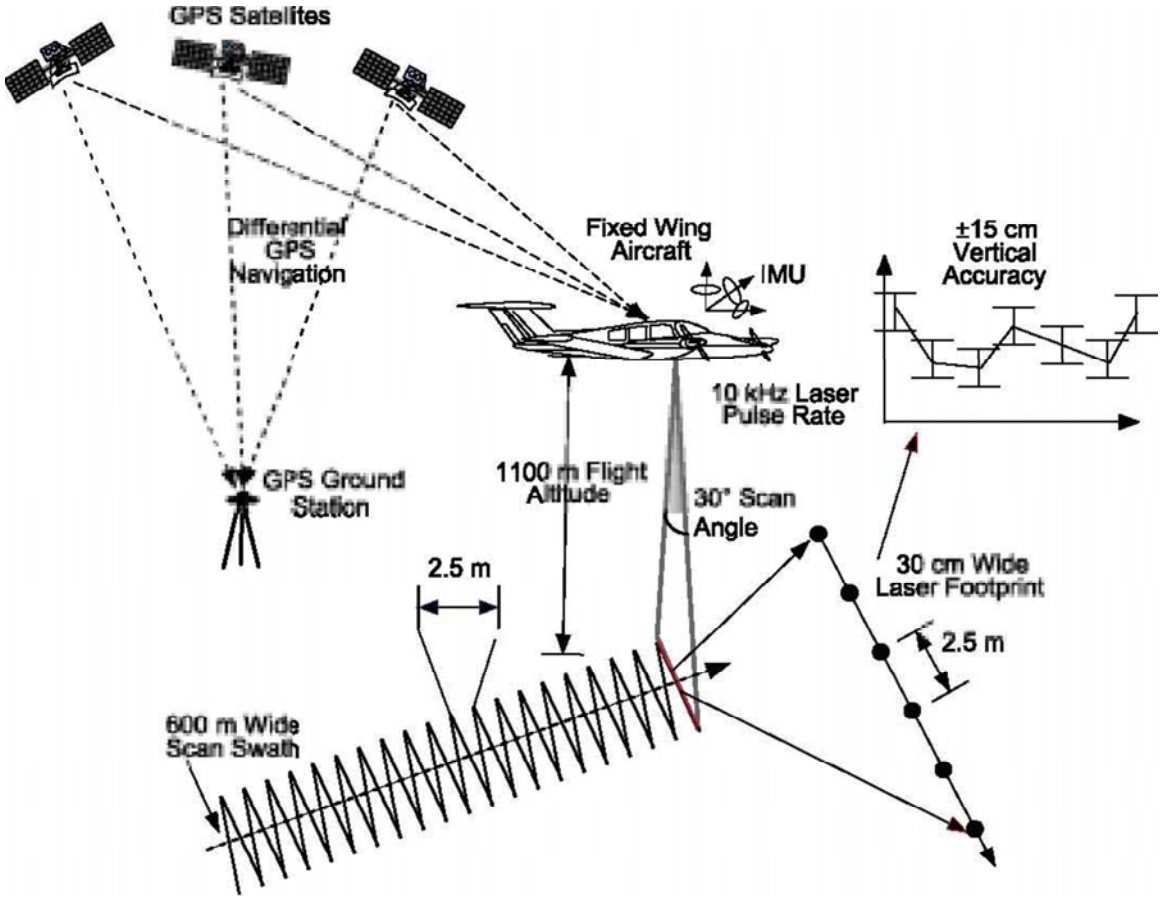


Figure 3: Schematic diagrams showing the components of a typical airborne LIDAR system and the acquisition parameters used in the FIU data collection in 2000-2002

public some warning information for the upcoming hurricanes in South Florida. Currently, we are mainly focusing on DTM and building reconstruction, which have major impacts for the animation performance.

1.4 Contributions

The main contributions of this dissertation are as follows:

1. First, a new algorithm called progressive morphology [34] is proposed to generate DTM. Different from other existing methods, this algorithm can be used to remove different sizes of objects on the ground automatically without any human interaction. Only few parameters are involved in the algorithm which makes the method very simple and robust. The method has been testified on many datasets and achieves good results.

2. Second, we developed a method based on area growing [35] to separate buildings from other non-ground objects which mainly consist of trees. Most of the existing algorithms to identify building measurements are not thoroughly verified on a huge volume of datasets and have certain limitations to reach good result. Our method is tested on a large number of datasets and proven to work well.

3. Third, we proposed an intuitive method [35] to estimate the domain direction of most buildings, which is a critical step to derive the footprint and 3D building model. Our method is more robust than the other existing algorithms and can be applied to most of the buildings. It has been proven in theory and by many experiments.

4. Fourth, we made some assumptions [35] about buildings and corresponding operations to adjust the footprint. These assumptions are less strict than those proposed by other existing methods. For example, segments oblique to the domain directions of buildings are allowed. Hence, the method can be applied to most of the buildings.

5. Fifth, a method is proposed to extract 2D topology of buildings and no threshold is involved. This method is more concise and robust than existing methods.

6. Sixth, a framework is developed for 3D building model reconstruction. A snake-based method including two energy functions are proposed to adjust the 3D building model.

7. Seventh, a 2D snake algorithm is proposed to efficiently derive the minimal energy of 2D snake problems.

1.5 Scope and Limitations of the Framework

The proposed framework has the following limitations:

- The identified buildings are still not perfect enough, especially when there are small facets, such as pipeline and chimney on the roof. Although these facets account very small parts of the whole building areas, they make great negative impact on the succeeding operations, such as footprint extracting and 3D reconstruction of buildings. The performance can be improved greatly with denser LIDAR scanning measurements. However, it is not realistical to get LIDAR measurements with enough density in many situations. Therefore, it is a meaningful task to detect the small facets on the building roofs under the current LIDAR measurement density.

- The domain direction estimation is based on the Douglas-Peucker algorithm. However, the result of the Douglas-Peucker algorithm is greatly affected by the related threshold. If a larger threshold is selected, some key points on the boundary of the building will be lost. On the contrary, if a smaller threshold is selected, some noise points will be erroneously classified as key points, which affect the direction estimation result. Therefore, it is very important to determine an appropriate threshold for Douglas-Peucker algorithm. Currently, the threshold is determined empirically and set as a constant for all buildings. We need to test more datasets and check whether the threshold is suitable for all buildings.

- Limited to the segmentation result, the reconstructed 3D models for some buildings are not perfect enough. For example, the situation where four or more roof planes intersect at one point cannot be detected and adjusted. They will be analyzed later.

1.6 Outline of the Dissertation Proposal

The organization of the dissertation proposal is as follows. In Chapter 2, the literature reviews are given in the areas of automatic DTM generation, building identification and footprint extracting from LIDAR. Chapter 3 describes the proposed progressive morphology algorithm for DTM generation. In Chapter 4, an algorithm based on area growing is introduced to separate buildings from other non-ground objects and derive the simple building model consisting of building footprint and average height. Chapter 5 introduces a framework to automatically derive the 2D topology and reconstruct 3D building model consisting of roof planes on each building. Most of 3D building model can be adjusted by using the 2D snake algorithm described in Chapter 5ch:2dsnake. Chapter 7 gives the conclusion along with the proposed future work.

CHAPTER 2
LITERATURE REVIEW

In this chapter, the existing approaches and methodologies used to filter out some major geometrical features from LIDAR data, such as terrain, buildings and trees, are summarized. Two ways are often utilized. One is to separate ground, buildings, trees, and other measurements from LIDAR data simultaneously. Examples of this method can be found in Maas [41] and Filin [18]. A more popular way is to separate the ground from non-ground LIDAR measurements first and then identify the building and tree points from non-ground measurements. Finally, detailed information about buildings and trees can be filtered out.

2.1 DTM generation

High-resolution Digital Terrain Models (DTMs) are essential for much Geographic Information System (GIS) related analysis and visualization. The airborne LIDAR technology is revolutionizing our way to acquire a high-resolution DTM by allowing rapid and inexpensive measurements of topography over a large area. These measurements generate a 3-dimensional cloud of points with irregular spacing. The laser-scanned objects include buildings, vehicles, vegetation, and "bare ground." To generate a DTM, measurements from ground and non-ground features have to be identified and classified. Removing non-ground points from LIDAR data sets is called filtering and has been proven to be a challenging task.

Kraus and Pfeifer [36] [37] utilized the linear least square interpolation iteratively to remove tree measurements in forest areas. The method was expanded later to filter buildings and trees in urban areas by Pfeifer et al. [44]. The iterative linear interpolation requires that a reduced elevation value at any point, which is obtained by removing a low degree polynomial trend surface from the original elevation at that point, follows a purely random process. However, the required ergodic property of the pure random process is hard to satisfy in urban areas where significant anthropogenic modification of natural terrain occurs. Therefore, the iterative linear interpolation is not guaranteed to converge when being applied to LIDAR measurements for these areas.

Vosselman [59] proposed a slope-based filter that identifies ground data by comparing slopes between a LIDAR point and its neighbors. A point is classified as a ground measurement if the maximum value of slopes between this point and any other point within a given circle is less than a predefined threshold. The

lower the threshold slope, the more objects will be removed. A reasonable threshold slope for a certain area can be obtained by using prior knowledge about terrain in the study area.

There are two basic errors in classifying LIDAR measurements by virtually any filtering method. One is commission error that classifies non-ground points as ground measurements [11]. The other is omission error that removes ground points mistakenly. The critical step in slope-based filtering is to determine an optimum threshold so that omission and commission errors can be minimized. Determining a slope threshold in terms of terrain information in the analyzed area is somewhat subjective. [59] demonstrated that a better filtering result could be obtained by using training data sets. However, the training data sets have to include all types of ground measurements in a study area to achieve good results, which is not always practical. Other attempts to improve the slope-based filter can be found in [48] and [54].

The implicit premise of applying the slope-based filter is that there is a distinct difference between the slope of terrain and that of non-ground objects such as trees and buildings. Satisfactory results have been achieved in our experiments by applying the slope-based filter to flat urban areas such as Miami, Florida. However, both omission and commission errors were large when this method was applied to vegetated mountain areas with a large slope variation.

Haugerud and Harding [25] developed an algorithm to filter tree points in forest areas by comparing local curvatures of point measurements. Ground measurements were selected by removing tree vertices iteratively from a Triangulated Irregular Network (TIN) constructed from LIDAR measurements. Alternatively, ground points can be classified by iteratively selecting ground measurements from an original data set. Axelsson [6] suggested adaptive TIN models to find ground points in urban areas. First, seed ground points within a user-defined grid of a size greater than the largest non-ground features are selected to compose an initial ground data set. Then, one point above each TIN facet is added to the ground data set every iteration if its parameters are below threshold values. The iteration continues until no points can be added to the ground data set. The problem with the adaptive TIN method is that different thresholds have to be given for various land cover types.

Another commonly used algorithm to remove non-ground objects is a mathematical morphology filter applied to a gray scale image [27] [46]. The elevation of trees, cars, and buildings is usually higher than that of surrounding ground points. If LIDAR points are converted into a regular, gray scale grid image in terms of elevation, then the shapes of buildings, cars, and trees can be identified by the change of gray tone. It is well known that compositions of algebraic set operations based on mathematical morphology can be used to identify objects in a gray scale image [24]. Therefore, mathematical morphology can be used to filter LIDAR data.

Mathematical morphology composes operations based on set theory to extract features from an image.

Two fundamental operations, *dilation* and *erosion*, are commonly employed to enlarge (dilate) or reduce (erode) the size of features in binary images. Dilation and erosion operations may be combined to produce *opening* and *closing* operations. The concept of erosion and dilation has been extended to multi-level (grayscale) images and corresponds to finding the minimum or maximum pixel value, respectively, within a specified neighborhood of each raster [47].

These concepts can also be extended to the analysis of a continuous surface such as a digital surface model as measured by LIDAR data. Consider a LIDAR measurement *p(x, y, z)*, the dilation of elevation z at x and y is defined as:

$$d_p = \max_{(x_p, y_p) \in w}(z_p) \tag{1}$$

Where points (x_p, y_p, z_p) represent p's neighbors (coordinates) within a window, w. The window can be a one-dimensional line or two-dimensional rectangle or other shapes. The dilation output is the maximum elevation value in the neighborhood of p. Erosion is a counterpart of dilation and is defined as:

$$e_p = \min_{(x_p, y_p) \in w}(z_p) \tag{2}$$

The combination of erosion and dilation generates opening and closing operations that are employed to filter LIDAR data. The opening operation is achieved by performing an erosion of the data set followed by dilation, while the closing operation is accomplished by carrying out a dilation first and then an erosion. Figure 4 shows the result of performing an opening operation using a line window. As the result demonstrates, an erosion operation removed tree objects of sizes smaller than the window size, while the dilation restored the shapes of large building objects. The ability of an opening operation to preserve features larger than window size is very useful in some applications. For example, the measurements for cliff edges can be preserved if the morphological filters are applied to the LIDAR measurements for rocky coasts.

Kilian et al. [27] proposed a method to remove non-ground points using a morphological filter. In their method, a point with the lowest elevation within a given window size is detected first. Then the points in this window that fall within a band above the lowest elevation are selected as ground points. The range of the band is determined by the accuracy of the LIDAR survey, which is normally 20-30 cm. All ground points are identified by moving the filtering window over the entire data set.

The selection of a filtering window size and the distribution of the buildings and trees in a specific area are critical for the success of this method. If a small window size is used in Kilian's method, most of the ground points will be preserved. However, only small non-ground objects such as cars and trees will be effectively removed. The points corresponding to the tops of large-sized building complexes that often exist in urban

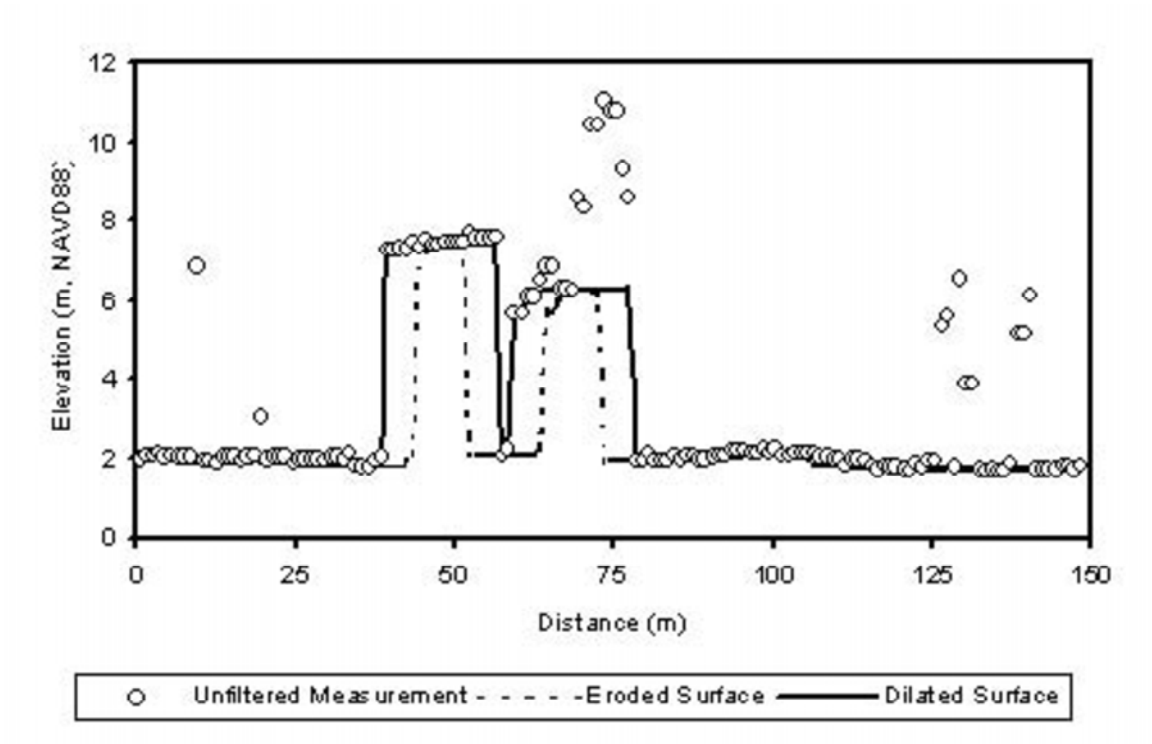


Figure 4: Unfiltered and filtered LIDAR measurements along a profile at Florida International University campus. The unfiltered points are sampled every 1×1 m^2 cell along the profile. If more than one measurement falls within a cell, the point with the minimum elevation is selected. If there is no measurement for a cell, nearest neighborhood interpolation is used to derive an elevation. The filtered data are obtained by applying an opening operation with a window size of 11 m. The profile location is shown in Figure 7. Note tree objects less than the window size are removed by erosion, while the large building objects are restored by the dilation.

areas cannot be removed. The risk of making commission errors is high. On the other hand, the filter tends to over-remove the ground points with a large window size. For example, road surface points next to a canal will be removed completely if the window size is larger than the width of a road. In addition, the tops of local mounds and sand dunes in flat coastal areas are often "chopped off" by using a large size window. Ideally, the window size of the morphological operation should be small enough to preserve all ground details and large enough to remove buildings, cars, and trees. Unfortunately, an ideal window size does not exist in the real world.

To avoid this conflict, Kilian et al. [27] proposed to apply the operations with different window sizes to the data set several times starting from the smallest size. Each point is assigned a weight related to the window size provided it is classified as a ground point. The larger the window size of an operation is, the higher the weight of a point. Finally, the terrain surface is estimated by using all the survey points with assigned weights. Although a better terrain surface could be derived using the weighted points from different sizes of morphological operations, this method does not make an improvement in separating ground and non-ground LIDAR measurements at the point level. Often a terrain model interpolated from identified ground points is preferred for most applications.

Lohmann et al. [46] used a dual rank morphological filter proposed by Echstein and Munkelt [15] to classify LIDAR point data. The dual rank filter initially sorts the neighborhood of a point, p, in terms of elevation to derive a rank function $R(p, i)$, where i ranges from 1 to n_p and n_p is the total number of points of p's neighbors (including p). The neighbors of a point are delineated by a window that is usually a circle and can be any shape. The dual rank filter is then defined as:

$$DR(p, i) = R(p, i) \circ R(p, n_p - i + 1) \tag{3}$$

The symbol "$\circ$" indicates the successive operations: the points are processed by the first rank operation, and then the second rank operation is performed on the results from the first operation. The dual rank filter becomes an opening operation when the rank value is 1 (i.e., i=1), and closing operation when the rank value is n_p ($i = n_p$). Promising results have been achieved by applying a dual rank filter to a test data set [46]. However, the size of a filtering window has to be determined interactively based on prior knowledge of the terrain in the filtered areas. An optimum size is hard to derive because a single fixed-size window of the dual rank filter cannot fit all non-ground objects.

The above morphological filters suffer from various problems such as the requirement of a pre-defined filtering window size, and need to be improved. In addition, for most applications, a highly automatic filtering tool to identify ground measurements is desired due to the large volume of LIDAR data involved. Furthermore, separating ground and non-ground measurements at the point level is preferred so that users

can generate a DTM using the interpolation method that fits their applications best. In chapter 3, a new morphological filter is introduced to remove non-ground measurements from LIDAR data set at the point level.

2.2 Building Identification

Two steps are involved to identify buildings from the LIDAR data set. First, non-ground measurements can be simply derived by removing identified ground data from a raw data set. To facilitate data processing, digital surface models (DSMs) interpolated from raw LIDAR measurements and digital terrain model (DTM) interpolated from ground measurements are often produced. The image for non-ground objects is derived by subtracting DTM from DSM. Examples using this method to derive non-ground objects can be found in Weidner and Forstner [63], Morgan and Tempfli [43], Rottensteiner and Jansa [49], and Elaksher and Bethel [2].

The next step is to extract building point measurements or pixels in the data for non-ground objects which are dominated by trees and buildings. The distinct difference between buildings and trees is that the roof surfaces are approximately planar, while canopy surfaces are irregular. Several parameters based on this observation have been proposed for segmenting buildings and trees. For example, the first derivatives of heights are either a zero (flat roof) or constant (sloped roof) for a planar surface, and the second derivatives of a sloped planar surface are zero. The first and second derivatives of an irregular surface should be variable. Morgan and Tempfli [43] applied Laplacian and Sobel operators to height surface to separate building and tree measurements. The problem with this method is that the derivatives from LIDAR measurements for roofs are not constant because of measurement errors. Small features such as chimneys, water tanks, and pipes on a roof surface can also produce abnormal derivative values. In addition, the derivatives at the edge of a building have a large variation, making it difficult to separate buildings from adjacent trees.

Another technique to separate building and tree measurements involves using a least squares method to estimate the parameters for a plane which fits a LIDAR point and its neighbors within a local window [18] [42]. It is expected that the deviations of roof points from their fitted planes will be small and the plane parameters will be similar and consistent, while deviations and plane parameters for tree points will be large and variable. Compared to derivatives, the plane parameters are less sensitive to individual outliers caused by chimneys and water tanks. The drawback of this method is that plane parameters are not robust at the boundary of the buildings because fewer points are available for parameter estimation.

Many airborne LIDAR systems are capable of deriving the first and last return measurements produced by multiple reflections of a laser pulse by the objects on the earth surface. The height difference between the first and last return measurements can be used to separate building and tree measurements [4]. The height difference is usually large for tree measurements and close to zero for building measurements. A

measurement is identified as a building measurement by comparing its height difference to a predefined threshold. However, this method does not work for areas covered by dense trees where laser pulses cannot penetrate. In addition, the elevation difference between first and last returns less than 3-4 m is not reliable because of the influence of the laser pulse width (typically 10 nanoseconds) [61].

Hough transform has also been used to identify building points from tree measurements [49] or directly from a raw LIDAR data set [60] [30]. Data in the physical space are transformed to and analyzed in the parameter space. The advantage of the Hough transform is its tolerance of gaps in the feature boundary. However, it is difficult to define the optimum cell size for voting in the parameter space, which is influenced by the error range of LIDAR measurements, the sampling density, and local height changes of a roof surface. Unfortunately, local height changes of individual surfaces are different; thus it is difficult to quantify these changes by using a single value. Usually, the cell size is set empirically, and if the cell size is too large, several real-world planes could be merged into one during building identification. In contrast, if the cell size is too small, one real plane could be split into several smaller planes. In addition, the adjacency of point measurements is not considered by the Hough transform. Therefore, LIDAR measurements for separate but closely adjacent buildings with the same height could be mixed into the same category.

The parameters of fitting planes for LIDAR points can be combined with the Hough Transform together to identify building measurements. First, the unit vector of the fitting plane for each LIDAR measurement and its neighbors is evaluated. Then it is mapped to a parameter space. Two parameter spaces are widely used. One of which is a Cartesian coordinate system and the unit vector is described by x, y, and z coordinates in the range between -1 and 1. Another one is unit spherical coordinate system and each unit vector is described by two angles falling in between 0^0 and 360^0. Normally both parameter spaces are evenly divided into cells. The points belonging to the same roof surface will vote for the same cell in the parameter space which will accumulate a large voting number. The plane represented by a cell of the parameter space will be identified as a building roof surface if its voting number exceeds a predefined threshold. Figure 5 demonstrates the identified buildings after applying this method with different cell sizes and parameter spaces on an area on FIU main campus. Figure 5(a) shows the non-ground points filtered by our proposed progressive morphology algorithm, while Figure 5(b) illustrates the buildings extracted based on our proposed area growing algorithm. The parameter space represented by Cartesian coordinate system is divided into cells with size 0.1 and 0.25 and the identified building measurements are demonstrated in Figure 5(c) and 5(d), respectively. The parameter space depicted by the unit spherical coordinate system is divided into cells with size 10^0 and $30^0$and the identified building points are demonstrated in Figure 5(e) and 5(f), respectively. We can see that

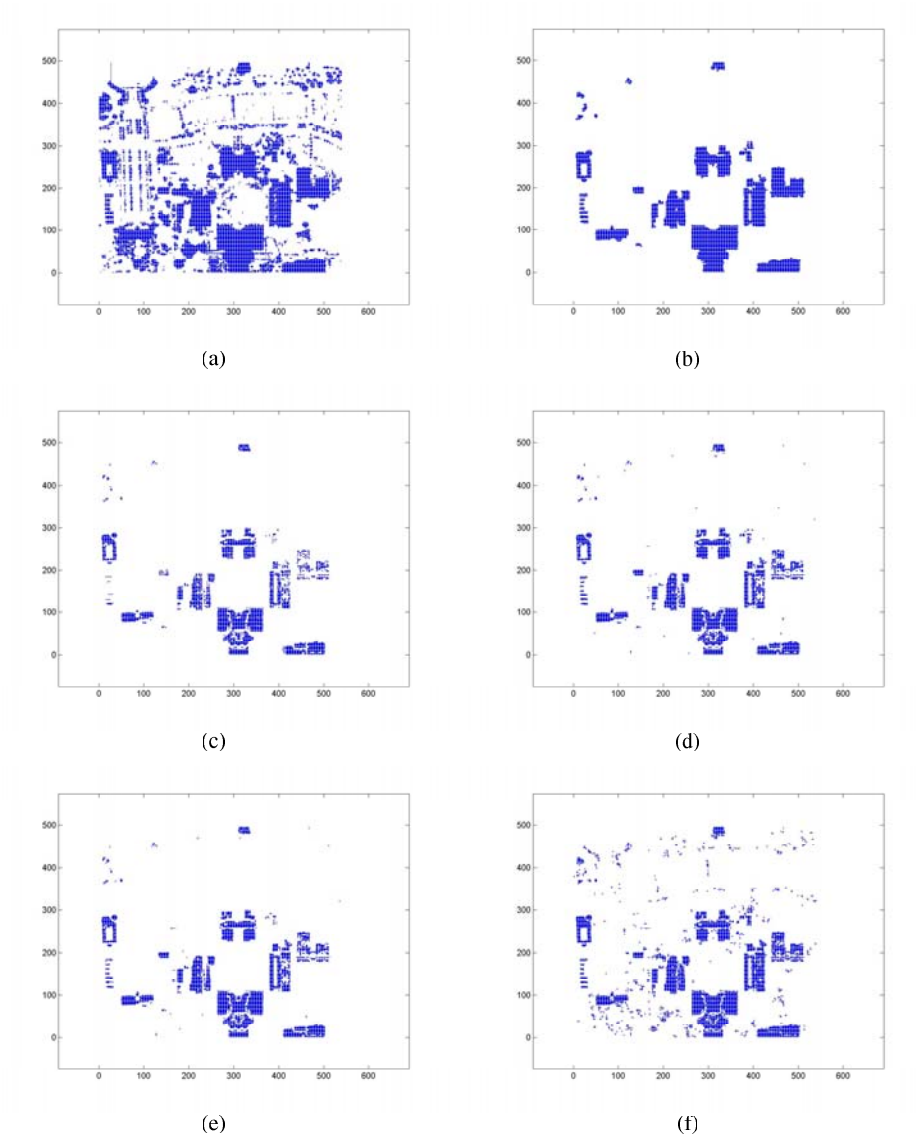


Figure 5: Buildings identified by applying Hough Transform and the plane parameters together (a) Non-ground objects in one area of FIU campus. (b) Buildings extracted by our proposed area growing algorithm. (c) & (d) Buildings identified after applying Hough Transform in the parameter space represented by Cartesian coordinate system with cell size 0.1 and 0.25, respectively. (e) & (f) Buildings identified after applying Hough Transform in the parameter space represented by unit spherical coordinate system with cell size 100 and 300 respectively

the segmentation result based on Hough Transform either contains many holes in the building roofs (Figure 5(c), 5(d) and 5(e)) or contains many noises (Figure 5(f)) compared with the result shown in Figure 5(b), which demonstrates that the method based on Hough Transform is not robust.

The LIDAR measurements for buildings and trees can be segmented based on one or more of the above-mentioned parameters. There are two ways to perform the segmentation task. One is the point or pixel level classification. Maas [41] employed raw elevation, Laplace filtered height, and maximum slope from LIDAR measurements to perform a supervised maximum likelihood classification. Filin [18] separated LIDAR measurements for buildings, vegetation, ground, and other features using an unsupervised classification based on the position of a point, the parameters of the tangent plane to the point, and the relative height difference between the point and its neighbors. Elberink and Maas [16] used unsupervised k-means classification in their texture-based segmentation. The problem with a point level classification is that the measurements for a building cannot be guaranteed to classify into the same category. Also, selection of training data sets for a supervised classification can be very time consuming [16]. An alternative way is to find a building area using a region growing algorithm based on a seed point [42]. This method considers the adjacency of LIDAR measurements. However, the results of region growing processes are variable, depending on the selection of seed points. The effect of selecting seed points on segmentation results has not been examined.

In the chapter 4, we proposed an algorithm which combines the area growing method and parameters for the fitting plane of each LIADR point together. This approach is robust to the selection of seed points and works well at the boundary of buildings in most situations.

2.3 Simple Building Model

Building footprint, height, volume, and three-dimensional (3D) shape information can be used to estimate energy demand, quality of life, urban population, property tax, and surface roughness [31]. These measures are essential for 3D city or urban landscape models, urban flooding prediction, and assessment of urban heat island effect. Traditionally, aerial photographs and high-resolution satellite images were the most effective data sources for extraction of building information. Manual derivation of building geometric data from a remote sensing image for a large area is cost prohibitive and time consuming. Therefore, numerous studies have been done to develop automated methods to extract footprints and 3D shapes of buildings [19] [39]and [45]. However, the success of the automated methods is limited due to the influence of sun shadow and relief displacement of high buildings in remote sensing images.

3D building models can be divided into two categories: simple and sophisticated. Geometric attributes for a simple building model consist of a footprint polygon and a height value. The geometric attributes for a sophisticated building model include not only a footprint polygon, but also planes or other types of surfaces for various parts of the roof as well as their projections (polygons) on the ground plane. Only

one fixed building height exists for a simple model, while building heights of the sophisticated model are variable. The advantage of the simple model is that it requires fewer geometric attributes to delineate a building. The buildings are represented by various types of "boxes" three-dimensionally; therefore, the 3D rendering is fast. The simple building models can be imported to commercial GIS software such as ArcGIS for 3D visualization. The simple building model is sufficient for applications such as numerical modeling of urban flooding and heat island effect, estimating urban population and energy demand, and large scale 3D visualization which does not require the details of buildings. The key to extracting a simple building model from LIDAR measurements is to derive footprints. The building height value can be represented using statistical height values such as mean and median of LIDAR measurements within a footprint.

The boundary of each identified building can be taken as a raw footprint and should be polished. It looks as being "zigzag" which is caused by noises and following reasons:

- Random measurement errors are contained in raw LIDAR measurements, such as 0.2 m elevation error.
- Some building measurements are misclassified as trees and then lost in the identified buildings. Some tree measurements are misclassified as buildings and then mixed with identified buildings. Both of them will affect the boundary of the identified buildings
- LIDAR points are irregularly spaced on the ground. The interpolation based on them will affect the boundary of buildings, which is demonstrated in Figure 2.3. The gray rectangle represents a building and the dash lines constitute an interpolation grid. Assume the nearest-interpolation is applied. The nearest LIDAR measurements surrounding grid points 1, 2 and 5 have the same height values as the building. Then they are classified as building points. Similarly, the grid points 3 and 4 are classified as ground points. The identified boundary of the building is represented by the solid and zigzag line

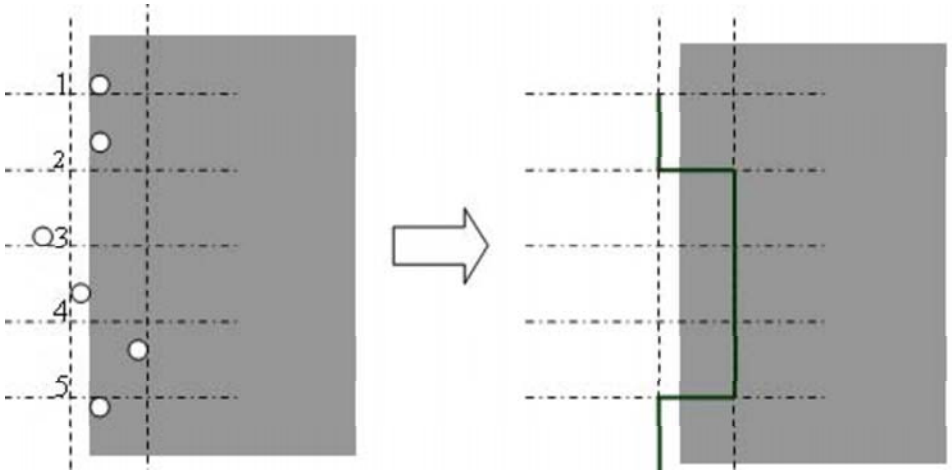


Figure 6: Negative effects caused by the interpolation on the boundary extraction

Some operations, such as boundary simplification and the domain direction estimation are very important to polish the raw footprint.

2.3.1 Boundary Simplification Algorithm

Several algorithms have been employed to simplify a polygon. They can be used to remove the noisy vertices on the raw building footprint and derive a coarse footprint. Latecki and Lakamper [38] proposed a method based on a conspicuous value to simplify polygon. Figure 7 illustrate how to measure the conspicuousness value of a point on a polygon. Assume A, B, C are three consecutive vertices of a polygon. b is the length of segment AC and a is the length of segment BC. τ is the turn angle at C. The conspicuousness value k of C can be measured as follows:

$$k = \tau * ab/(a + b) \tag{4}$$

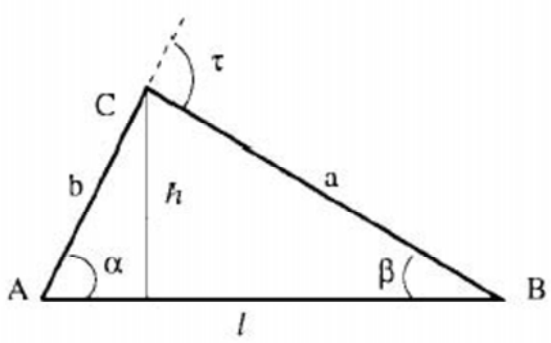


Figure 7: Evaluation of the conspicuousness value k of a vertex

The larger the k value of the point, more important that point is to the polygon. The method takes a top-down approach to extract the conspicuous points. Firstly all the boundary points are taken as conspicuous points. Then the point with least conspicuous value will be removed, because it is least important to the polygon and taken as noise point. Such operation will be iterated until all of the noise points are removed. Figure 8 gives an example on how the method works. Figure 8a demonstrates the original polygon containing some noises. Vertexes with small conspicuous values are removed iteratively and Figure 8b depicts one of the intermediate results. Figure 8c illustrates the polygon after most of the noises are removed. The drawback of this method is that it is difficult to automatically define the termination condition.

The Douglas-Peucker algorithm [14] is another alternative to simplify polygons with noise. It takes a bottom-up approach to locate the key points on a polygon. First, two points are selected as start and end key points. Then a line connecting these two points is formed and the distance of the remaining points to the line is evaluated. If the maximum distance is less than a predefined distance threshold, the algorithm stops. Otherwise, the corresponding point with the maximum distance is taken as a new key point. The same

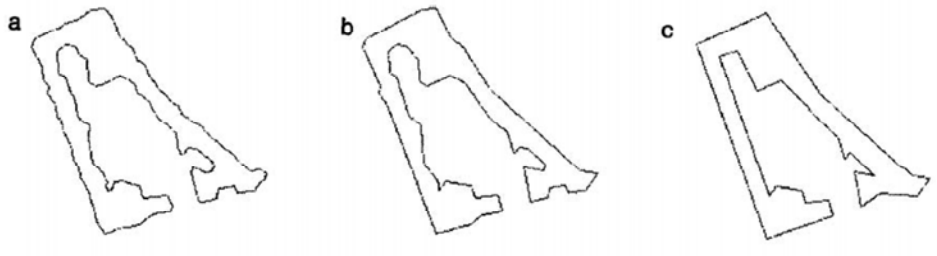


Figure 8: Polygon simplification based on conspicuous value

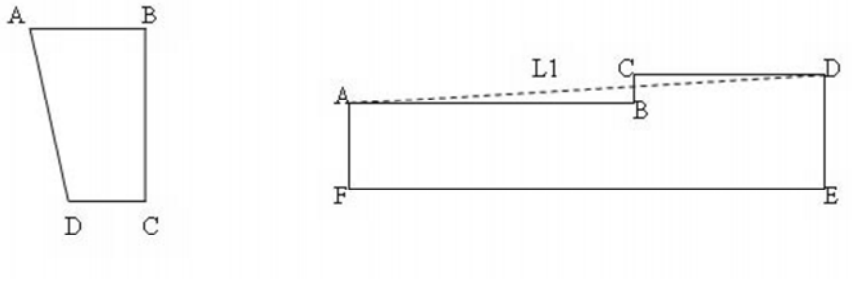
operation is applied on the polygon between each pair of two consecutive key points until no key point can be found any more.

2.3.2 Domain Direction Estimation

The polygon simplification algorithms occasionally removes critical corner vertices of a raw footprint due to the "zig-zag" pattern of boundary lines, thus distorting the orientation of segments and producing acute or obtuse angles between segments. In order to recover the removed critical vertices and distorted segments, the building footprints were divided into two categories. For the first category, there are two dominant directions for a building footprint polygon and most segments are parallel or perpendicular to each other. Most commercial and residential buildings belong to this category. For the second category, more than two dominant directions exist or no dominant directions exist at all. Only a few buildings belong to the second category and they are usually special cases for a study area. The critical step to identify these two categories of building footprints is to estimate dominant directions. Maas and Vosselman [41] derived dominant directions using invariant parameters, based on an assumption that the roof type of a building is already known. However, building roof types are not available in many cases before building footprints are identified [42]. Also, it is often difficult to classify the roofs of complex buildings into any given type.

A 2D Hough Transform can be used to estimate the dominant directions of a building footprint. It is based on the assumption that normally the direction of the longest segment on the footprint polygon is same as the domain directions of the building. Each cell in the parameter space describes the parameters of a line. Each boundary point passing through a line will vote once for the corresponding cell in the parameter space. A global max voting number is reached at the cell corresponding to the longest line segment which can be easily identified. However, the method has two major drawbacks as shown in Figure 9. One of which is that sometimes the direction of the longest line segment is different from either of the domain directions of the building. As shown in Figure 9a, AD is the longest segment on the boundary of the building. AB and BC correspond to the two domain directions of the building. Obviously, the direction of AD does not match with either of the two domain directions. Another disadvantage of the method is that it is very difficult to determine

the optimum cell size of the parameter space, which makes the method not robust. Figure 9b illustrates how a non optimal cell size will affect the performance of the method. Here the footprint of the building consists of corners A, B, C, D, E and F. The directions of EF and AF are the same as the dominant direction of the building. Line segment BC is assumed to be much shorter compared to the other ones. Considering the effect from the cell size of the parameter space, the boundary points located on AB, BC and CD may contribute to the cell corresponding to the line L1 because they are considered as passing through L1 when a bigger cell is selected. The direction of L1 will be falsely taken as the domain direction of the building because the corresponding cell has the maximum voting number.



(a) (b)

Figure 9: The domain direction of a building is erroneously estimated by 2D Hough transform as (a) the direction of line segment AD (b) the direction of line L1

Alharthy and Bethel [4] assume that buildings have only two dominant directions that are perpendicular to each other and all the footprint edges should be parallel to either of the domain directions. They employed the histogram of boundary points to estimate the domain direction. Based on the same assumption, Sampath and Shan [52] used the least squares model to estimate the domain direction. Both methods cannot be applied to buildings with edges oblique to domain directions. In chapter 5, a method based on the weighted line segment lengths is proposed to overcome this problem.

2.4 3D building model reconstruction

The three-dimensional (3-D) building model is one of the fundamental geographical information system data components that can be used to architecture and city planning, and hurricane animation and computer game. Accurate 3-D building models are essential for 3-D city model, urban planning, disaster emergency response and management, urban flooding prediction, and assessment of urban heat island effect.

Traditionally, aerial photographs and high-resolution satellite images are most effective data sources for the extraction of building models. Some methods are proposed to generate 3-D building models from aerial

photograph semi-automatically. For example, [22] [21] introduced a system named as CCM, which generates 2-D topology in manual mode on an analytical plotter or a digital station first, and then reconstruct 3-D building models semi-automatically. Only hundreds of objects can be measured in one day. Derivation of building models manually from a remote sensing image for a large area is cost prohibitive and time consuming. Therefore numerous studies have been done to develop automatic methods to extract 3-D models of buildings [19] [39] [45]. However, the success of the automatic methods is limited because of the influence of the sun shadow and topographic relief of a building in remote sensing images. Most authors admit that their approaches do not work in densely build-up area.

Airborne light detection and ranging (LIDAR) technology provides a promising alternative for measuring buildings. Airborne LIDAR systems derive irregularly spaced 3-D point measurements of objects, including ground, building, trees, and cars scanned by the laser beneath the aircraft. Compared to aerial photographs and satellite images, LIDAR measurements are not influenced by sun shadow and relief displacement. However, voluminous point data pose a new challenge for automated extraction of building information from LIDAR measurements because many raster image processing techniques cannot be directly applied to irregularly spaced points.

In the past decade, a number of approaches have been developed to derive 3-D building models automatically from LIDAR measurements. [17] categorized these methods as model-driven and data-driven ones. Model-driven method fit some primitive building models into the LIDAR measurements. [41] estimates parameters of primitive building based on invariant moment analysis. [7] and [8] extended the method to complex buildings by splitting their ground plan into simple building primitives and modeling these primitives individually. However, ground plan is not always available, which limits the application of the model-driven method.

A data-driven method normally consists of three steps to reconstruct a 3-D building model. First, the building measurements are identified and grouped into different roof planes. Then 2-D topology of each building is derived and represented by a set of connected plane surfaces projected onto a horizontal plane. Finally 2-D topology and roof planes are refined by enforcing geometric constraints and 3-D building models are reconstructed based on refined 2-D topology and roof planes. Some authors [35] developed algorithms to identify building measurements and roof planes. The next step is to derive 2-D topology.

2-D topology can be derived from identified roof planes. [50] approximates pixels on edges with line segments first and then intersect these line segments to derive the vertices on the 2-D topology. The details about how to approximate these edge pixels and intersect line segments are not thoroughly discussed. [4] derives the raw footprint of each roof plane by connecting its boundary points and simplifies the raw footprint by applying the Douglas-Peuker algorithm [14] to it. Simplified footprints of each roof plane are snapped into

2-D topology. Since neighboring footprints may overlap or have gaps, it is not easy to snap them together. The details and performance about the snap operation are not thoroughly presented in [4].

3-D building models can be reconstructed based on 2-D topologies and identified roof planes directly. The quality of reconstructed 3-D building model is affected by two factors. 2-D topology is noisy because of irregularly spaced LIDAR measurements and the estimated roof plane parameters are not precise enough because of the LIADR measurement error and segmentation error. Many geometric constraints have been proposed to modify and regularize the 2-D topology. Some methods enforce a number of constraints at the same time. [21] proposed seven constraints formulated as weighted observation equations and enforce them by using least squares adjustment (LSA). Before LSA is applied, points, lines or planes related to each constraint should be grouped together manually, which limits the application of this method. The performance of enforcing so many constraints one time is not demonstrated. Other methods realized the difficulty of enforcing a large number of constraints at the same time and just enforced important constrains, especially parallelism, which assumes that each building has two domain directions perpendicular to each other and most of edges on the 2-D topology should be parallel to one of them. These methods estimate the domain directions first and then adjust the edges on the 2-D topology according to the domain directions. Thus domain direction estimation is critical to these methods.

[60] assumes that all edges are parallel to either of the domain directions and takes the direction of the intersection line segment of neighboring roof planes as one of the domain direction. The assumption is too strict and can not be applied to buildings with edges inclined with both domain directions. Meanwhile, the method did not consider the estimation errors of roof plane parameters. [35] proposed a method to overcome the aforementioned problems and a footprint outline is utilized. Data source like 2-D topology and roof planes are not used and may help enhance the estimation result.

After the domain direction is estimated, most of edges should be adjusted to be parallel to one of domain directions. [35] proposed some operations to refine the footprint iteratively. However, these operations are limited to 1D topology, such as footprint outline, and are difficult to extend to 2-D topology. [50] adjust some edges by replacing them with the intersection line segments of the neighboring planes. However, this method cannot be applied to buildings mainly consisting of flat roof planes.

CHAPTER 3

DTM GENERATION

In this chapter, a method called progressive morphology is introduced to remove the non ground measurements and generate DTM.

3.1 Progressive morphology

It has been shown that morphological filters can remove measurements for buildings and trees from LIDAR data [27], but it is difficult to detect all non-ground objects of various sizes by using a fixed filtering window size. This problem can be solved by varying window sizes in morphological filters. By performing an opening operation to a laser-scanned data with a line window that increases in size gradually, the progressive morphological filter can extract buildings and trees at various sizes from LIDAR data set.

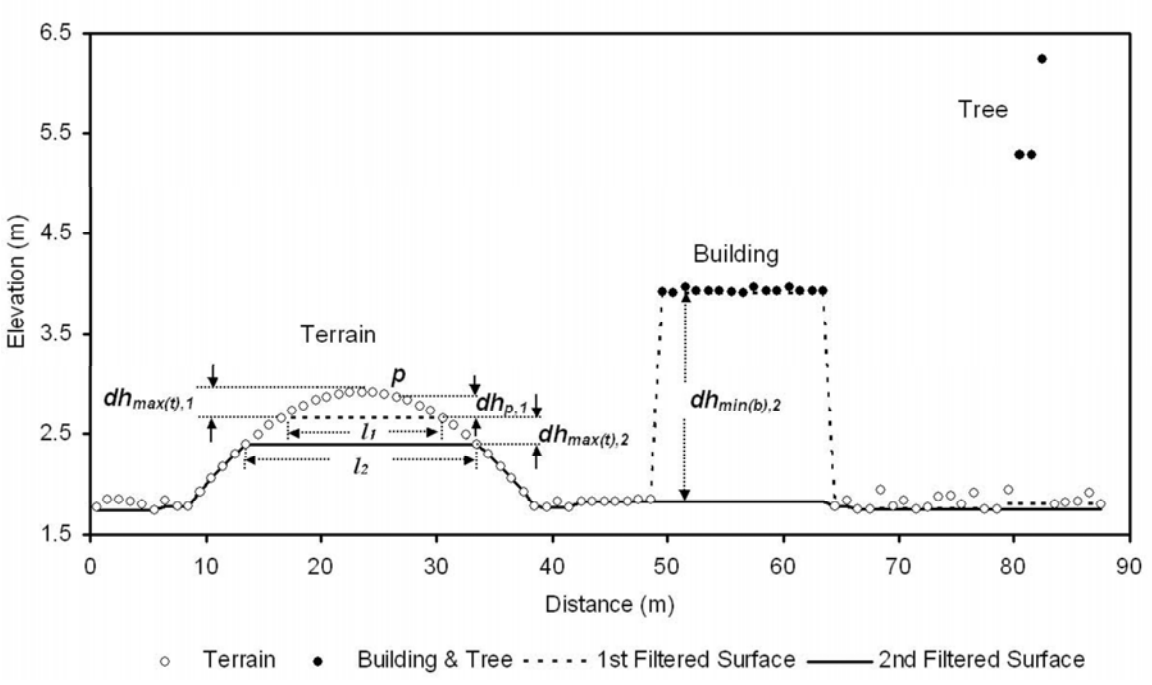


Figure 10: The process of the progressive morphological filter to identify terrain and building objects. The dots represent synthetic points based on LIDAR surveys. The first filtered elevation surface (dashed line) is obtained by applying an opening operation with window size of 15 m (l_1) to the raw point data. The second filtered elevation surface (solid line) is derived by applying an opening operation with window size of 21 m (l_2) to the first filtered surface

Figure 10 illustrates the process of a progressive morphological filter. The first filtered surface is derived by applying an opening operation with a window l_1 to the raw data. The building is preserved after filtering because its size is larger than that of l_1. Since the peak of the terrain is smaller than the size of l_1, it has been cut off and replaced by minimum elevation within l_1 on the first filtered surface. Therefore, the LIDAR measurements from the top area of the terrain would be removed if the opening operation were performed on the data directly. In addition, the filtered data in the ground area are usually lower than the original measurements. These problems can be overcome by introducing an elevation difference threshold.

Each building has certain size and height. There is an abrupt change in elevation between the roof and base of a building, while the elevation changes of the terrain are gradual (Figure 10). The difference in the elevation change patterns between buildings and terrain can help the filter to separate the building and ground measurements. Suppose that $dh_{p,1}$ represents the height difference between the original terrain measurements and the filtered data in the initial iteration at any given point p (Figure 10), and $dh_{T,1}$ represents elevation difference threshold. Point p is classified as a ground measurement if $dh_{p,1} \leq dh_{T,1}$, and as a non-ground measurement if $dh_{p,1} > dh_{T,1}$. Let $dh_{max(t),1}$ stand for the maximum height difference between the original terrain measurements and the filtered data in the initial iteration (Figure 10). If a $dh_{T,1}$ is selected such that the $dh_{max(t),1}$ for terrain are less than $dh_{T,1}$, then the LIDAR measurements for terrain will be preserved. Here, it is assumed that the $dh_{T,1}$ value that is related to the window size of the filter is given. How to derive this threshold value will be discussed later.

In the next iteration, we increase the window size to $l_2$and apply another opening operation to the filtered terrain and building in Figure 10. The terrain has been further smoothed. The maximum height difference between the previous and this filtering operation is $dh_{max(t),2}$. The ground measurements within $dh_{max(t),2}$ will be preserved assuming that $dh_{max(t),2}$ is smaller than $dh_{T,2}$, which is the elevation difference threshold for the current filtering operation. The building object is removed totally since the size of the building is smaller than the current filtering window size. Suppose that the minimum elevation difference for the building between previous and current filtering operation is represented by $dh_{min(b),2}$, which is approximately the height of the building. The building measurements will be removed assuming that $dh_{min(b),2}$ is larger than $dh_{T,2}$.

Generally, elevation difference threshold $dh_{T,k}$ is set to be the minimum height value of the building objects in an analyzed area at iteration k. Taking $dh_{T,k}$ as the threshold, for any given point p at kth opening operation, we mark p as a ground measurement if $dh_{p,k} \leq dh_{T,k}$, and as a non-ground measurement otherwise. In this way, the measurements for buildings with various sizes can be identified by gradually increasing the widow sizes and applying an opening operation repeatedly until a window size is greater than the size of the largest building. Since the size of a tree is usually less than that of a one story residential house and its elevation is higher, the above building filtering procedure can be applied to tree removal as well.

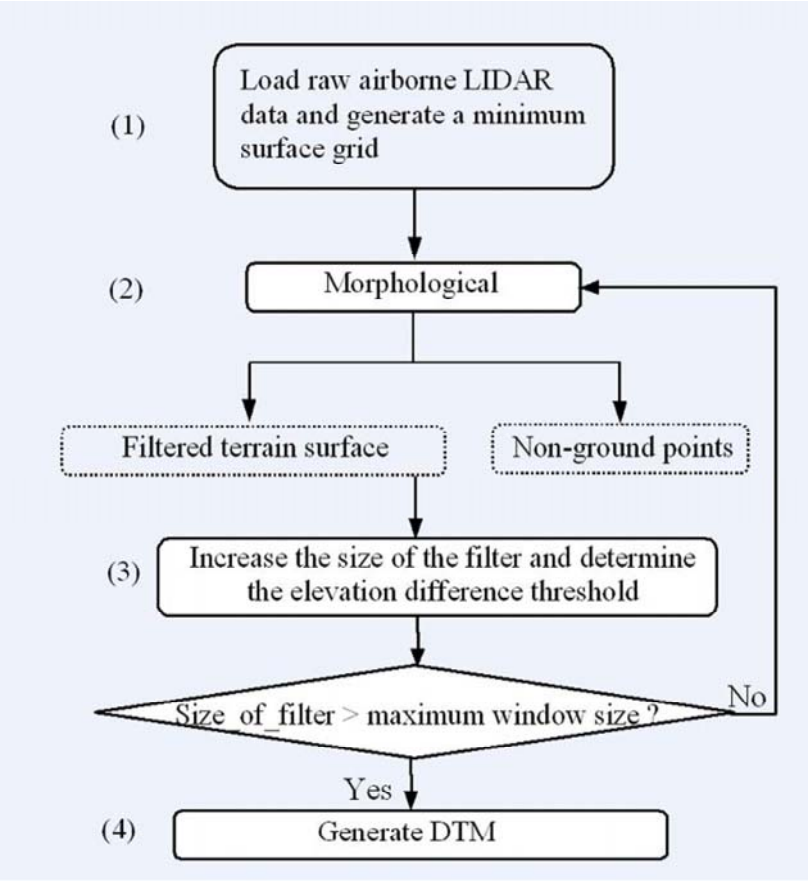


Figure 11: The framework of the progressive morphological filter

The detailed steps to use the progressive morphological filter to construct the DTMs are shown in Figure 11 and given as follows:

Step 1: The irregularly spaced (x,y,z) LIDAR measurements are loaded. A regularly spaced minimum surface grid is constructed by selecting the minimum elevation in each grid cell. If a cell contains no measurements, it is assigned the value of nearest point measurement.

Step 2: The progressive morphological filter whose major component is an opening operation is applied to the grid surface. At the first iteration, the minimum elevation surface together with an initial filtering window size provides the inputs for the morphological filter. In the following iterations, the filtered surface model obtained from the previous iteration and an increased window size from Step 3 are used as input for the filter. The output of this step include a) the approximate surface model from the morphological filter and b) the detected non-ground points. The non-ground points are selected based on an elevation difference threshold.

Step 3: The size of the filter window is increased and the elevation difference threshold is calculated. Steps 2 to 3 are repeated until the size of the filter window is greater than a predefined maximum value. This value is usually set to be slightly larger than the maximum building size.

Step 4: The last step is to generate the DTMs based on the data set after non-ground measurements have been removed.

Note that each cell of the minimum surface grid generated in Step 1 contains an original or interpolated LIDAR point with elevation representing the cell value. Any filtering operation performed on the grid is actually applied to points in cells. Therefore, the progressive morphological filter classifies the LIDAR measurements at the point level. By progressively increasing the filter size (as in Step 3), the non-ground objects such as the cars, trees, and buildings can be identified and removed, while the ground measurements will be preserved. This is the advantage brought by the multi-step progressive filtering process.

3.2 Parameters for Progressive Morphological Filter

The selections of the window size and elevation difference threshold are critical to achieve good results when applying the morphological filter. For window size selection, one straightforward choice is to increase the window size linearly by the following formula

$$w_k = 2kb + 1 \tag{5}$$

Where k =*1,2,...M*, and b is the initial window. The maximum window size (number of cells) is equal to $2Mb+1$. Taking $2kb+1$ as the window size guarantees that the filter window is symmetric around the central point so that the programming of opening operation is simplified. The advantage of increasing the window size linearly is that gradually changing topographic features are well preserved. However, considerable computing time is needed for an area with large non-ground objects.

Alternatively, the window size can be increased exponentially to reduce the number of iterations used in the filter.

$$w_k = 2b^k + 1 \tag{6}$$

where b is the base of an exponential function, $k = 0,1, 2, \ldots M$, and again $2b^M+1$ is equal to the maximum window size.

The elevation difference threshold can be determined based on the slope of topography in the study area. There is a relationship between the maximum elevation difference $dh_{\max(t),k}$ for the terrain, window size w_k, the terrain slope s (Figure 10) assuming that the slope is constant.

$$s = \frac{dh_{\max(t),k}}{(w_k - w_{k-1})/2} \tag{7}$$

Therefore, the elevation threshold $dh_{T,k}$ is given by

where dh_0 is the initial elevation difference threshold, $s$is the slope, c is the cell size, and dh_{max} is the maximum elevation difference threshold.

For flat urban areas, primary non-ground objects include cars, trees, and buildings. The size of individual cars and trees is much less than that of the buildings, so most of them are often removed in the first several iterations, while the large buildings will be removed last. Since the slope of terrain in a flat urban area is small, the maximum elevation threshold dh_{max} can be set to a fixed height (e.g., the lowest building height) to ensure that building complexes are identified. The optimum $s$is usually achieved by iteratively comparing the filtered and unfiltered data. On the other hand, the non-ground objects in the mountain areas are primarily vegetation (trees). There is no need to set up a fixed maximum elevation difference threshold to remove trees, and dh_{max} is usually set up as a largest elevation difference in the study area.

3.3 Algorithm and Implementation

The progressive morphological filter can be either one- or two-dimensional depending on its window shape. The filter is two-dimensional if its window is a two dimensional shape such as rectangle or circle, while the filter is one-dimensional if its window is defined by a segment of a line. The algorithms for one and two-dimensional filters are similar. For simplicity, yet not losing generality, only the input, output and algorithm for the one-dimensional progressive morphological filter using an array are presented as follows. The Erosion and Dilation functions used in the progressive morphological filter are also given.

The above one-dimensional erosion algorithm can be easily expanded into a two-dimensional one with a square window by performing erosion in the x direction first and then in the y direction. The same rule can be applied to dilation as well.

The major computation time needed by the progressive morphology filter is the erosion and dilation in addition to the interpolation. It is easy to see that the opening algorithm requires *O(wN)* time to perform an erosion and dilation, where w is the window size of the morphological filter and N is the product of the number of rows and columns for an array (A) holding LIDAR data. For M windows, the time complexity is equal to

$$O(\sum_{k=1}^{M} w_k N) \tag{8}$$

[58] proposed a recursive dilation/erosion algorithm requiring only 3 comparisons per element whatever the size of the filtering window. Then the time complexity for the opening operation should be:

$$O(\sum_{k=1}^{M} 6 * N) = (6 * M) * N \tag{9}$$

The time complexity can be further decreased. A series of filtering windows $w_i$with increasing size are applied in the progressive morphology filter. Define the raw elevation surface as f and the i'th filtered

Table 1:

Algorithm 1: The progressive morphological filtering algorithm

Input:

- A set of points representing LIDAR measurements. Each point has three components, x, y and z, to represent horizontal coordinates and elevation of a LIDAR measurement.
- Cell size c.
- Parameter $b$in equation 3.1 or 3.2.
- Maximum window size.
- Terrain slope s.
- Initial elevation difference threshold dh_0.
- Maximum elevation difference dh_{max}.

Output:

- Two sets of the classified points representing ground and non-ground measurements.

1. Determine the minimum and maximum x and y values.
2. Determine the numbers of rows (m) and columns (n) using m = floor|(max(y)-min(y))/c|+1 and n = floor|(max(x)-min(x))/c|+1.
3. Create an array $A[m, n]$ for LIDAR points, $p(x, y, z)$. Traverse every point to determine the cell in which the point will fall according to its x and y coordinates. If more than one point falls in the same cell, select the one with minimum elevation.
4. Interpolate elevation of cells in A which do not contain any points using the nearest neighbor method. Set the x and y coordinates of those interpolated cells as zero to distinguish them from those cells that contain LIDAR points. Copy A to array B. Initialize elements of an integer array $flag[m, n]$ with 0.
5. Determine series of w_k using Equation 3.1 or 3.2, where w_k $\leq$ Maximum window size.
6. $dh_T = dh_0$
7. for each window size w_k
8. for i=1 to m
   (a) $P_i = A[i;]$($A[i;]$ represents a row of points at row $i$in$A$andP_i is an one dimensional array)
9. $Z \leftarrow P_i$ (Assign elevation values from $P_i$to an elevation array Z)
10. Z_f = erosion(Z,w_k)
11. Z_f = dilation(Z_f,w_k)
12. $P_i \leftarrow Z_f$ (Replace z values of $P_i$with the values from Z_f)
13. $A[i;]= P_i$ (Put the filtered row of points P_i back to row $i$of array A)
14. for j= 1 to n
15. if $Z[j]$-$Z_f[j]$> dh_T then $flag[i, j] = w_k$
16. end for j loop
17. end for i loop
18. if (dh_T ¡, dh_{max})
19. $dh_T = dh_{max}$
20. else
21. $dh_T = s(w_k - w_{k-1})c+dh_0$
22. end for window size loop
23. for i = 1 to m
24. for j = 1 to n
25. if ($B[i, j](x)$ >0 and $B[i, j](y)$ >0)
26. if ($flag[i, j] = 0$)
27. $B[i, j]$ is a ground point
28. else
29. $B[i, j]$ is a non-ground point
30. end for j loop
31. end for i loop

Table 2: Parameters for the progressive morphological filter. Both the FIU campus and Puget Sound are in meters.

Erosion and Dilation functions Erosion(Z, w_k):	1. for $j = 1$ to n 2. $Z_f[j] = \min_{j-[w_k/2] \le l \le j+[w_k/2]} (Z[l])$ 3. return Z_f
Dilation(Z, w_k):	1. for $l = 1$ to n 2. $Z_f[j] = \max_{j-[w_k/2] \le l \le j+[w_k/2]} (Z[l])$ 3. return Z_f

elevation surface as f_i. According to the progressive morphology filtering algorithm, there is:

$$f_i = f_{i-1} \circ w_i \tag{10}$$

Here $f_0 = f$. We can prove that

$$f_i = f \circ w_i \tag{11}$$

Theorem: Given a signal f and two structure elements X & Y, if $X \subseteq Y$, we have:

$$(f \circ X) \circ Y = f \circ Y \tag{12}$$

Proof:

For the opening operation, there is:

$$(f \circ X) \subseteq f \tag{13}$$

The above equation can derive:

$$(f \circ X) \circ Y \subseteq f \circ Y \tag{14}$$

Combing the property of opening operator with$X \subseteq Y$, there is

$$f \circ X \supseteq f \circ Y \tag{15}$$

The above equation can further derive:

$$(f \circ X) \circ Y \supseteq (f \circ Y) \circ Y \tag{16}$$

The idempotent property of the opening operator gives:

$$(f \circ Y) \circ Y = f \circ Y \tag{17}$$

Combing 3.11, 3.13 with 3.14, there is:

$$f \circ Y \supseteq (f \circ X) \circ Y \supseteq f \circ Y \tag{18}$$

Which is satisfied only when $(f \circ X) \circ Y = f \circ Y$

For the progressive morphology, there is$w_1 < w_2 < ... < w_i$and

$$f_i = f_{i-1} \circ w_i = (f_{i-2} \circ w_{i-1}) \circ w_i = ... = (((f_0 \circ w_1) \circ w_2)...)w_{i-1}) \circ w_i$$

(3.16)

Applying the theorem 3.9 on the above equation recursively, we can derive:

$$f_i = (((f_0 \circ w_1) \circ w_2)...)w_{i-1}) \circ w_i = ((f_0 \circ w_2)...)w_{i-1}) \circ w_i = ... = f_0 \circ w_i$$

(3.17)

Define $d(f, w)$ and $e(f, w)$as dilation and erosion operation on elevation surface f with window w. There is$f_i = f \circ w_i = e(d(f, w_i), w_i)$. In the actual implementation, the window sizes are selected as: $2, 4 \ldots .2M$. Compared to window w_{i-1}, two elements are added into window w_i and two max comparisons per element are required to derive $d(f, w_i)$ from$d(f, w_{i-1})$. For the erosion operation, 3 minimum comparisons per element are required according to [58]. Then the time complexity to derive all the f_i is:

$$M * O(2N) + M * O(3 * N) = O(5 * M * N) \tag{19}$$

3.4 Test Data Set

The morphological filter was tested on two LIDAR data sets: urban settings on low-relief topography and a forested section on high-relief topography. The urban test site, located at the Florida International University (FIU) campus in Miami, Florida, covers about 1.8 km^2 area. There are residential houses, large buildings, single trees, forests, parking lots, open ground, ponds, roads, a major highway, and a canal in this area (Figure 12). The overall slope of the terrain is very gentle except for several small mound areas. The data set was collected by an Optech ALTM 1210 LIDAR mapping system in April 2000. This system was mounted on a Cessna 337 aircraft and jointly operated by Florida International University (FIU) and University of Florida. By flying at a speed of 200 km per hour, altitude of 600 m, off-nadir scan angle of 18°, and laser repetition rate of 10,000 pulses per second, we derived a 400 m wide swath of laser range data. The objects were measured by 15 cm diameter laser beams spaced approximately 2 m apart. Flight lines were spaced 300 m apart to avoid possible data gaps.

Figure 12: Aerial photograph for the University Park campus of Florida International University. 648 random sample points are also overlain over the photograph. The ground and non-ground measurements identified from these samples by the progressive morphological filter are represented by white and black dots, respectively. The white rectangles represent the range of Figure 13 and 14

The test data set for the high-relief area came from the Puget Sound LIDAR Consortium (http://www.pugetsoundlidar.org). The site consists of 1.3 square kilometers of tree covered mountain land. The distribution of trees varies with topography, and usually is dense in the valleys and sparse on the ridges. The terrain varies considerably with slopes ranging from 0.1 to 1.5. The LIDAR data were derived using TerraPoint LLC's laser altimeter, and the data set includes up to four returns for each laser pulse. Each flight surveyed a 600 m wide swath with a 0.9 m diameter laser footprint spaced every 1.5 m. An average point density of 1/m^2 was derived by measuring the area twice with a 50% minimum side-lap between swaths.

3.5 Results

The progressive morphological filter was applied to two test LIDAR data sets to examine its filtering effect. The opening operation was applied to both $x$and$y$ directions at every step to ensure that the non-ground objects were removed. The filtering parameters used in our experiments are listed in Table 3. The window size was incremented by using an exponential function (as defined in Equation 6).

There are about 1,031,230 LIDAR measurements for this area and the number of grid cells to hold the data is 2,021,010. Among them, 711,920 cells have data, and about 30% of the points were removed as repeated measurements for each cell. About 74% of the points in cells having data were classified as ground measurements by the progressive morphological filter.

For high-relief test area, the initial dh_0 was selected to be 0.1 m, and maximum elevation difference threshold was set to be the largest elevation difference value in the analyzed area. The terrain slopes of the

Table 3: Shaded relief maps for the grids generated from unfiltered (a) and filtered (b) LIDAR data from the FIU campus. The grids of cell size 0.5 m were generated by applying Kriging interpolation to LIDAR data with search radius of 100 m in Surfer. Note the effect of the remaining tree points in the filtered data on the DTM (b).

Location	**FIU Campus**	**Puget Sound**
Cell size (c in Equation 3.4)	1	0.3
Base of the exponential window (b in Equation 3.2)	2	2
Increment step for windows (k in Equation 3.2)	0,1, 2, ..., 8	0, 1, 2, ..., 5
Slope (s in Equation 3.4)	0.08	1.2
Initial threshold (dh_0 in Equation 3.4)	0.25	0.1
Maximum threshold (dh_{max} in Equation 3.4)	2.5	210

Puget Sound area in Washington State are relatively steep, ranging from 0.1 to 1.5, and therefore, the s in Equation 3.4 was set to 1.2, which is close to the maximum slope.

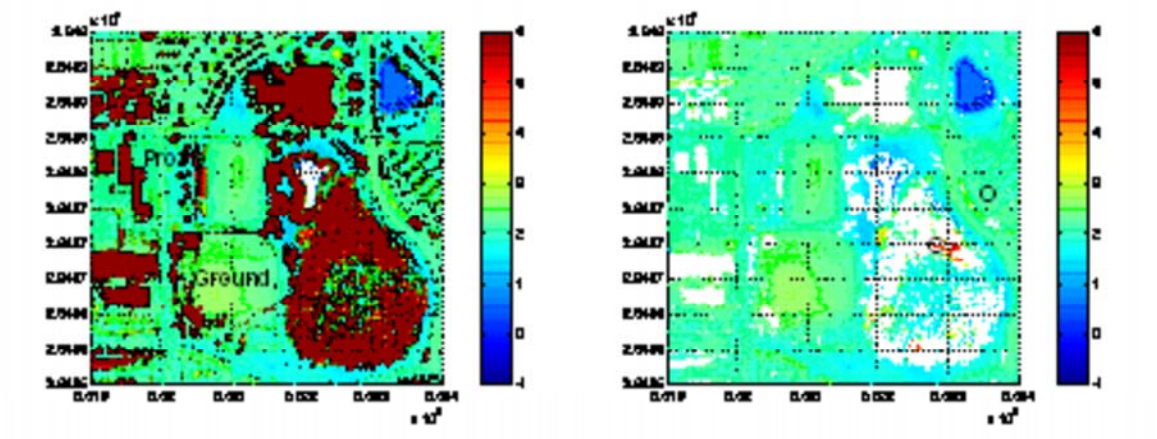
There are about 2,684,240 LIDAR measurements for the mountain area and the number of grid cells to hold the data is 15,005,163. Among them, only 1,274,795 cells have data, and about 53% of the points were removed as repeated measurements for each cell because this data set includes multi-returns of the same laser pulse. 52% of the points in cells having data were classified as ground measurements by the filter.

3.6 Accuracy Analysis

Like other filtering methods, the progressive morphological filter is subject to omission and commission errors. In order to measure the effectiveness of the filter, these two errors have to be examined. Both qualitative and quantitative methods were employed in this study. A qualitative method checks whether non-ground features such as buildings are excluded entirely and ground features like small mounds are included completely by visually comparing the unfiltered and filtered data. The quantitative method examines the correctness of the filtered measurements at the point level from a random sample.

The raw LIDAR data, filtered measurements, black and white aerial orthophotographs, and field investigation were used to quantitatively examine the filtering errors for the FIU campus data set. The aerial photographs were of resolution of 0.3 m and taken in 1999. The evergreen vegetation at the FIU campus changes little through time. Therefore, vegetation and building information from aerial photographs can be used to help identify filtering errors although the aerial photographs were taken at the time different from the LIDAR surveying. Quantitative analysis of filtering accuracy for the mountain data set was not performed because the aerial photographs for the study area are not available and it is too expensive to do a field examination.

Figure 13a and 13b show the unfiltered and filtered LIDAR measurements for an area occupied with cars, single trees, buildings, a small forest and ground. As can be seen from the figures, most of the cars, trees, and buildings were removed successfully by the filter. However, some commission and omission errors did occur. A few measurements for trees remained (**C** in Figure 13b), and a small mound was mistakenly removed (**O** in Figure 13b). The first error occurs because of the high tree density surrounding **C** (Figure 13a). Few



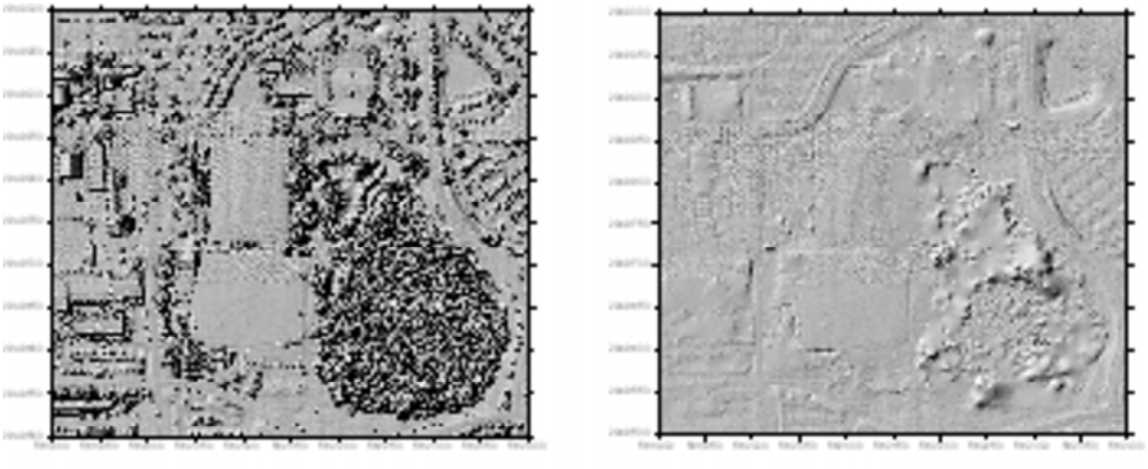
(a) (b)

Figure 13: (a) Unfiltered and (b) filtered LIDAR point measurements for an area at the FIU campus with cars, single trees, buildings, and a small forest. The elevation values higher than 6 m were assigned the same color (red) for display purposes. The horizontal coordinates (x and y) are in UTM Zone 17 referenced to NAD83. Elevation (z) coordinates are referenced to NAVD88. The map units are in meters. Note that the filter removed most of the non-ground objects successfully, but some tree measurements (C) were not removed completely and some ground points were filtered out mistakenly (O).

true ground LIDAR measurements were derived because most laser pulses were reflected by the canopy and did not reach the ground. The elevation changes of some tree tops are similar to those of low topographic relief. Figure 14 shows the shaded relief maps for the grids generated from unfiltered and filtered LIDAR data using Kriging interpolation method [62]. The effect of commission errors on DTM in the forest area is obvious (Figure 14b). The reason for the omission of the small mounds is that the slopes of these mounds are relatively steep, which is larger than s in Equation 7 .

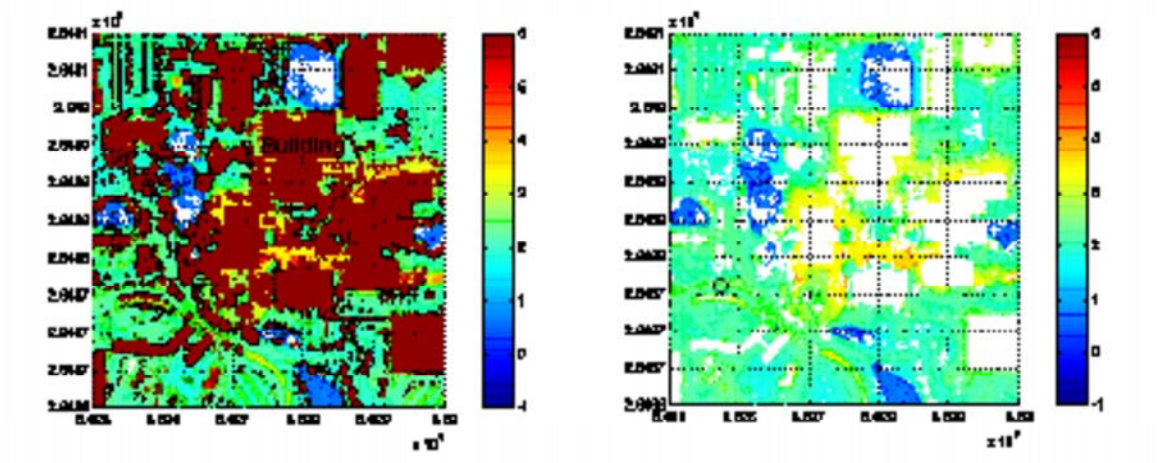
Figure 15 and 16 show the unfiltered and filtered LIDAR point measurements and shaded relief grid maps for a dense building area at the FIU campus. In general, the progressive morphological filter performed well in this area. Omission errors occurred in a few spots because of the complicated composition of buildings and ground objects (**O** in Figure 15b).

A simple random sampling scheme [51] was employed to select the points to quantitatively examine omission and commission errors of the filtering results for the FIU campus data set. First, 1600 random points were generated within a rectangle of 2000 m long (x or east-west direction) and 800 m wide (y or north-south direction) that is the extent of the LIDAR data set. Second, the row and column locations of those 1600 samples in array B, which holds a minimum elevation grid of 1 m cell size interpolated from unfiltered LIDAR data set (Step 4 in Algorithm 1), were determined based on their x and y coordinates. Third, 648 cells of the grid were derived by removing those sample cells which do not have LIDAR measurements. Fourth, the ground measurements in those 648 cells were marked as 1 and the non-ground points were marked as 0 in terms of filtering results. Finally, the 648 measurements were examined point by point manually to find



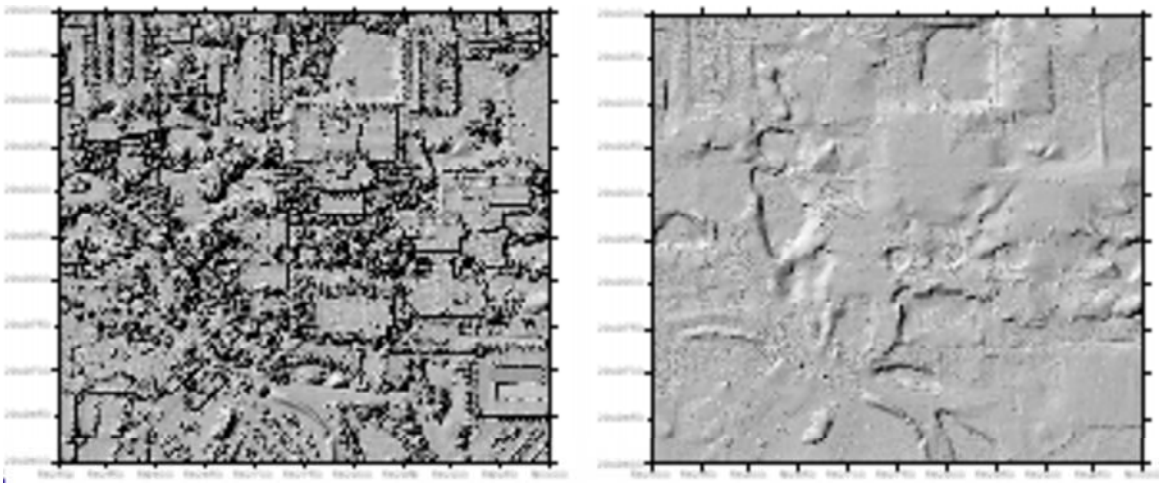
(a) (b)

Figure 14: Shaded relief maps for the grids generated from unfiltered (a) and filtered (b) LIDAR data from the FIU campus. The grids of cell size 0.5 m were generated by applying Kriging interpolation to LIDAR data with search radius of 100 m in Surfer. Note the effect of the remaining tree points in the filtered data on the DTM (b)



(a) (b)

Figure 15: (a) Unfiltered and (b) filtered LIDAR point measurements for an area at the FIU campus with many buildings. The buildings were removed by the filter completely, but some omission errors occurred (O).



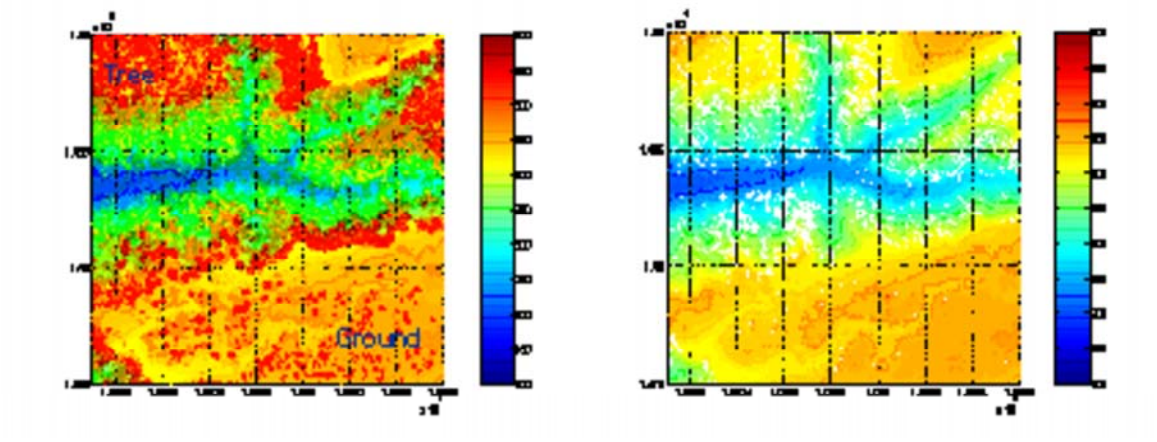
(a) (b)

Figure 16: Shaded relief maps for two grids generated from unfiltered (a) and filtered (b) data at Puget Sound area. The topography of valleys is clear, and linear features such as roads are well preserved in the filtered image

the filtering errors by overlaying them into the aerial photographs (Figure 12) and a digital surface model in ArcGIS. The digital surface model with cell size of 0.5 m was generated by applying a Kriging interpolation to the raw LIDAR measurements. All errors detected from overlay analysis were further examined in the field to avoid possible misinterpretation.

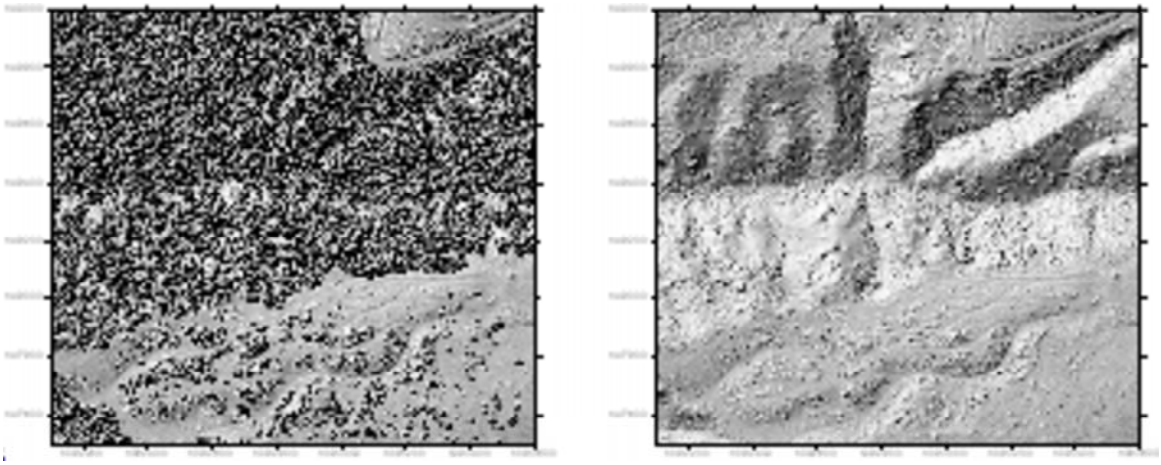
Quantitative error analysis shows that there were 17 omission and 2 commission errors in 648 samples, about 3% of the total, indicating that the progressive morphological filter works well. One reason for having more omission errors than commission errors is that the parameters were set up in such a way that the filter can remove most of the non-ground measurements. The other reason is that the samples are not dense enough to pick up the scattered commission errors in small areas (**C** in Figure 13b). Different sampling sizes and schemes can affect the error analysis considerably [11]. The effect of different sample size and sampling scheme such as cluster sampling on accuracy analysis of the filtering results needs to be studied further in the future.

Comparison of Figure 17a and 17b shows that tree measurements were well removed, while ground points were preserved in the high-relief test area. The shaded relief maps (Figure 18a and 18b) for the two grids generated from unfiltered and filtered data illustrate that the topography of valleys is much clear in the filtered image. Linear features such as roads are also well preserved by the filter. The filtered dataset does contain some commission errors where trees were not entirely removed. These show up as mottled areas on the otherwise smooth topographic surface (Figure 18b).



(a) (b)

Figure 17: (a) Unfiltered and (b) filtered LIDAR measurements for mountains at Puget Sound, Washington State. The unfiltered data include all-return LIDAR measurements. The horizontal coordinates are in Washington State Plane North coordinate system and refer to datum NAD83. Elevations refer to NAVD88. All coordinate units are U.S. Survey Feet. The trees were removed from the LIDAR data set, while the ground measurements were well preserved.



(a) (b)

Figure 18: Shaded relief maps for two grids generated from unfiltered (a) and filtered (b) data at Puget Sound area. The topography of valleys is clear, and linear features such as roads are well preserved in the filtered image

3.7 Discussion

Some LIDAR measurements are removed in generating a regularly spaced minimum elevation grid before the progressive morphological filter is applied to the points. The disadvantage of this implementation is that good ground measurements are removed when a cell includes more than one point. This problem can be minimized by selecting a grid cell size smaller than survey spacing. The removed ground points usually have little effect on DTM generation because LIDAR measurements are so dense in space that major topographic features are rarely missed. The advantage to preprocess the LIDAR to generate a minimum elevation grid is to make the algorithm implementation easy by using an array. In addition, the data set including multi-returns can be handled by selecting a point of minimum elevation in a cell assuming that the lowest point has more chance to be a ground measurement.

The time of filtering increases linearly as the total number of LIDAR measurements increase based on Equation 8 . Computation becomes time-consuming when the number of LIDAR measurements is large. It is impractical to apply the progressive morphological filter to process all measurements as a single file because millions of points were collected for each survey. For efficient processing, the data can be split into tiles with a user-specified size, and then the filter is applied to these tiles.

LIDAR data sometimes may contain a few points with large negative elevation values drastically lower than those of their neighbors. These measurements are called negative blunders, and their source is still in debate [25]. The proposed progressive morphological filter cannot remove negative blunders by only performing the opening operation. When a DTM is interpolated from ground measurements including negative blunders, conical pits like "bomb craters" [25] will be generated. Fortunately, a negative blunder is distinctive from their surroundings in elevation and occupies a small area in space (often as an individual point). Therefore, it can be removed by performing a closing operation immediately following a series of opening operations that are applied to the LIDAR data set as described previously.

CHAPTER 4
SIMPLE BUILDING MODEL CONSTRUCTION

This chapter presents a framework for building footprint extraction from LIDAR measurements and an accuracy analysis for the proposed methods. Building measurements are identified by region growing using a local plane-fitting technique and footprints are derived and adjusted based on estimated dominant directions.

4.1 Identifying Building Measurements

4.1.1 Variance and Unit Vector

After the ground measurements are identified, the heights of the non ground measurements can be derived by subtracting elevations interpolated using identified ground measurements from elevations of non-ground measurements. In order to identify buildings, non-ground measurements whose heights are less than 2 m were removed to minimize the effect of trimmed bushes.

Given a point $p_0(x_0, y_0, z_0)$, a Cartesian coordinate system (x, y and z) is established with p_0 as the origin. In this coordinate system, a best fitting plane for p_0 and its eight neighbors are derived by using the least squares method. Assume that the plane is defined by:

$$z = ax + by + c \tag{20}$$

The parameters (a, b, c) can be derived by minimizing the sum of squares due to deviations (SSD)

$$min(SSD) = min \sum_{(p_k)\in M} (z_k - h_k)^2 \tag{21}$$

Where M is the set for p_0 and its neighbors, and h_k and z_k are observed and plane fitted surface elevations, respectively. The normalized parameter (a, b, c) is termed as the unit vector of the point p_0. The min(SSD) is named as the variance of the point p_0. For each non-ground object area, inside and boundary points are identified. If at least one of the eight neighbors of a point is a ground measurement, the point is defined as a boundary point. Otherwise, the point is an inside point. For the simplicity of implementation, variance and unit vectors are only calculated on inside points. Figure 19 demonstrates the variance and unit vector of inside points of one area in FIU main campus. The area is enclosed by a white rectangle in Figure 19(a). There is a

single building surrounded by several trees in the area, which is shown in Figure 19(b). It is obvious that the inside points on the building roof have very small variances and their unit vectors are similar to each other. On the contrary, points on the surface of trees have larger variances and their unit vectors greatly differ from each other.

4.1.2 Grid Based Area Growing

The inside points are sorted based on their SSDs in ascending order. The point with minimum SSD is labeled and selected as the first seed point for region growing. The neighbors of a seed point are judged whether they belong to the same category through a plane-fitting technique. A plane is constructed based on the points in the category using a least square fit. The elevation from the candidate point to this plane is compared to a predefined threshold Δh_T to select the point. Δh_T is determined by the elevation error of the LIDAR survey and is usually 15-30 cm. If a neighbor is found to be the same category, it is labeled and added to the category. The neighbors of the growing area are examined further and the process is continued until no point can be added into the category. Next, the unlabeled inside points are sorted in terms of their SSDs and region growing starts again from a labeled seed point with a minimum SSD. The region-growing processes are repeated until all measurements are segmented. For the area discussed in Figure 19, 227 patches are formed after area growing and demonstrated in Figure 20(a). Most patches are small and contain very few points. The largest patch with identifier 1 corresponds to the building roof. It grows from the beginning point denoted as 0 in Figure 20(b) and stops after 55 iterations of growing.

After non-ground measurements are segmented, non-building measurements are removed by four post processing steps. First, areas of patches are compared with a predefined area threshold *Min_Surface* (e.g. 5 m^2) to eliminate small and fragmented patches for vegetation. Second, the removed small patches representing chimneys, water tanks, and pipe lines of buildings in the first step are recovered and merged into the large building patches. The condition to include removed small patches is that they are completely within a large patch remained after the first step. Third, dangling points are identified if they do not neighbor with any inside points. They are taken as noises and removed to simplify succeeding operations. Finally, the remaining points are merged in terms of their connectivity. Through this process, adjacent roof surfaces from the same building, having different slopes, are merged into a large building patch. Relatively small areas in the merged patches are removed again by comparing their area values to a threshold *Min_Building* that approximates the minimum area of a building, e.g. 60 m^2. Figure 21 demonstrates the result after applying each step on the segmentation shown in Figure 20. The identified building is shown in Figure 21(d). It is also demonstrated in Figure 19(b) and marked by those blue points.

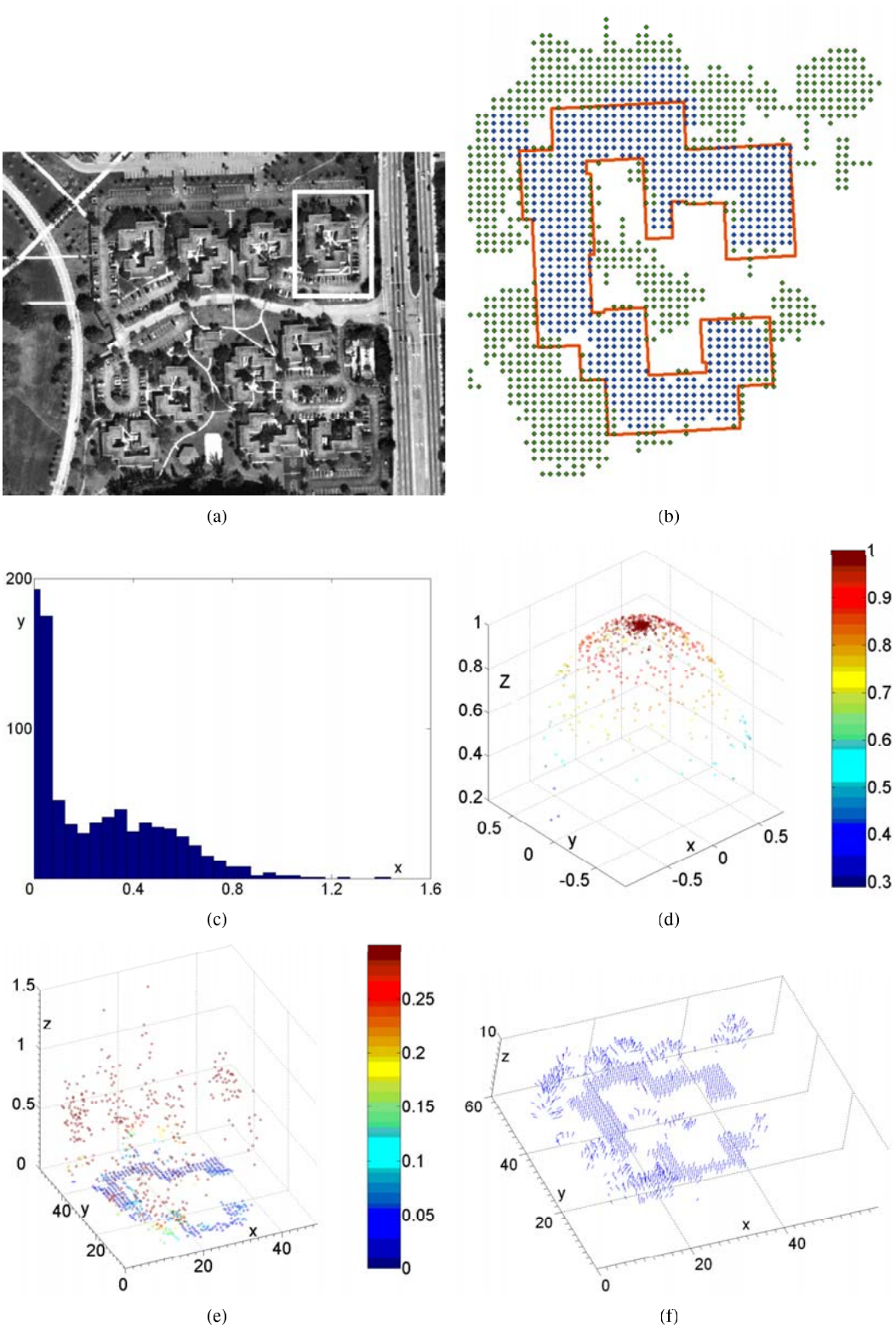
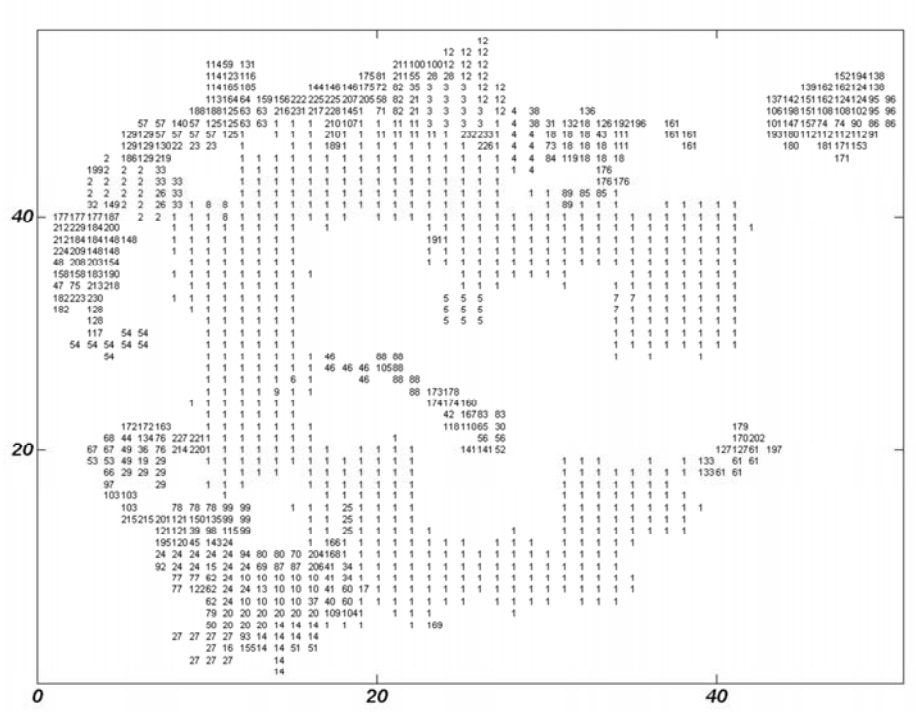
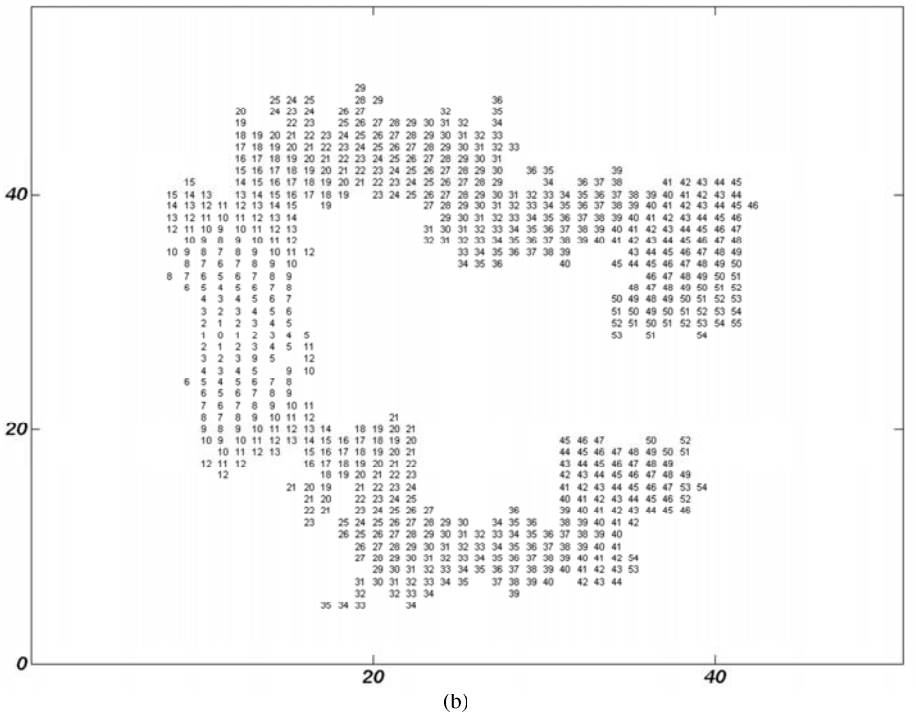


Figure 19: Variances and unit vectors of the inside points in an area of FIU main campus. a) aerial photography of the area b) non ground points filtered out after applying progressive morphology c) histogram of the variances. The x & y axes represent variance and the number of points corresponding to each variance d) distribution of the unit vector, which is represented by x, y & z coordinates e) variance z of each point located at (x, y) f) unit vector of each point located at (x, y, z). Each unit vector is represented by the blue arrow



(a)



(b)

Figure 20: Grid-based area growing result of the area showed in Figure 19(a) illustrates the grouped patches. Each number represents the identifier of the patch to which the corresponding point belongs (b) demonstrates the growing order of points belonging to patch 1 in Figure 19(a)

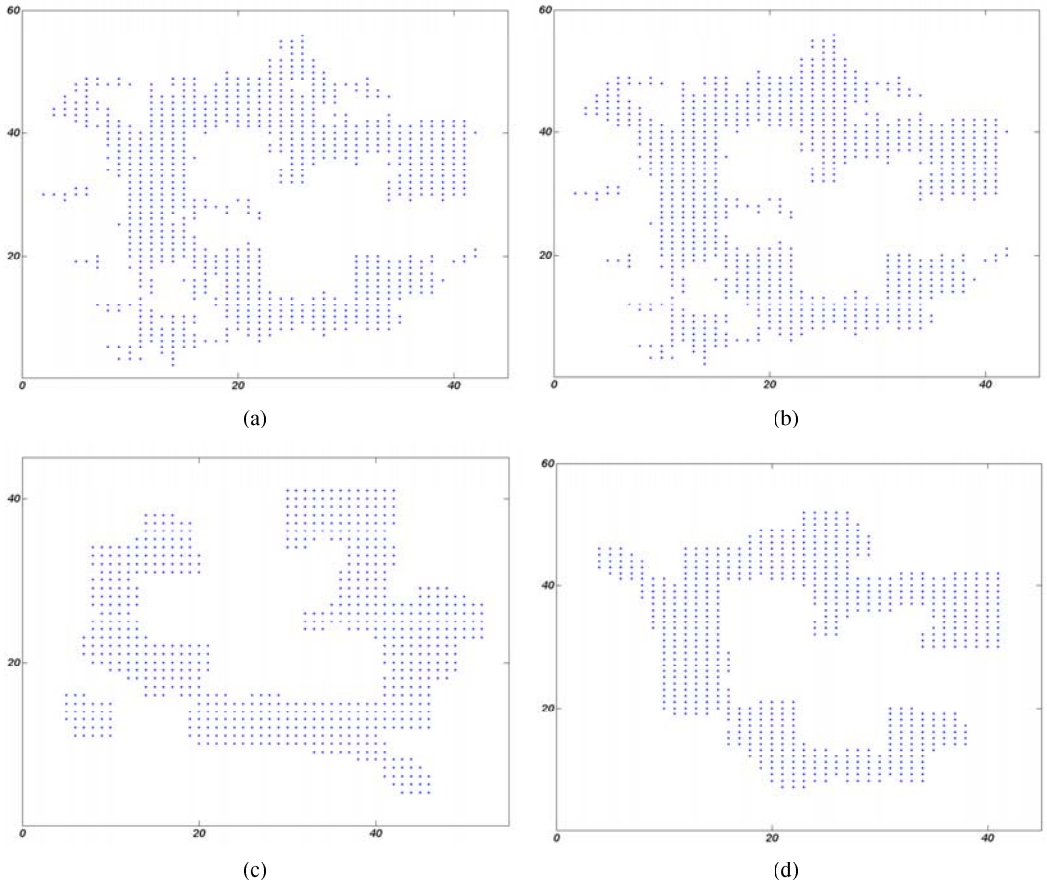


Figure 21: Intermediate results after applying post processing steps on the segmentation shown in Figure 20(a). (a) Small patches are removed (b) some small patches surrounded by the left patches are recovered (c) dangling points are removed (d) small connecting areas are removed

4.1.3 TIN Based Area Growing

Area growing can be implemented based on TIN (triangulation interpolated network). Raw LIDAR measurements are used for triangulation and no interpolation errors are involved. Therefore the segmentation result can keep more detailed 3D information about buildings. For the TIN based region growing algorithm, a TIN is created first using identified non-ground measurements and ground measurements surrounding the non-ground areas. Delaunay is the most famous algorithm in triangulation [5]. Triangles in a Delaunay triangulation are best-possible with respect to regularity, which is beneficial to interpolation and is what we desired. Therefore, Delaunay is selected for triangulation. The unit vector and variance are estimated for each triangle using six vertices of three neighboring triangles sharing one edge with the triangle. For example, as shown in Figure 22, triangle $P_1P_2P_3$ has three neighbors $P_1P_2P_4$, $P_2P_3P_6$ and $P_1P_3P_5$. Its unit vector and variance are determined by the fitting plane for vertexes of the four triangles which contain P_1, P_2, P_3, P_4, P_5 and P_6.

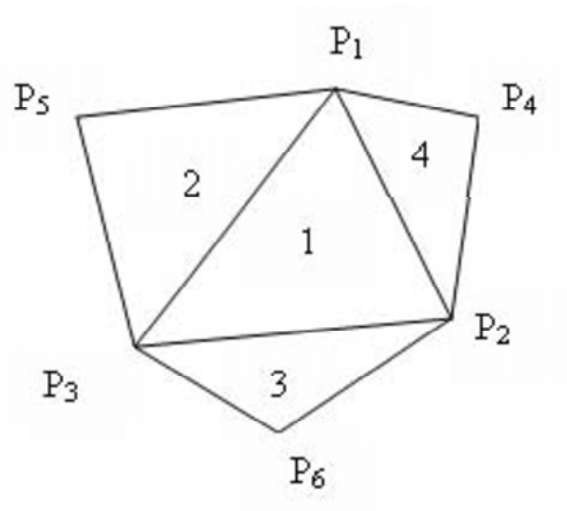


Figure 22: Neighbors of a triangle in the TIN

A triangle is considered an inside triangle if none of its vertices are ground measurements. Otherwise, the triangle is a boundary triangle. Three vertices of an inside triangle with minimum SSD are selected as seed points for region growing. The vertices of neighbor triangles sharing one edge with the seed triangle are selected as candidate points for region growing. The region growing process for these candidates is same as that based on the grid data structure. After the segmentation, the patches containing non-building measurements can be removed by the following two steps. First, the area of a patch is compared with the predefined threshold *Min_Surface* to eliminate small patch for vegetation. Second, the removed patches representing chimneys, water tanks and pipe lines in the first step are recovered if they are completely surrounded by the large patches left in the first step.

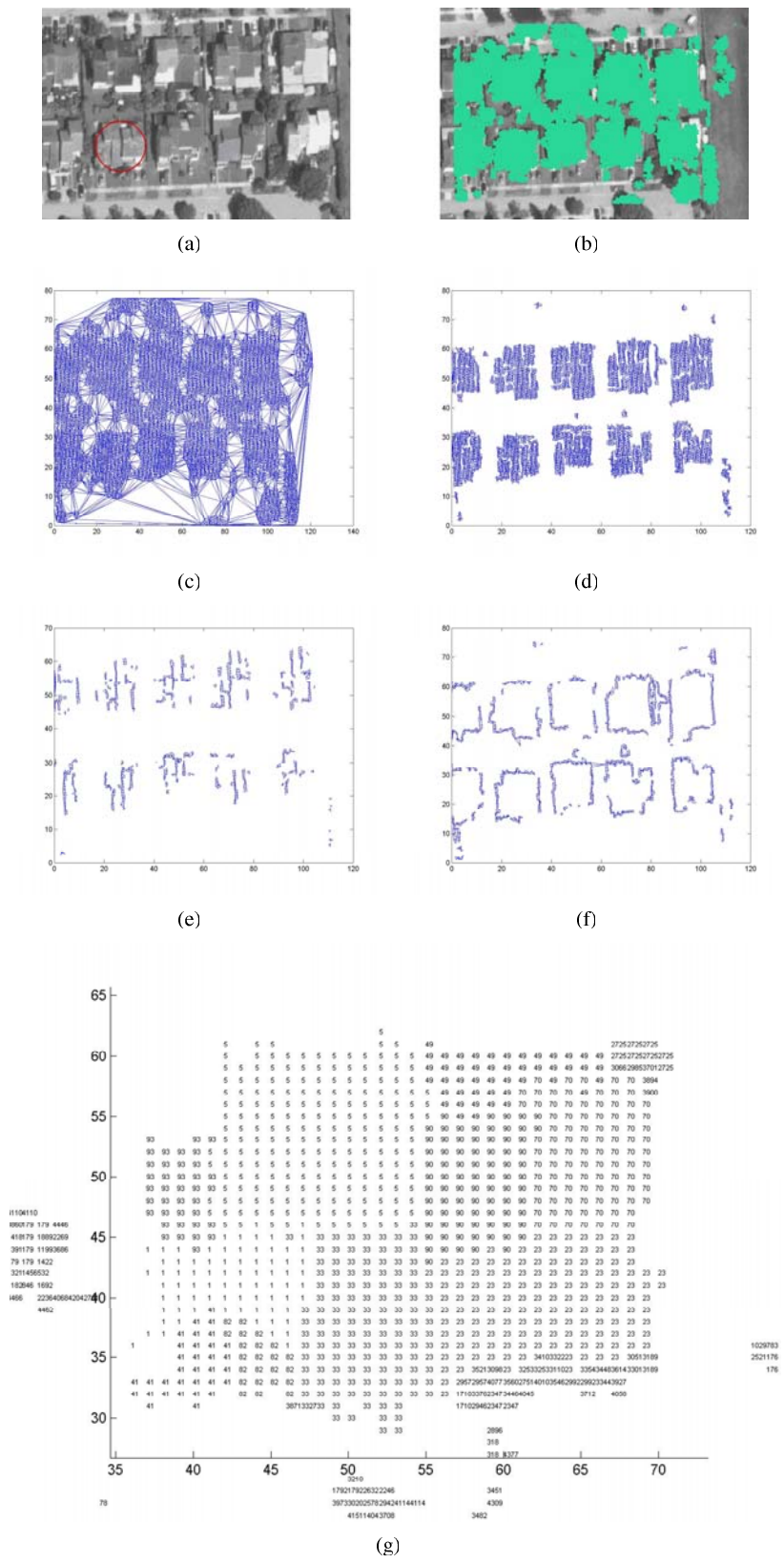


Figure 23: TIN based area growing result of one area around FIU main campus(a) aerial photograph of several residential houses around FIU main campus (b) LIDAR measurements for triangulation (c) triangulation result (d) triangles on the large patches (e) triangles connecting two or three large patches (f) boundary triangles (g) patch identifiers of the grid points on the building enclosed by the red circle shown in the aerial photography

To facilitate the succeeding operations, such as footprint extraction, each grid point should be judged whether it is a tree point or building point. For a boundary triangle, at least one of its vertices is ground measurement. If a grid point in a boundary triangle is nearest to one ground vertex, it will be identified as a ground point. Otherwise, it will be identified as a building point. There are two categories of inside triangles. The first category of inside triangle is located in a facet of the building roof and its three vertexes belong to the same patch. Grid points in such kind of triangle are classified into the same patch as that of the three vertexes of the triangle. The second category of triangle connects two or three building facets and its three vertexes belong to different patches. Grid points in this triangle are classified into the same patch as that of the vertex nearest to it.

Figure 23 demonstrates the segmentation result after applying TIN based area growing. Figure 23(a) depicts an aerial photography of an area around FIU campus. This area consists of several small residential houses surrounded by some trees. The roof surface of each building is very complicated and consists of several small facets with varying slopes. The segmentation result based on grid data structure is not satisfactory. LIDAR measurements for triangulation are displayed as green points in Figure 23(b) and the triangulation result is demonstrated on Figure 23(c). Figure 23(d) illustrates the triangles in the large patches. Triangles connecting two or three large patches are shown in Figure 23(e). They can be used to locate the boundaries separating neighboring facets of the roof. Figure 23(f) illustrates the boundary triangles. They can be used to outline the boundary of buildings. Each grid point in the non-ground areas is classified into a facet whose identifier is represented by a number in the corresponding location. Figure 23(g) displays the grid points of the building marked by a red circle in Figure 23(a).

The experiment result demonstrates that TIN based area growing is more sensitive to the height change. Some errors, such as building points misclassified from tree points by grid based area growing, can be lessened by TIN based area growing. However, noises on the building areas make more negative effects on TIN based area growing result than grid based area growing result. Generally speaking, there is no big difference between the building points identified from grid and TIN based data structures in terms of our experiments. In the following discussion, building points identified from grid based area growing method are utilized.

4.2 Boundary Extraction

The boundary point of each identified building area should be traced and stored in order in an array to facilitate the succeeding processes, such as footprint extracting. We proposed an algorithm to trace the boundary. First, a boundary point P_0 is randomly selected and stored in an array R for the boundary. One of its ground neighbors $P_1$is randomly selected as the predecessor of P_0. Search the neighbors of P_0 in the clockwise direction beginning from its predecessor P_1. The first found non-ground boundary point P_2in the neighbors is identified as the successor of P_0 and stored in the array R. Then P_0 is assigned as the predecessor

of $P_2$and the same searching operation is applied on $P_2$to find its successor. If its successor Q is in the array R but is not the head of the array, a loop starting from and ending at Q is found. Points in the loop are labeled, stored in another array and removed from the array R. If Q is the head of the array R, the searching will be finished and points in the array R are labeled. The left unlabeled boundary points are searched with the same operations till all the boundary points are processed. The searching can be based on 4-connectivity or 8-connectivity. They are shown in Figure 24(a) and 24(b) respectively. Figure 24(c) gives an example to trace the boundary. P_1 is the predecessor of $P_0$and located in the neighbor of P_0 indexed as 5. The neighbors of P_0indexed as 6, 7, and 0 are searched. The last one is the first found boundary point and taken as successor of P_0.

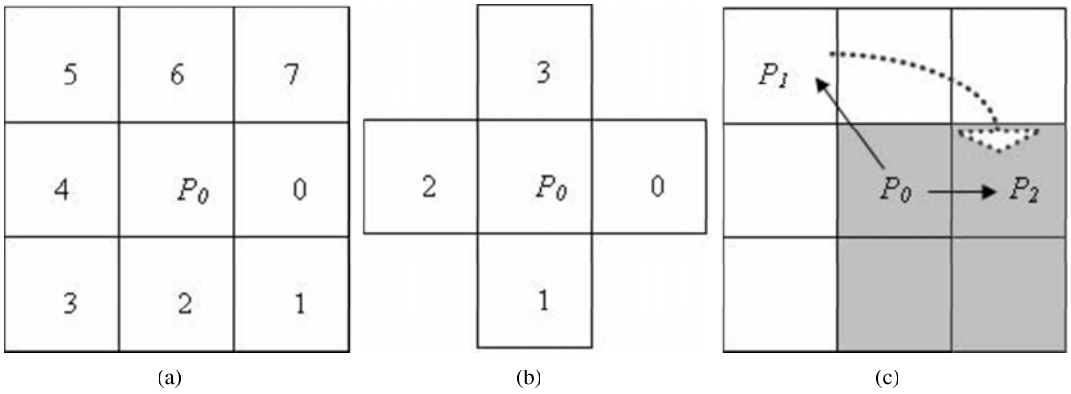


Figure 24: Boundary tracing (a) direction notation: 8-connectivity (b) 4-connectivity (c) the procedure to find the successor P_2 given a point $P_0$and its predecessor P_1

Figure 25 demonstrates the traced boundaries of the building shown in Figure 19, which contains three boundaries marked by blue, green and red contours respectively. The searching begins at the point circled by the blue rectangle. After all the points on the red contour have been processed, the successor of the current searching point, which is the point circled by the red rectangle, is found in the array for the boundary. Then the boundary corresponding to the red contour is removed from the array and stored. Similarly, the green and blue boundaries can be identified.

4.3 Derivation of Building Footprints

With the building measurements identified, a raw footprint can be obtained by connecting the boundary points (Figure 26a). However, the boundary of the raw footprint contains too many details and ”zig-zag” noise because of the irregularly spaced point measurements and the grid based interpolation. In this section, the following three steps describe the method to reduce the noise of raw footprints.

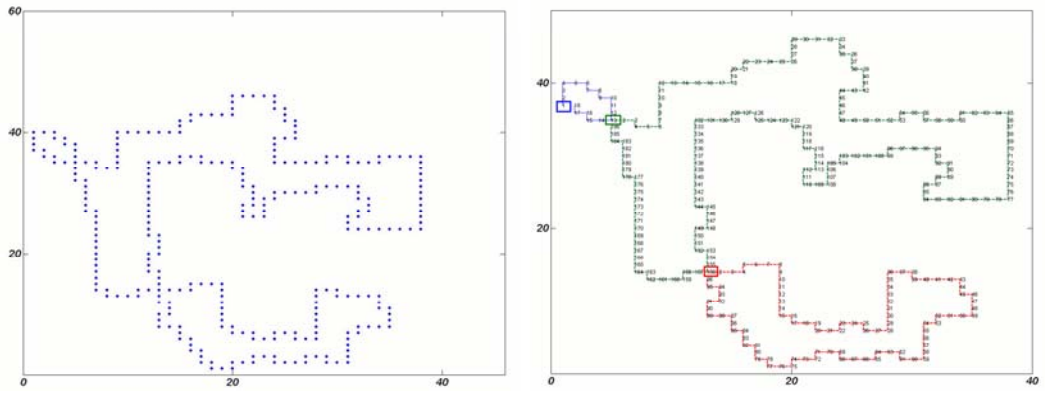


Figure 25: Boundary tracking result (a) the boundary points (b) three boundaries are detected. Each number represents the index of the corresponding point in the boundary array

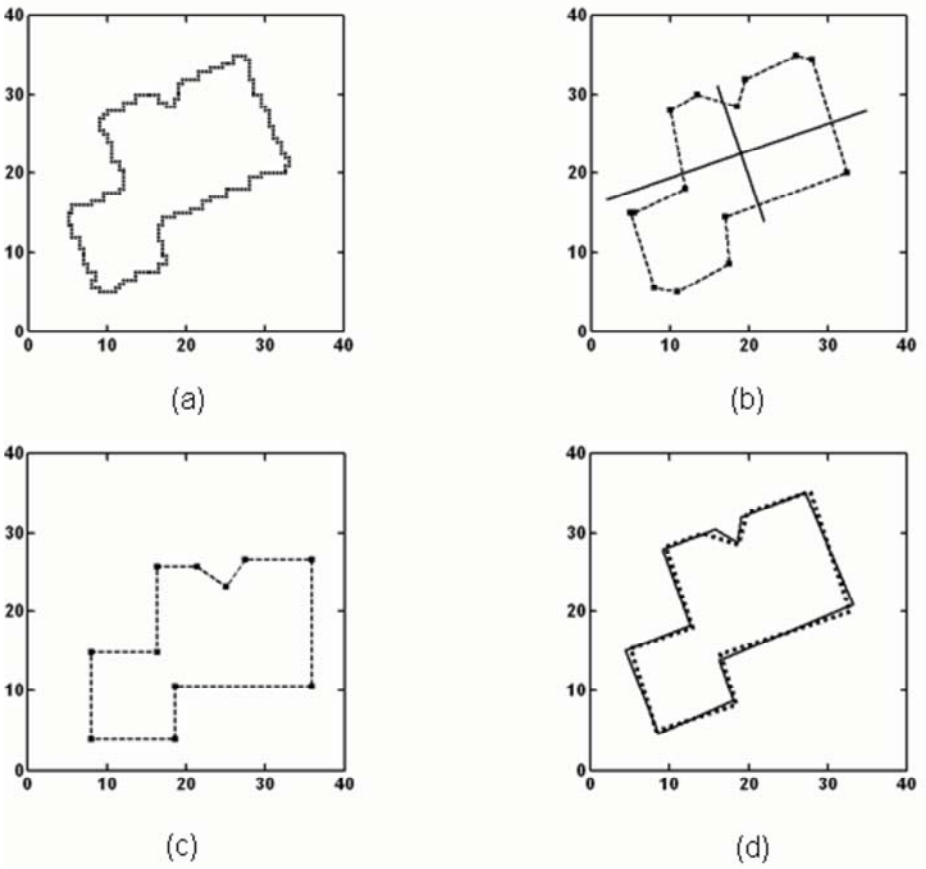


Figure 26: Example to illustrate the framework for extracting building footprints. The x and y coordinate units are in meters. (a) The raw footprint derived by connecting boundary points of identified building measurements through the region growing algorithm. The raw footprint is noisy due to the interpolation of the irregularly spaced LIDAR measurements. (b) The coarse footprint (dash line) that was derived by applying the Douglas-Peucker algorithm to the raw footprint and the estimated dominant directions (solid line). (c) The adjusted footprint that recovered the critical corner vertices removed by the Douglas-Peucker algorithm. The footprint was rotated clockwise according to the estimated dominant directions. (d) Comparison of the final footprint (dot line) with corresponding known footprint (solid line).

4.3.1 Extracting the Coarse Footprint

Several algorithms have been employed to reduce the vertices of a noisy raw building footprint and derive a coarse footprint. Latecki and Lakamper [38] proposed a method based on a conspicuous value to remove boundary noise. The drawback of this method is that it is difficult to automatically define the termination condition. Weidner and Forstner [63] eliminated noisy vertices of a footprint polygon by comparing the heights of triangles formed by three consecutive vertices with a predefined threshold. The Douglas-Peucker algorithm [14] is another alternative to simplify lines with noise. The Douglas-Peucker algorithm generalizes lines by forming a line connecting start and end points first, and then recursively selecting a left point with a largest distance to the line until a predefined distance threshold *T_Douglas* is reached. The Douglas-Peucker algorithm was implemented in our framework because of its simplicity.

4.3.2 Estimating the Dominant Directions of a Building

The Douglas-Peucker algorithm occasionally removes critical corner vertices of a raw footprint due to the "zig-zag" pattern of boundary lines, thus distorting the orientation of segments and producing acute or obtuse angles between segments (Figure 26b). In order to recover the removed critical vertices and distorted segments, the building footprints were divided into two categories. For the first category, there are two dominant directions for a building footprint polygon, and most segments are parallel or perpendicular to each other. For the second category, a considerable number of segments are oblique to the two dominant directions, or more than two dominant directions exist. The critical step in identifying these two categories of building footprints is to estimate the dominant directions.

Maas and Vosselman [41] derived dominant directions using invariant parameters, based on an assumption that the roof type of a building is already known. However, building roof types are not available in many cases before building footprints are identified [42]. Also, it is often difficult to classify the roofs of complex buildings into any given type. A 2D Hough Transform can be used to estimate the dominant directions of a building footprint. The major limitation of the Hough Transformation is that it is difficult to determine the optimum cell size in parameter space.

We have proposed a new method to estimate the dominant directions of a building footprint based on weighted line segment lengths. Let x' and y' represent possible dominant directions in a 2D coordinate system x and y (Figure 27). The dominant directions x' and y' are related to the coordinate system x and y through a counterclockwise rotation by an angle φ ($0^0 \leq \varphi < 90^0$)). Therefore, the key step to estimate the dominant directions is to find the rotation angle φ. Assuming that the counterclockwise intersection angle between a line segment and x axis is $\theta_i (0^0 \leq \theta_i < 180^0)$, we define

$$SL = \sum_{i=1}^{N} g(L_i) f(\beta_i(\theta_i, \varphi)) \qquad (22)$$

where N is the total number of vertices of a building footprint, L_i is the segment length, β_i ($0^0 \leq \beta_i < 45^0$) is the minimum intersection angle between a segment and the nearest axis in the coordinate system x' and y', and is determined by θ_i and φ. $g()$ is the weight function based on L_i, and $f()$ is the weight function based on and θ_i and φ. The dominant building directions can be estimated through finding an optimum φ so that SL will reach a minimum. A linear function is employed to represent $g()$

$$g(L_i) = L_i / \sum_{i=1}^{N} L_i \tag{23}$$

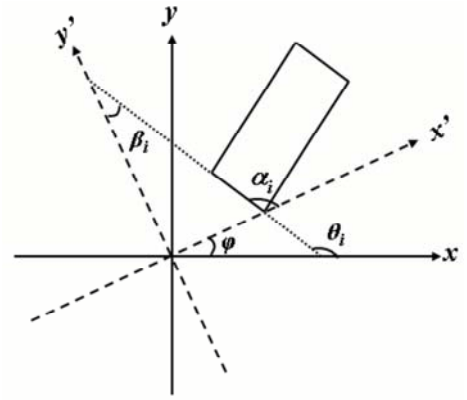


Figure 27: Relationship between angles β_i, θ_i, α_i and φ. The coordinate system x' and y' is a counterclockwise rotation of the coordinate system x and y by an angle φ. θ_i is the counterclockwise intersection angle between a segment of a building footprint (solid square) and the axis x. α_i is the counterclockwise intersection angle between a segment and the axis x'. β_i is the minimum intersection angle between a segment and the axes x' and y'

Finding the optimum φ depends heavily on $f()$ which can be in many forms such as linear and exponential. We would like to construct $f()$ such that the segments close to the dominant direction will have a small contribution to SL. Here a linear function is used to represent $f()$

$$f(\beta_i(\theta_i, \varphi)) = \beta_i / 45 \tag{24}$$

Obviously, the closer the segment is to the dominant direction, the smaller the $f()$. β_i is determined by

$$\beta_i = \begin{cases} min(\alpha_i, 90 - \alpha_i) & : \quad \alpha_i \leq 90 \\ min(180 - \alpha_i, \alpha_i - 90) & : \quad \alpha_i > 90 \end{cases} \tag{25}$$

where $\alpha_i (0^0 \leq \alpha_i < 180^0)$ is the counterclockwise intersection angle between a line segment and the axis x' and has the following relationship with θ_i and φ

$$\alpha_i = \begin{cases} \theta_i - \psi & : \quad \theta_i \geq \varphi \\ 180 + \theta_i - \varphi & : \quad \theta_i < \varphi \end{cases} \tag{26}$$

Numerically, the optimum φ is found by comparing SL values for angles between 0^0 and 90^0. After φ is derived, the building footprint is rotated so that the x and y axes are aligned with the dominant directions of the buildings.

Although the algorithm is simple, this method to estimate the dominant direction of a building is very robust and has the following property:

Estimated dominant directions are the same as the directions of parallel and perpendicular segments as long as the total length of oblique lines is less than the total length of parallel and perpendicular segments of a footprint.

Theorem 1 *Let directions for parallel and perpendicular segments be* x *and* y*, and possible dominant directions are* x' *and* y' *that are derived by rotating* x *and* y *axes with an angle φ (Figure 28). Since θ_i is 180^0 (or 0^0) and 90^0 for parallel and perpendicular segments, respectively, β_i for parallel and perpendicular segments is the same (β_D) and equal to the rotation angle φ or the complementary angle of φ. Therefore, the contribution (SL_D) of rotated parallel and perpendicular segments to SL can be represented by*

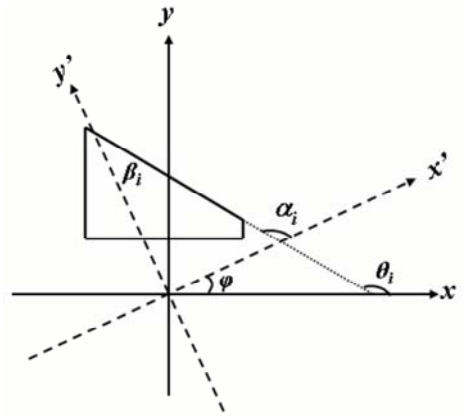


Figure 28: A footprint with several parallel and perpendicular segments and an oblique line. The parallel and perpendicular segments align with dominate directions that are x and y axes. Angles β_i, θ_i, α_i and φ have the same definitions as those in Figure 27.

$$\begin{aligned} SL_D &= \sum_{i=1}^{N} g(L_i)\beta_i/45 \\ &= \beta_D L_D/45L_T \\ &= \begin{cases} \varphi L_D/45L_T & : \quad \varphi \leq 45 \\ (90-\varphi)L_D/45L_T & : \quad \varphi > 45 \end{cases} \end{aligned} \tag{27}$$

where L_D is the total length of parallel and perpendicular segments, and L_T is the total length of segments of a footprint. SL_D is a period function with minimum values at 0 and 90 and maximum value at 45 (Figure 29). The amplitude and period of SL_D are L_D/L_T and 90, respectively.

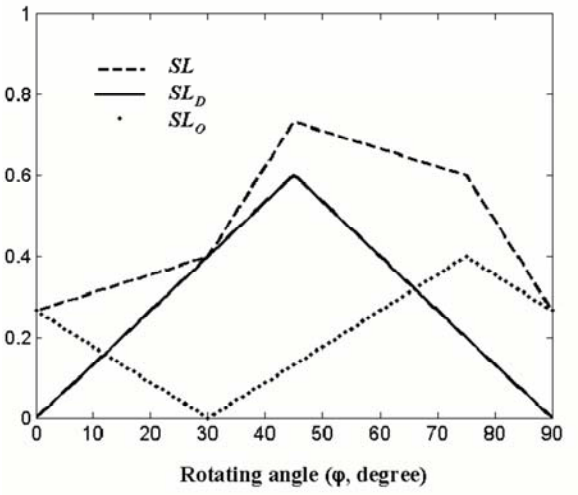


Figure 29: SL_D, SL_O, and SL for a footprint with several parallel and perpendicular segments and an oblique line. The length of parallel and perpendicular segments is 60% of the total length of segments, and the length of the oblique line is 40% of the total length of segments. The θ_{Oi} for the oblique line is 120^0. The directions of parallel and perpendicular segments are the dominant directions of the footprint.

The contribution (SL_O) of rotated oblique lines to SL can be described by

$$SL_O = \sum_{i=1}^{M} SL_{Oi} = \sum_{i=1}^{M} L_{Oi}\beta_{Oi}(\varphi, \theta_{Oi})/45L_T \tag{28}$$

where M *is the number of oblique segments, SL_{Oi} is the contribution of* ith *oblique segment to SL, L_{Oi} is the length of* ith *oblique segment, and β_{Oi} is the minimum intersection angle between the oblique segment and the nearest axis in the coordinate system* x' *and* y'. *Similar to SL_D, SL_{Oi} has period of 90 and amplitude of L_{Oi}/L_T. The initial phase of SL_{Oi} is determined by the counterclockwise intersection angle θ_{Oi} between the oblique segment and* x *axis. Figure 29 shows the change of SL_{Oi} as a function of φ when θ_{Oi} is 120^0.*

In order to facilitate the analysis, we divided SL into two sections: $\varphi \leq 45^0$ and $\varphi > 45^0$ In section $\varphi \leq 45^0$, SL becomes

$$SL = \varphi L_D/45L_T - \sum_{i=1}^{M} L_{Oi}\beta_{Oi}(\varphi, \theta_{Oi})/45L_T \tag{29}$$

The derivative of SL with respect to φ is

$$\frac{\partial SL}{\partial \varphi} = \frac{1}{45L_T}(L_D + \sum_{i=1}^{M} \pm L_{Oi}) \geq \frac{1}{45L_T}(L_D - L_O) \tag{30}$$

where L_O is the total length of oblique segments. The sign before L_{Oi} is determined by θ_{Oi} and φ. Since $L_D > L_O$, no matter how SL_O changes, SL, like SL_D, increases monotonically in section $\varphi \leq 45^0$. Similarly, we can demonstrate that SL decreases monotonically in section $\varphi > 45^0$. This indicates that SL reaches minimum at $\varphi = 0$ or $\varphi = 90$. There are no better dominant directions for the footprint other than the directions of parallel and perpendicular segments as long as the total length of parallel and perpendicular segments is larger than that of oblique lines.

4.3.3 Adjusting the Footprint

For the first category of building footprints which have two distinct dominant directions, a unique optimum φ is derived. The SL value for a rotated footprint should be small. However, for the second category of building footprints, multiple optimum φ could be derived. In such a case, the first optimum φ is used to rotate the footprint. The SL value for the second category of footprints should be large. Therefore, the SL value for each rotated building footprint was employed to classify the footprint. If the SL value is less than a threshold T_SL (e.g., 0.3), the footprint is classified as the first category which has two dominant directions. Otherwise, the footprint belongs to the second category. For the first category, we propose an algorithm to recover the critical vertices based on two assumptions:

- Each building has two dominant directions and they are perpendicular to each other
- Most (e.g., 80%-90%) of the boundary segments are parallel to one of the two dominant directions

Segments of the rotated footprint are adjusted using four types of operations. The ***split*** operation is used to adjust certain horizontally and vertically oblique segments. The adjustable horizontally oblique segment is the line whose projection on the x axis is equal to or greater than the y axis, and whose projection on the y axis is less than a threshold $T\_Projection$. The adjustable vertically oblique segment is the line whose projection on the x axis is less than the y axis, and whose projection on the x axis is less than $T\_Projection$. Figure 30a shows how an adjustable horizontally oblique segment P_1P_2 is straightened by adding two more points P'_3 and P'_4 using ***split***. Line P_1P_2 is replaced by three lines $P_1P'_3$, $P'_3P'_4$, P'_4P_2, among which P'_3P_4' is horizontal. The x coordinates of vertices P'_3 and P'_4 are the same as those of P_1 and P_2. The y coordinates are the average of P_1 and P_2, respectively. A similar operation can be applied to an adjustable vertically oblique segment by exchanging the x and y coordinates.

The ***intersect*** operation is employed to recover the missing corner as shown in Figure 30b. This operation compares the ratio R of the area of the triangle P_2PP_3 to that of triangle P_1PP_4. If the R is less than the threshold T_Ratio, the four points P_1, P_2, P_3, and P_4 are replaced with points P_1, P and P_4.

The ***merge*** operation adjusts four consecutive points P_1, P_2, P_3, and P_4 when P_1P_2 is parallel to P_3P_4, but not collinear. Without loss of generality, P_1P_2 is assumed to be horizontal. The operation can be applied

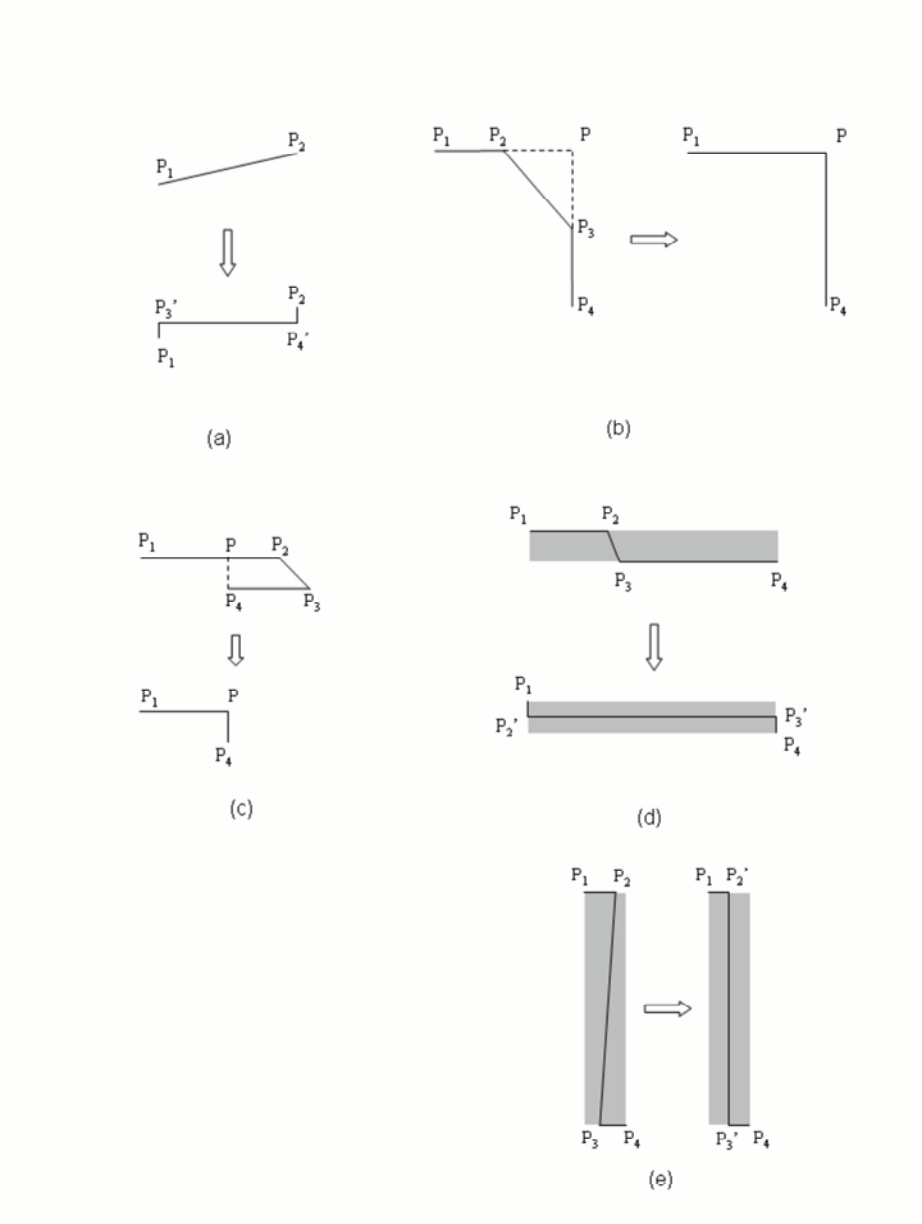


Figure 30: Operations proposed to adjust segments of a footprint (a) ***split***, (b) ***intersect***, (c) first type of ***merge***, (d) second type of ***merge***, and (e) third type of ***merge***.

to the case when P_1P_2 is vertical by exchanging the x and y coordinates. There are three types of merge operations. The first type is for the case that P_1P_2 is opposite to P_3P_4 (Figure 30c). If the distance between P_1P_2 and P_3P_4 is less than the threshold $T\_Deviation$, the merge operation replaces the four original points with P_1, P and P_4. This operation is used to remove the long, thin portion protruding from a footprint. For the other two types of merge operations, P_1P_2 has the same direction as that of P_3P_4. The difference is that the projection of line P_1P_4 on the x axis is longer than that on the y axis for the second type (Figure 30d), while the projection on the x axis is shorter for the third type (Figure 30e).

A horizontal segment $P_2'P_3'$ is constructed for the second type of merge so that P_1P_2' and $P_3'P_4$ are vertical. The y coordinates of P_2' and P_3' are between those of P_1 and P_4 and are inversely proportional to lengths $L_{P_1P_2}$ and $L_{P_3P_4}$

$$y = y_3 + (y_2 - y_3)\frac{L_{P_1P_2}}{L_{P_1P_2} + L_{P_3P_4}} \tag{31}$$

where y_2 and y_3 are coordinates of P_2 and P_3 on the y axis. If the difference $|y_2 - y_3|$ is less than the threshold $T\_Deviation$, the original four consecutive points P_1, P_2, P_3, and P_4 are converted into P_1, P_2', P_3' and P_4 by the ***merge*** operation. The third type of merge operation is illustrated in Figure 30e. A vertical $P_2'P_3'$ is constructed by using the average of x coordinates of P_2 and P_3 and y coordinate of P_2 or P_3. If the absolute difference between y coordinates of P_2 and P_3 is less than threshold $T\_Deviation$, the points P_1, P_2, P_3, and P_4 are replaced by P_1, P_2', P_3' and P_4.

The ***remove*** operation is designed to remove the redundant vertices of a line. Extra vertices may exist on a segment of the footprint after operations ***split***, ***intersect***, and ***merge*** are performed. These extra collinear vertices are eliminated using the remove operation immediately after performing one of three other operations.

The segments of the first category of building footprints are adjusted iteratively using the four operations starting from a small value of $T\_Projection$. $T\_Projection$ is incremented gradually until the percentage of the lengths of horizontal and vertical segments of a footprint is over the predefined threshold $T\_Footprint$. $T\_Footprint$ is usually set to be between 80% and 90% of the perimeter of the adjusted footprint. Figure 26c displays a building footprint refined from the coarse footprint (Figure 26b). Corner vertices lost due to the line simplification by the Douglas-Peucker algorithm were recovered successfully by our footprint adjustment algorithm.

The same operations and procedure are employed to adjust the second category of building footprints. However, it is not appropriate to use the threshold $T\_Footprint$ as a termination condition because a considerable number of segments are not parallel to the estimated dominant directions. A footprint could be distorted severely if a large $T\_Footprint$ is used. Therefore, an alternative termination condition for footprint adjustment is employed. The iteration is stopped when $T\_Projection$ is greater than a threshold $T\_Projection\_Final$ to avoid adjusting long oblique segments.

Table 4: Parameters for extracting building footprints

Cell size (c_s) for progressive morphological filter	0.5 m
Height difference to aggregate a point (Δh_T)	0.2 m
Minimal surface on the roof ($Min\_Surface$)	5 m^2 (20 c_s^2)
Minimal building area ($Min\_Building$)	60 m^2 (240 c_s^2)
Douglas distance ($T\_Douglas$)	1.5 m (3 c_s)
Threshold for footprint classification (T_SL)	0.3
Threshold for split adjustment ($T\_Projection\_Final$)	2 m (4 c_s)
Threshold for deviation ($T\_Deviation$)	2 m (4 c_s)
Threshold for triangle area ratio (T_Ratio)	0.1
Threshold for footprint evaluation ($T\_Footprint$)	0.85

4.4 Data Processing

The study area is located at and around the campus of Florida International University (FIU), covering 6 km^2 of low relief topography. Surveyed features include residential houses, complex buildings, individual trees, forest stands, parking lots, open ground, ponds, roads, and canals. The data were collected in April 2000 and August 2003 with Optech ALTM 1210 and 1233 systems operated by FIU, respectively. The Optech system recorded the coordinates (x, y, z) and intensity of the point measurements corresponding to first and last laser returns. The 2000 data set consists of three overlapping, 400 m wide swaths of 15 cm diameter footprints spaced approximately 2 m apart. The 2003 data set consists of five overlapping, 340 m wide swaths of 13 cm diameter footprints spaced approximately 1 m apart.

Building footprints were extracted from two test LIDAR data sets for the FIU campus and adjacent areas to examine their effectiveness. The thresholds used in our experiments for the FIU campus dataset are listed in Table 4. These thresholds were derived empirically by visually comparing the results with gridded raw LIDAR measurements. This is feasible because it took 9, 2, and 0.7 minutes for a PC with 2.8 GHz processor and 2 GB RAM to perform morphological filtering, building measurement identification, and footprint derivation for the FIU campus dataset. A two dimensional array with about 7.2 million elements was employed to represent raw and interpolated points covering an area of 1.8 km^2. Sensitivity analysis showed that small changes in these thresholds have little impact on the final results.

Aerial photographs, a building footprint map from the FIU Planning and Facility Management Department, and field investigation were used to qualitatively and quantitatively evaluate the derived footprints. The aerial photographs were collected in 1999 at a resolution of 0.3 m. The Planning and Management Department footprint map was made mainly through ground surveying when buildings were constructed and included 62 buildings. All surveyed buildings can be found on the aerial photographs and they did not change over the time. Therefore, these buildings can be used to quantify errors introduced by our framework.

4.5 Results

Both qualitative and quantitative methods were employed to measure the errors committed in extracting building footprints in this study. A qualitative method checks the quality of estimated dominant directions and

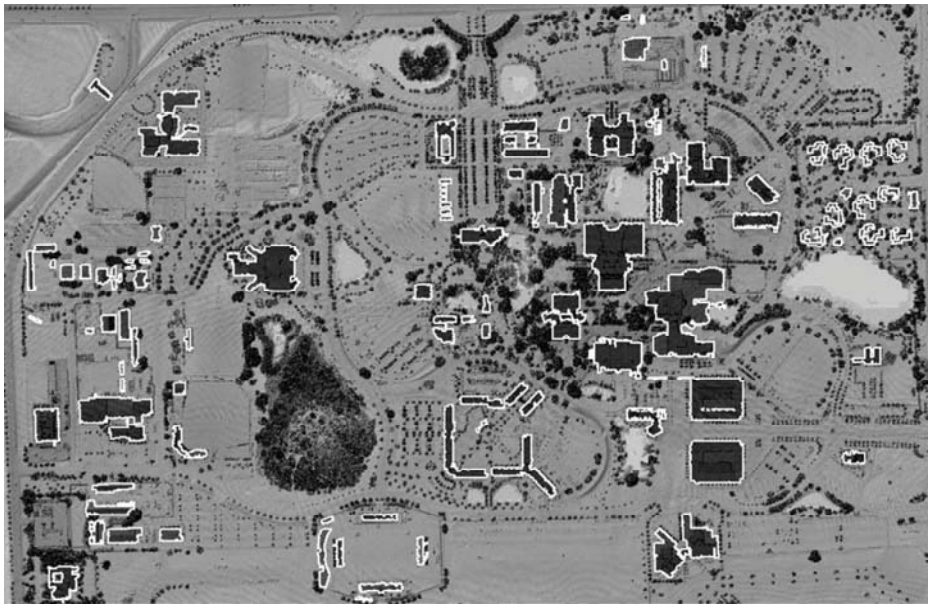
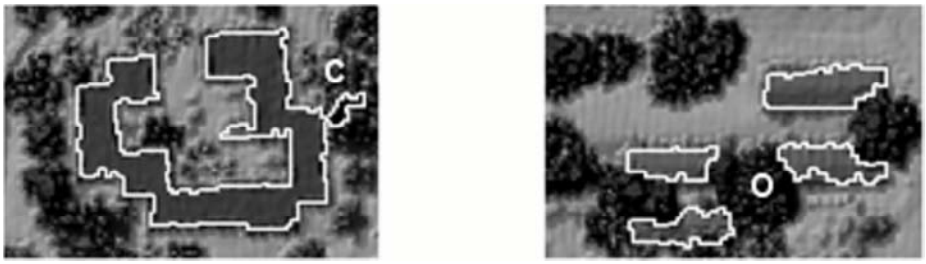


Figure 31: The raw footprints for buildings at FIU Campus. Raw footprints were derived by connecting boundary points of identified building measurements using the region growing algorithm. The background image was derived by interpolating raw LIDAR point measurements.

derived footprints by visually comparing the extracted footprints with those in maps and aerial photographs. The quantitative method examines the accuracy by extending the count-based metric method proposed by Shufelt and Mckeown [53] and area-based metric method suggested by Ruther et al. [23]. The count-based metric method quantifies commission and omission errors in the number of buildings identified. The area-based metric method measures the changes of two footprints derived from different stages of the proposed framework. Raw footprints derived by connecting boundary points from identified building measurements are placed over known building footprints to examine the accuracy of the region growing segmentation algorithm. The omission error is measured by the percentage of area not in a raw footprint but in the known building footprint. The commission error is measured by the percentage of area in the raw footprint but not in the known building data set. Raw and adjusted footprints are compared to evaluate the changes induced by the vertex recovery algorithm. The commission and omission errors from comparison of final building footprints with corresponding known building footprints are employed to measure the performance of the entire framework.

To demonstrate the effectiveness of the proposed framework for building footprint extraction, the results from last return measurements of the 2003 data set are presented in detail in this section. Figure 31 displays raw footprints for identified building areas at the FIU campus. Count-based error analysis shows that all buildings were identified and no tree areas were mistakenly included. Area-based error analysis compared raw footprints with known building footprints. The results show that omission and commission errors are 10% and 2%, respectively, indicating that the segmentation algorithm worked well to identify most building

measurements. The commission error occurs because the elevation change of some trees adjacent to buildings is similar to those of buildings (Figure 32a). These tree measurements were mistakenly included by the region growing segmentation. The omission errors are 6 percent larger than commission errors. The major reason for the relatively large omission error is that there are 40 buildings whose areas are larger than 600 m^2 in the known building footprint data set. Most roofs of the large buildings are flat with narrow high surfaces at edges. These edges are often removed by plan-fitting technique because of large elevation deviations. Another factor is the trees covering roofs that prevent the portion of buildings underneath the trees from being identified (Figure 32b).



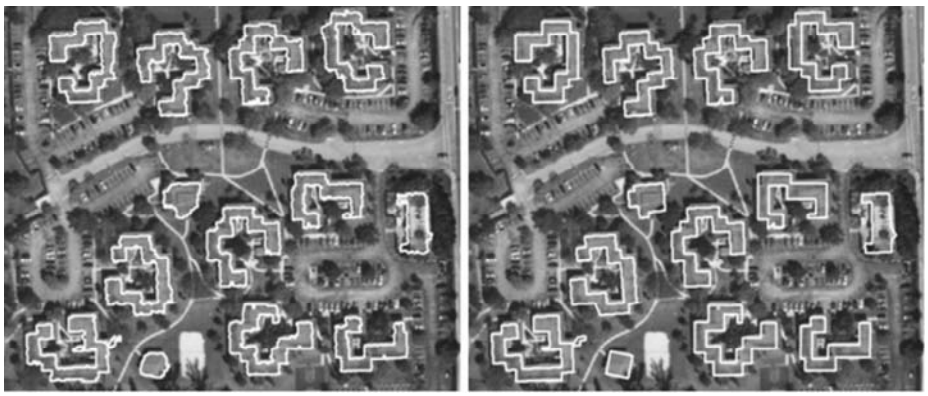
(a) (b)

Figure 32: Examples of errors caused by the region growing building segmentation algorithm. (a) Commission error (**C**): the flat tree top next to the building is misidentified as a part of the building. (b) Omission error (**O**): the corner portions of buildings were missed because the roofs are partially covered by trees.

The effect of the footprint adjustment algorithm is well illustrated in Figure 33. Figure 33a shows that the raw building footprints have noise on their boundaries. Most of the noise was removed in the adjusted footprints as shown in Figure 33b and the smoothness of the footprint polygons was greatly improved. To test the deviation caused by the footprint adjustment algorithm, raw footprints were compared with adjusted footprints. The results show that the deviation is small. About 3% of the raw footprints are not contained in adjusted footprints, and 3% of the footprints are mistakenly added to the adjusted footprints. By visually comparing the adjusted footprints with raw footprints, we found that in most cases the adjusted footprints preserve the original geometric shape.

The effectiveness of the algorithm to estimate dominant directions was evaluated by visually examining all final footprints. The algorithm worked well for all buildings at the FIU campus. For example, Figure 61a shows a complex building which consists of five major parts. The dominant directions were estimated to be nearly horizontal in terms of three major rectangles. A circle portion (C in Figure 61a) was approximated by several line segments (Figure 61b). A small oblique rectangle (R in Figure 61a) whose direction is different from the dominant directions was also adjusted appropriately.

Figure 35 shows a 3D building map for FIU campus based on final building footprints and heights. The heights of the buildings were derived by averaging the elevation differences between building measurements



(a) (b)

Figure 33: Comparison of raw (a) and adjusted (b) building footprints. The small "zig-zag" noise in the raw footprints was removed in the adjusted footprints, making the adjusted footprints look more realistic.

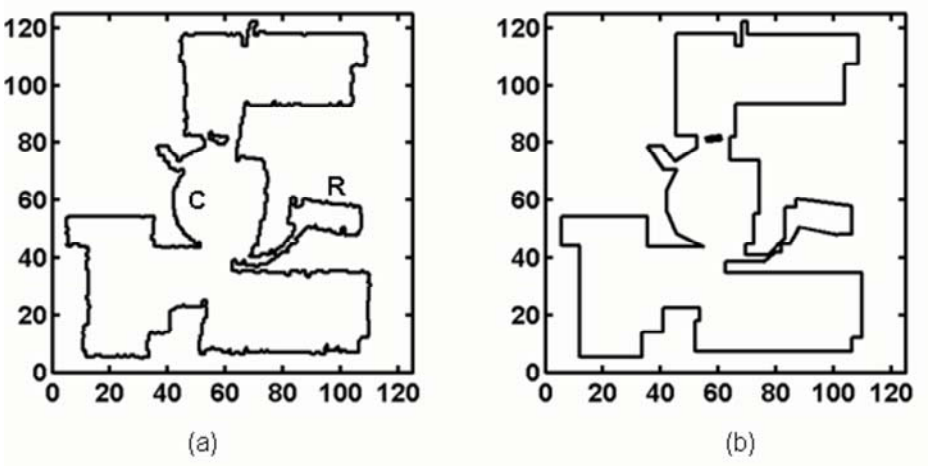


Figure 34: The raw (a) and final (b) footprint for a complex building. The *x* and *y* coordinate units are meters. Two dominant directions of the footprint are nearly horizontal and vertical. Note that the arc of a circle (C in a) of the footprint is approximated by a polyline. A small rectangle (R in a) that is not aligned with the dominant directions is well preserved.

and the digital terrain model (DTM) interpolated from ground measurements. These building models have been used to construct a 3D synthetic visual environment to animate hurricane-induced fresh water flooding at the FIU campus. Comparison of 62 final footprints with footprints from the map provided by the FIU Planning and Facility Management Department shows that 10% of the building footprints were mistakenly removed, and 2% of the footprints were incorrectly included into the final output by our framework. Both the omission and commission errors for the final footprints are almost the same as those caused by the region growing segmentation algorithm. This further proves that the deviations caused by the footprint adjustment algorithm have little effect on the final result. Therefore, the errors in final footprints mainly come from the errors before the footprint adjustment is performed.

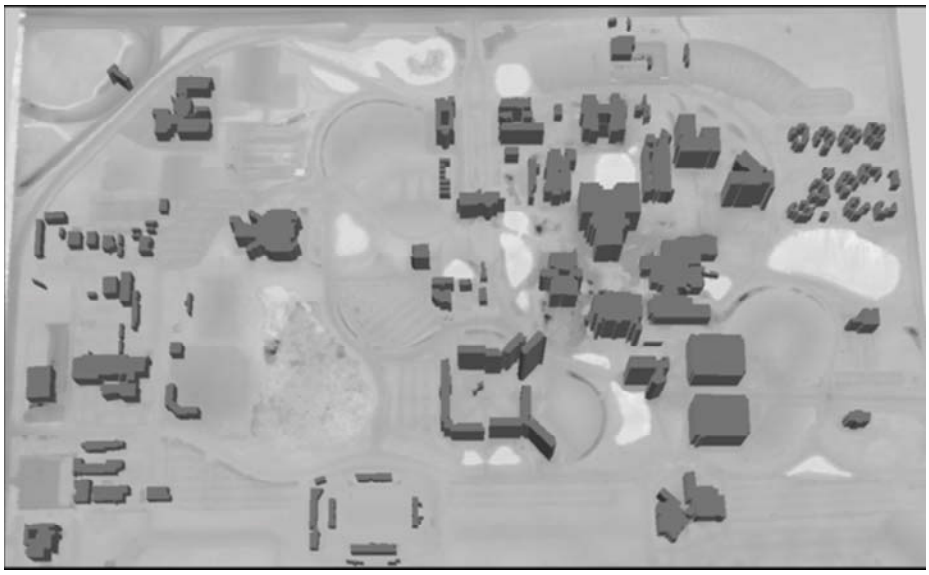


Figure 35: 3D building models for FIU campus. Each building model was created using the final footprint and average building height derived from LIDAR measurements. The DTM for building bases was derived by interpolating ground measurements identified by the progressive morphological filter.

Building extraction results for a residential area are displayed in Figure 36. The known building footprint data for the residential area were derived by digitizing buildings from aerial photographs. The relief displacement of the residential houses in the orthorectified photograph was small due to their low heights. The digitized footprints were overlaid over a grid interpolated from raw LIDAR measurements to ensure quality. The data set included 211 building footprints, and the parameters used by the framework are the same as those listed in TABLE 4. All buildings were identified from LIDAR data and there were no commission and omission errors from the count-based accuracy analysis. Both area-based omission and commission errors for final footprints were about 6%, indicating that the framework worked well in residential areas.

Several factors influences the accuracy of building measurement identification. One of them is the point measurement used in computation. Most airborne LIDAR systems are capable of deriving first and last return

Figure 36: Building footprints extracted from LIDAR data for a residential area.

measurements for an emitted laser pulse. Both first and last return measurements can be used to identify the building footprints. However, there are different advantages and disadvantages in using them. The first return measurements suffer fewer errors from multipath reflections which can be caused by many factors. For example, when a laser pulse hits glass walls or windows, it can enter the room and bounce several times before it finally reaches the sensor. The multipath reflection of a laser pulse can lead to incorrect low elevation measurements for the roof of a building. These low elevation points are often removed as ground measurements by the progressive morphological filter, resulting in small holes in the building footprint. There are many more multipath errors in the last return measurements than in the first return measurements in our data set. However, the last return measurements have more of a chance to penetrate into the vegetation and reach the ground. This helps the filter to separate the ground and non-ground measurements. Also, the last return measurements display more spatial change than the first return in tree areas, which helps separate the building measurements from tree measurements. The current building identification methods have been applied to both first and last return data in the study area. It was found that the last return measurements have a better overall performance. Improvement of the current algorithms by combining the first and last return measurements need to be further investigated.

The performance of the region growing algorithm to segment building measurements in the proposed framework is critical to the footprint extraction. The processes of region growing segmentation rely on the selection of seed points. In our algorithm, the seed points are selected in terms of minimum SSDs based on a plane-fitting technique. To examine the robustness of the algorithm, other methods for selecting seed

points were tested for the FIU campus data set. One alternative method is to start seed point selection at the left upper corner of a non-ground area, and then select seed points based on an increased order of x coordinates and a decreased order of y coordinates. Comparison of the seed point selection starting at the left upper corner with that based on minimum SSDs shows that slight differences sometimes occur in areas of individual patches (roof surfaces) within a building footprint, especially for those points at the boundaries between two roof surfaces. However, there is little difference between the identified measurements for the whole building when individual patches are merged for the building.

Three other cases with different combinations of increased or decreased x and y coordinates have also been tested. The results show that the maximum difference between identified building measurements using these combinations and those from minimum SSDs is less than 1%. A random selection of seed points was also performed for individual building areas, and the results also show that the selection of seed points has little impact on the building measurement identification. This indicates that the region growing algorithm based on plane-fitting is robust for segmenting building measurements.

LIDAR measurement density also has great effect on segmentation results. The LIDAR points for 2000 and 2003 data sets are spaced approximately 2 m and 1 m, respectively. To test the effect of point density, building footprints were extracted from both data sets. Comparison of the footprints for the same buildings shows that the 2003 data set with a higher point density produced a much better result for the FIU campus because small building surfaces removed in the 2000 data set were preserved in the 2003 data set. The total of commission and omission errors of final footprints for the 2000 data set was about 17%, a 5% increase compared to a total error of 12% for the 2003 data set (TABLE 5). The effect of point density on the residential area is more substantial as the total area-based error increases to 35%. This large error is mainly due to the fact that residential buildings are small and 2000 LIDAR measurements are not dense enough to capture the boundary of the buildings.

4.6 Conclusion

A framework including a series of algorithms has been developed to extract a building footprint from LIDAR measurements. The framework includes three major components: (1) the progressive morphological filter for separating the ground and non-ground measurements, (2) a region growing algorithm based on a local plane-fitting technique for segmenting building measurements, and (3) the Douglas-Peucker algorithm for removing noise in a footprint, an algorithm for estimating the dominant direction of a building, and an algorithm for adjusting the footprint based on estimated dominant directions. The entire process is highly automatic and requires little human aid, which is very useful for processing voluminous LIDAR measurements.

The novel algorithm for direction estimation is capable of identifying dominant directions as long as the

Table 5: Error analysis results for two datasets covering the FIU campus and a residential area. The known building footprints were provided by the FIU Planning and Facility Management Department and include 62 buildings. The known building footprint data for the residential area include 211 buildings and were derived by digitizing buildings on aerial photographs and an image interpolated from raw LIDAR measurements.

Omission Error	FIU Campus (2003)	FIU Campus (2000)	Residential (2003)	Residential (2000)
Raw footprint *vs.* Known footprint	10.4%	17.1%	6.6%	30.3%
Raw footprint *vs.* adjusted footprint	3.0%	3.1%	4.8%	8.4%
Adjusted footprint *vs.* Known footprint	10.4%	17.2%	6.0%	30.6%
Commission Error	**FIU Campus (2003)**	**FIU Campus (2000)**	**Residential (2003)**	**Residential (2000)**
Raw footprint *vs.* Known footprint	1.8%	1.4%	6.0%	5.3%
Raw footprint *vs.* adjusted footprint	3.2%	3.2%	4.9%	7.5%
Adjusted footprint *vs.* Known footprint	1.9%	1.4%	5.5%	4.8%

total length of parallel and perpendicular segments is larger than the total length of oblique segments in a footprint. The allowance of oblique segments in a footprint enables users to perform footprint refinement for complex buildings in urban environments. The algorithms for dominant direction estimation and footprint adjustment can also be applied to generalize noisy raw building footprints derived from aerial photographs and high-resolution satellite images.

Application of the framework to the FIU campus and a residential area shows that the algorithms identified building measurements from LIDAR data and extracted footprints effectively. The quantitative accuracy analysis indicates that all buildings were identified and about 12% of the area errors were committed by the proposed algorithms, despite the fact that there are several complex building shapes on the FIU campus. These results provide a good basis for refining the footprint manually in a GIS environment for engineering applications which need highly accurate footprint information.

The point density of LIDAR measurements influence the accuracy of building footprint extraction, and approximately 1 m spaced LIDAR points are needed to achieve the above accuracy. The region growing segmentation algorithm for identifying building points from non-ground measurements is critical for building footprint extraction. Experiments demonstrated that region growing segmentation based on local plane-fitting is robust and not sensitive to seed point selection.

CHAPTER 5
3D BUILDING MODEL RECONSTRUCTION

5.1 Introduction

This chapter presents a framework for automatic 3-D building reconstruction from LIDAR measurements without the help of ground plan. First, the area growing result to segment roof planes is studied. Second, 2-D topology of each building is derived by analyzing the roof plane neighboring relationship. Then domain directions of each building are estimated by using roof plane information. Finally, roof plane parameters and 2-D topologies are adjusted for the 3-D building model reconstruction.

The chapter is arranged as follows. Section 5.2 describes the algorithms that derive 3-D building model. Section 5.3 describes the sample LIDAR data set used by this study and the parameters for data processing. Section 5.4 examines the results by applying the 3-D building model reconstruction to the sample data set and discusses several factors influencing the performance of the algorithm. Section 5.4 includes conclusions.

5.2 3-D Building Model Reconstruction

5.2.1 Extraction of building roof planes

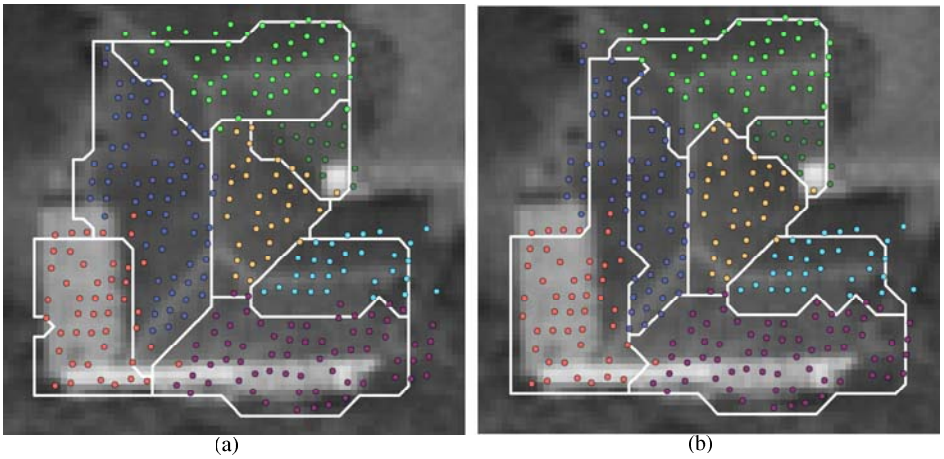
The ground and non-ground measurements are separated as a first step in the proposed framework using a progressive morphological filter [34]. We selected the progressive morphological filter because this filter identifies the ground and non-ground measurements well for the study areas that are located in a coastal urban setting with gentle slopes. Alternative filters can also be used in this step if those filters can produce a better classification. Before filtering and building identification, a 2-D array was employed to represent points falling in cells of a mesh overlaying the data set to facilitate the computation. The cell size (c_s) of the mesh is usually set to be less than the average spacing of LIDAR points to reduce information loss. Each point measurement from the LIDAR data set is assigned to a cell in terms of its x and y coordinates. If more than one point falls in the same cell, the point with the lowest elevation is selected as the array element. If no point exists in a cell, the array element for the cell is assigned by its nearest neighbor. All the successive operations are based on the 2-D array.

After the non-ground measurements are filtered out, building measurements are separated from non-building (mainly vegetation) data by using the region growing algorithm which takes the plane-fitting technique as the growing criteria [35]. The region growing algorithm is selected because it only takes few thresholds and the procedure is simple and easy to implement. Meanwhile, the region growing result to segment individual buildings is not greatly affected by the height difference Δh_T to aggregate a point and then the segmentation result for buildings is robust. However, the region growing result to segment individual roof planes, which is more important to 3-D building model reconstruction, is not robust and greatly affected by Δh_T.

The appropriate value $\Delta h_T$for roof plane segmentation can be derived from LIDAR measurement error which obeys a normal distribution. The standard deviation σ of the error is determined by the LIDAR system. In our framework, σ is analyzed and predefined as 6cm. From the statistical point of view, when a region corresponding to a plane is growing and a small value like 2σ is assigned to Δh_T, omission error is prone to occur since about 10% of measurements on the plane have deviations over than Δh_T and are missed in the growing result. On the contrary, when a larger Δh_Tlike 3σ is selected, commission error is prone to occur since measurements closely neighboring with the growing plane may be added in the growing result. A Δh_T value between 2σ and 3σ can balance the commission and omission error and derive a good segmentation result. Figure 37 compares the region growing result to segment roof planes in a building area using three different Δh_T values. The dots represent the building measurements segmented when Δh_T is 2.5*σ (15cm) and their colors correspond to the roof planes to which they belong. The white polygons as shown in Figure 37(a) and Figure 37(b) separate roof planes identified when Δh_T are 2σ (12cm) and 3σ (18cm) respectively. As shown in Figure 37(a), some measurements on the red roof plane are missed and omission error occurs. In Figure 37(b), the red plane partially covers its neighboring roof planes and commission error occurs. After comparing the segmented roof planes with the building shown in the aerial photography, we conclude that generally the best region growing result to segment roof planes is derived when Δh_T is 2.5*σ.

5.2.2 2-D Topology Extraction

2-D topology of a building is represented by a set of connected polygons which are the projections of roof facets on a horizontal plane. The edges and vertices which form the topology are derived from grouped LIDAR measurements for individual facets of a roof. To facilitate the discussion, measurements for each roof facet for a building are assigned a unique positive integer label starting from one (Figure 39(a) and Figure 39(b)). Non-building measurements surrounding the building are labeled as 0. In order to obtain the boundaries of a roof facet, the cell size of the mesh covering the data set is reduced by half and new points are inserted between old cells. The label of a newly inserted point p is determined by checking four pairs



(a) (b)

Figure 37: Region growing result to segment roof planes by applying plane-fitting algorithm to a single strip of LIDAR measurements in a building. The height difference to aggregate a point Δh_T is set as (a) 2σ= 12cm (b) 3σ= 18cm. The colored dots represent the segmented roof planes when Δh_T is 2.5σ

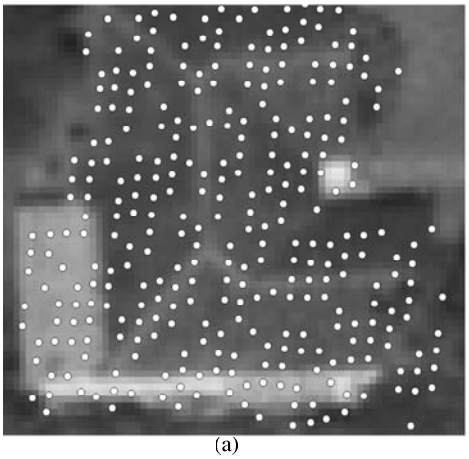
of neighbors of p as illustrated in Figure 38. If the label at the left bottom corner is the same as the label at the right upper corner, p is assigned the same label as its neighbors (Figure 38). If not, other pairs of neighbors are checked. If all four pairs of neighbors of p have different labels, p is identified as a boundary point separating two or more neighboring roof facets and is assigned a label value -1 (Figure 39(c)).

Figure 38: patterns used to determine the label of a newly inserted grid point

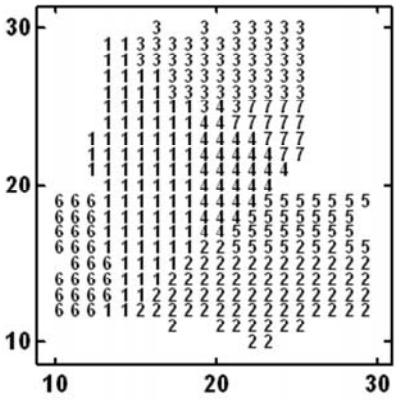
A boundary points which joins at least three different roof facets (Figure 39(d)) is classified as a vertex the 2-D topology, while a boundary point which separates only two neighboring roof facets is defined as an edge point. An edge is derived by connecting all edge points between two vertices. The vertices and edges from boundary points constitute a raw 2-D topology. The raw 2-D topology is simplified using Douglas-Peuker algorithm to reduce the noise due to the interpolation of irregularly spaced LIDAR points. Some parallel and perpendicular edges in the simplified topology are distorted because no geometric constraints are applied during the simplification (Figure 39(e)).

5.2.3 Roof and Edge Classification

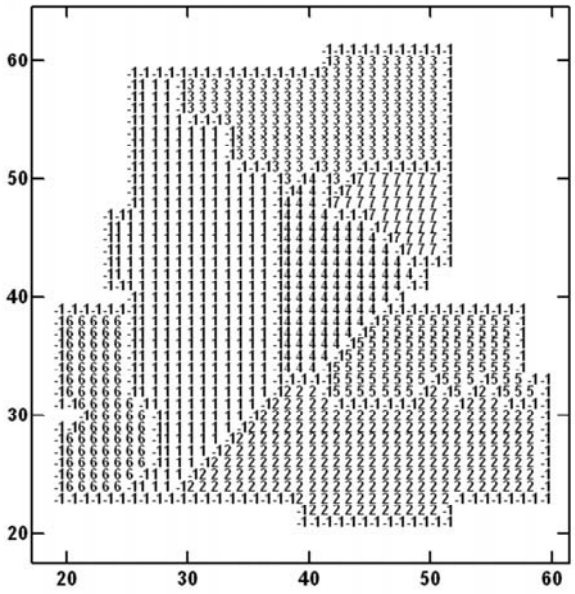
In order to correct the distorted edges, dominant building directions need to be estimated, which involves the edges and areas of building roof facets. Building facet edges can be classified into two categories: intersection and step edges. A step edge separates either two paralleling planes shown in Figure 40(a), or



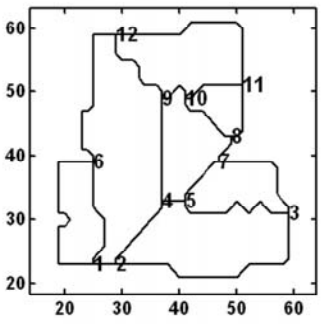
(a)



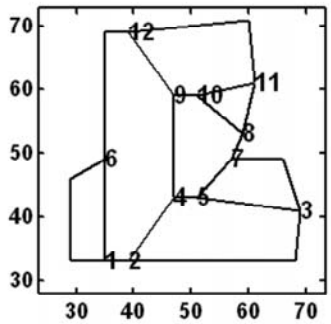
(b)



(c)

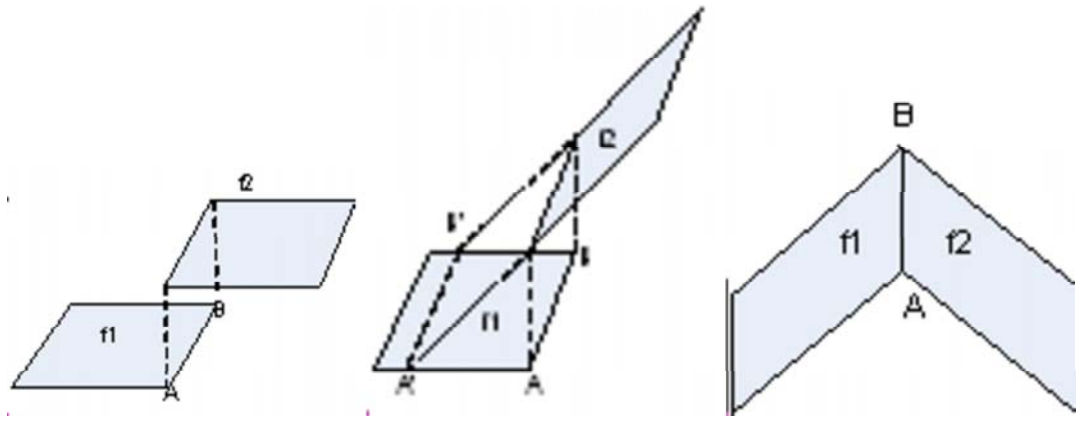


(d)



(e)

Figure 39: Demonstrates the procedures to extract the 2-D topology of a residential house. a) The aerial photography of the house and overlaid LIDAR measurements. b) Identified roof planes c) Labels of grid points on the doubled 2-D array, where boundary point is labeled -1. d) The raw 2-D topology. e) Simplified 2-D topology



(a) (b) (c)

Figure 40: Type of the edge AB on the building roof. (a) Step edge separating two paralleling planes (b) Step edge separating two intersection planes with height discontinuity (c) Intersection edge separating two intersection planes with height continuity

two intersection planes with height discontinuity shown in Figure 40(b). An intersection edge separates two neighboring roof planes with height continuity shown in Figure 40(c).

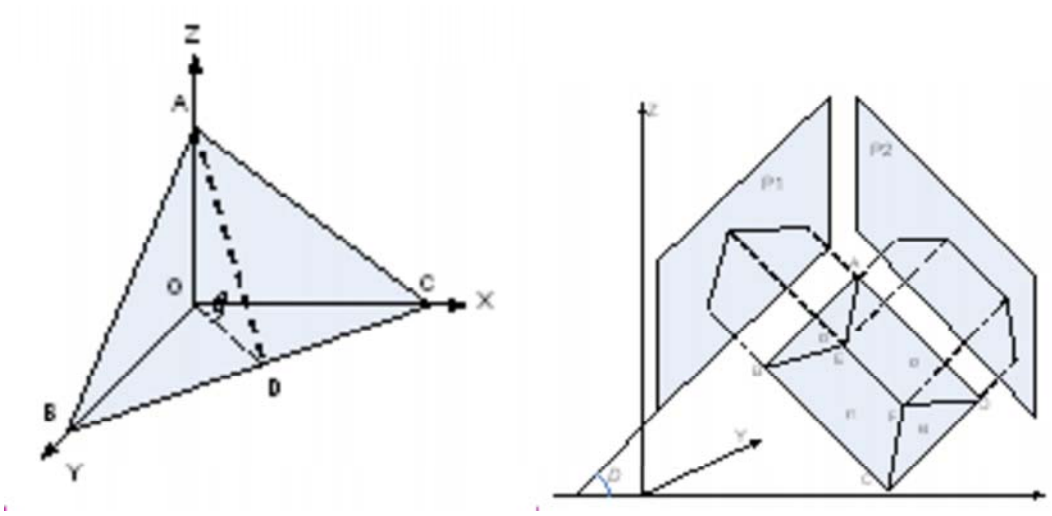
Building roofs can also be classified into two categories: flat and non-flat roof. The flat roof mainly consists of horizontal facets which are parallel to the ground. One way to determine whether a roof facet is parallel to the ground is to examine the standard deviation (s) of elevation measurements for a roof facet:

$$\min(SSD) = \sum_{(\overline{z}_k - z_k)}^{2}, var = \sqrt{\min(SSD)} \tag{32}$$

where n is the total number of measurements for the roof facet, $\overline{z}$ is the mean elevation of the measurements. If s is less than the threshold T_s, the plane is set to be horizontal. The ratio of the areas from all horizontal roof facets to the entire roof area is employed to determine the roof type. If the ratio is over than the threshold T_Ratio, the roof is defined as a flat roof, otherwise, the roof is defined as non-flat. Different methods presented in the following sections will be employed to adjust the 2-D topology for these two types of roofs.

5.2.4 Domain Direction Estimation

The dominant directions for a non-flat building can be estimated by the directions of intersection edges of all roof planes by assuming that all intersection edges are parallel to one of two dominant directions [60]. This method works fine for building measurements with a high point density (about 10 points per m2). However, many LIDAR data sets have a low point density low density, such as 1 2 points per m.2 In such cases, measurement errors and sparse points make the estimation of roof plane parameters unreliable, especially for roof facets with small areas. In addition, complex buildings have considerable number of intersection



(a) (b)

Figure 41: Demonstrates a)the θ angle of the plane P is $\angle COD$. Here vertices A, B and C are located on the plane P and segment OD is perpendicular to BC. b) A building with hipped roof consisting of four plane surfaces f_1, f_2, f_3 and f_4. Plane P_1 is perpendicular to f_1 and f_2. Plane P_2 is perpendicular to f_3 and f_4. Both P_1 and P_2 are perpendicular to the X-Y plane.

edges which are neither parallel nor perpendicular to the domain directions. For example, intersection edges, such as *AE*, *BE*, *CF* and *DF*, are oblique to both domain directions for a building with hip roofs as shown in Figure 41(b).

Zhang et al. [35] proposed a method to estimate the domain directions based on the lengths and angles of line segments of a footprint outline. This method assumes that most segments of the footprint outline are parallel to one of two domain directions. The segments oblique to both domain directions are also allowed in a footprint. Zhang et al. [35] have proven that reliable dominant directions can be derived as long as the total length of oblique segments is less than the length of segments parallel or perpendicular to two dominant directions. However, the edges separating different roof planes within a footprint outline are not included in the estimation of dominant directions in Zhang et al's method. Here we extended their method to estimate the dominant direction by considering the areas or edges of all roof facets. We assume that there are two dominant directions which are perpendicular each other for a building and most edges of roof facets are parallel to one of the dominant directions.

For a non-flat roof plane P in a 3-D coordinate system, we define θ as the counter clockwise rotation angle from the XOZ plane to a plane which is perpendicular to both plane P and XOY. The θ angle is determined by the parameters of the roof plane equation $z = A * x + B * y + C$:

$$\theta = \begin{cases} \arctan(B/A) & \arctan(B/A) \geq 0 \\ \arctan(B/A) + 180 & \arctan(B/A) < 0 \end{cases} \quad (33)$$

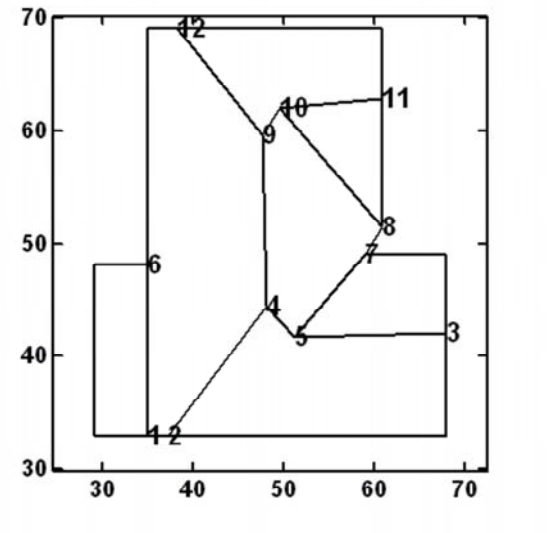


Figure 42: Demonstrates the adjusted 2-D topology of the building shown in Figure 39(e) by replacing intersection edges with the intersection line segment of neighboring roof planes

where θ is in the range [0, 180). The dominant directions of a building are derived by optimizing the rotation angle ϕ using a function (SS) of the area s_i and orientation θ_i of non-flat roof facet i. SS is determined by Equations similar to 21 -23 presented in [35]. Different from these equations, N is the total number of non-flat roof planes, s_i and θ_i are the area and θ angle of the i'th non-flat roof plane on the building. The optimum ϕ is found by comparing SS value for angles in the range [0, 90] and the minimum SS value reaches at the optimum ϕ.

Figure 41(b)shows an example of the dominant directions for a building with non-flat roof surface. Both plane f1 and f2 are perpendicular to plane P1 and their angles are the same as the angle of one dominant direction D. Both plane f_1 and f_2 are perpendicular to plane P1 and their θ angle are equal to one domain direction D. Similarly, both plane f_3 and f_4 are perpendicular to plane P2 and their θ angles are same as the angle of the dominant direction perpendicular to D.

The dominant directions for a flat roof building cannot be determined using the above equations since both a and b are 0. For such a case, we simply extended Zhang et al's method by including all edges of roof facets instead of only footprint outline into the equations for estimating the dominant directions.

5.2.5 Roof Facet Adjustment

After the dominant direction of a building is estimated, the building is rotated clockwise first so that x and y axes align with dominant direction. Then, roof facet edges of a building are adjusted by enforcing geometric constraints. The major geometric constraints are listed as follows [21]:

- Parallelism constraint – groups of edges are parallel to each others

- Planar face constraint - groups of measurements are located on a same plane face
- Right angle constraint – some edges are perpendicular to each other
- Ridge points and eave points height constraint - some ridge or eave points have same height
- Planar faces symmetry constraint - some planar faces are symmetry

In our framework, we adjust the edges by enforcing parallelism constraint on roof planes first and then on 2-D topology, considering that parallelism constraint on roof planes can partially guarantee the parallelism constraint on 2-D topology. These two parallelisms together can guarantee other constraints, such as ridge and eave point height constraints. For example, as shown in Figure 41, enforcing the parallelism constraint adjusts the four roof planes to be parallel to the dominant directions, which makes segment EF to be parallel to the $X-Y$ plane. The ridge points on EF will have same height because of this adjustment. Enforcing parallelism constraint on 2-D topology adjusts most of edges to be parallel to either domain direction. Thus, points on the segment AB, BC, CD or DA have the same heights and the eave point's height constraint is enforced automatically.

Roof Plane Adjustment

"Parallel constraint" on roof planes is enforced using the standard deviation (s) of the elevation measurements for a roof plane in Equation 32 . If s is less than the threshold T_s, the plane is set to be horizontal and the plane equation becomes $z = \overline{z}$. Otherwise, we further checking whether the plane is perpendicular to XOZ plane. First, 3-D (x, y, z) measurements for the plane are converted into 2-D points (x, z) by projecting 3-D points into the XOZ plane. Then, the line segment with equation $z = ax + b$ fitting the 2-D points are derived by minimizing the sum of squares due to deviations (SSD):

$$var = \sqrt{\min(SSD)}, \min(SSD) = \sum (z_k - \overline{z})^2, \overline{z} = \frac{\sum z_k}{n} \quad (34)$$

where $z_k^{'}$ and z_k are the estimated and measured elevations for a point and v is the variance of the deviations. If v is less than the threshold T_v, the plane is perpendicular to the XOZ plane and its plane equation is adjusted to be: $z = ax + b$. Otherwise, we check whether the plane is perpendicular to YOZ plane using similar procedure. If the non-flat roof plane is neither perpendicular to XOZ nor YOZ, its plane equation is not adjusted.

After roof planes are adjusted, the 2-D topology of a flat or non-flat roof building will be adjusted first using the snake algorithm introduced in section Topology Adjustment I. Then, the 2-D topology of a non-flat building is further refined by the method presented in section Topology Adjustment II.

Topology Adjustment I

The snake algorithm is utilized to refine the topology of roof facets of a building. The snake algorithm was introduced by [40] to locate the feature of interest in an image. First, an initial contour encloses the feature of interest is selected, and then the contour is pushed or pulled towards the target feature by minimizing energy functions which represent constraint forces. The total energy E_{total} of a contour with parametric representation $v(s) = (x(s), y(s))$ consists of internal energy E_{int} and external energy E_{ext}, and can be written as:

$$E_{total} = \int_0^1 E_{int}(v(s)) + E_{ext}(v(s))ds \tag{35}$$

The internal energy represents the forces which constrain the contour to be smooth and are formulated as:

$$E_{int}(v(s)) = (\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2) \tag{36}$$

where the first-order energy term $v_s(s)$ makes the points closer to each other and the second-order energy term $v_{ss}(s)$ favor points to be equidistant. The external energy attracts the contour towards the feature of interest, such as the high intensity area or high gradient area. The minimization of energy can be implemented using finite differences [40], finite differences [10], or dynamic programming [1]. Dynamic programming method was employed in our framework because the method guarantees global minimum and is numerically stable.

Snake algorithm based on dynamic programming assumes that each vertex on the contour is adjustable to a position in a set of positions around the vertex. Each position of the vertex corresponds to one state of the vertex. A set of the vertices on the contour corresponding to the minimal energy represent the vertices for an optimum 2-D topology. The key for deriving the optimum 2-D topology is to define appropriate energy functions for the snake algorithm. Traditional inner energy functions for a smooth contour in the Equation 36 are not suitable for the topology adjustment because there are intersection angles between two adjacent edges and the transition between two edges is not smooth. In addition, the distances between vertices of the topology are not equal in many cases.

Since the objective of footprint adjustment is to enforce parallelism on the footprint outline while keeping the adjusted footprint as close to its original location as possible, we define new energy functions *direction energy* E_{Dir} to enforce the parallelism constraint and *deviation energy* E_{Dis} to limit the deviation of the adjusted footprint from its original position. Given an edge $e' = (v', w')$ joining two vertices v' and w' on the footprint, and v' and w' are one of the possible states for vertices v and w, we define the direction energy E_{Dir} for the edge $e' = (v', w')$ as:

$$E_{Dir}(e' = (v', w')) = \begin{cases} 0 & c(e) = 1, v'_x = w'_x \\ 0 & c(e) = 2, v'_y = w'_y \\ 0 & c(e) = 3, |v'_y - w'_y| > T\_\mathit{Projection} \quad or \ |v'_x \neq w'_x| > T\_\mathit{Projection} \\ |v - w| & others \end{cases} \tag{37}$$

where $c(e)$ with values of 1, 2 or 3 represents horizontal, vertical and non-adjustable oblique edges, respectively. A non-adjustable oblique edge is the line whose projection on x or y axis is larger than a threshold $T\_Projection$. The purpose to introduce non-adjustable lines is to preserve oblique edges in the 2-D topology. All remained edges are classified as adjustable ones. A zero value is assigned to the energy function for edge e' in the 2-D topology if e' is either parallel to x and y axes which represent the dominant directions, or e' is a non-adjustable oblique line. A penalty value proportional to the length of the edge is assigned to an adjustable edge e'. Through this energy function, the adjustable horizontal and vertical edges tend to deform and align with the dominant directions, and non-adjustable oblique edges tend to remain oblique.

This energy function assigns zero to e' if an adjustable horizontally oblique segment e is adjusted to be horizontal, or an adjustable vertically oblique segment e is adjusted to be vertical, or a non-adjustable oblique segment e remains oblique. Otherwise, a penalty value proportional to the length of the edge is assigned to the adjusted edge e'. Similarly, energy function E_{Dis} is formulated as:

Energy function E_{Dis} is defined as the sum of the distance values between points on adjusted edge (v', w') and original edge (v, w):

$$E_{Dis}(e' = (v', w')) = \sum_{p \in \overline{v'w'}} D_{chess}(v, v', w, w') \tag{38}$$

where p is a points on the edge connecting points v' and w'.The smaller the deviation energy of the adjusted edge e', the closer e' is to the original edge e. In order to compute E_{Dis}, a distance transform (ref) is applied to the 2-D topology to derive a grey scale image whose pixel intensity indicates shortest distances to boundaries. The edges between roof facets are rasterized using the same grid mesh for 2-D topology extraction in the previous section. Distance values of edge cells are initialized as 0 and distance values of their direct neighbors are assigned 1. Distance values of direct neighbors of cells with values of 1 are assigned 2 and so on. Only the distance values of points in an area close to the original 2-D topology need to be calculated since each vertex on the 2-D topology is only allowed to move within a small window W_v*W_v. Figure 43 demonstrates the distance value of the image after applying distance transform to the 2-D topology shown in Figure 3(e). The distance value is calculated up to 3. The total energy for each adjusted edge e' is determined as follows:

$$E(e' = (v', w')) = C_{Dir} * E_{Dir}(e') + C_{Dis} * E_{Dis}(e') \tag{39}$$

where C_{Dir} and C_{Dis} are weights for two energy terms. Snake algorithm transforms each adjustable vertex on the 2-D topology and finds a optimum combination of vertices by minimizes the sum of energy for all edges on the 2-D topology.

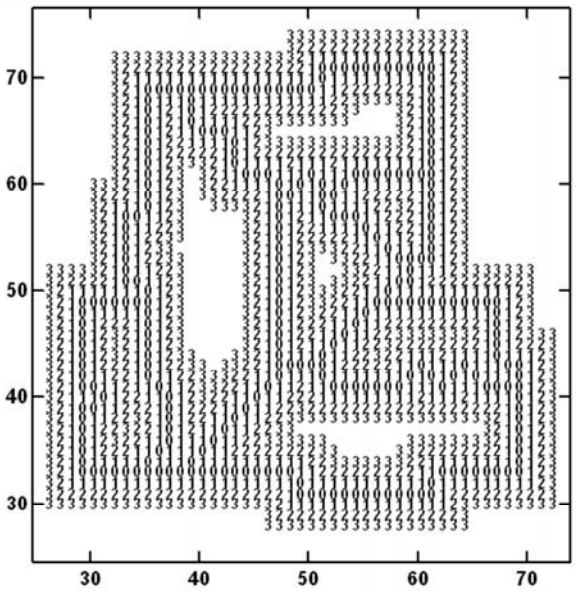
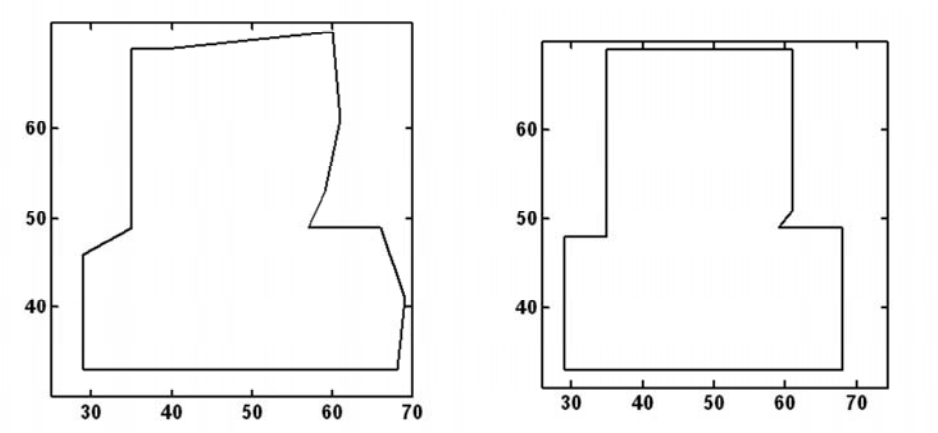


Figure 43: Image derived by applying distance transform to the 2-D topology shown in Figure 39(d).

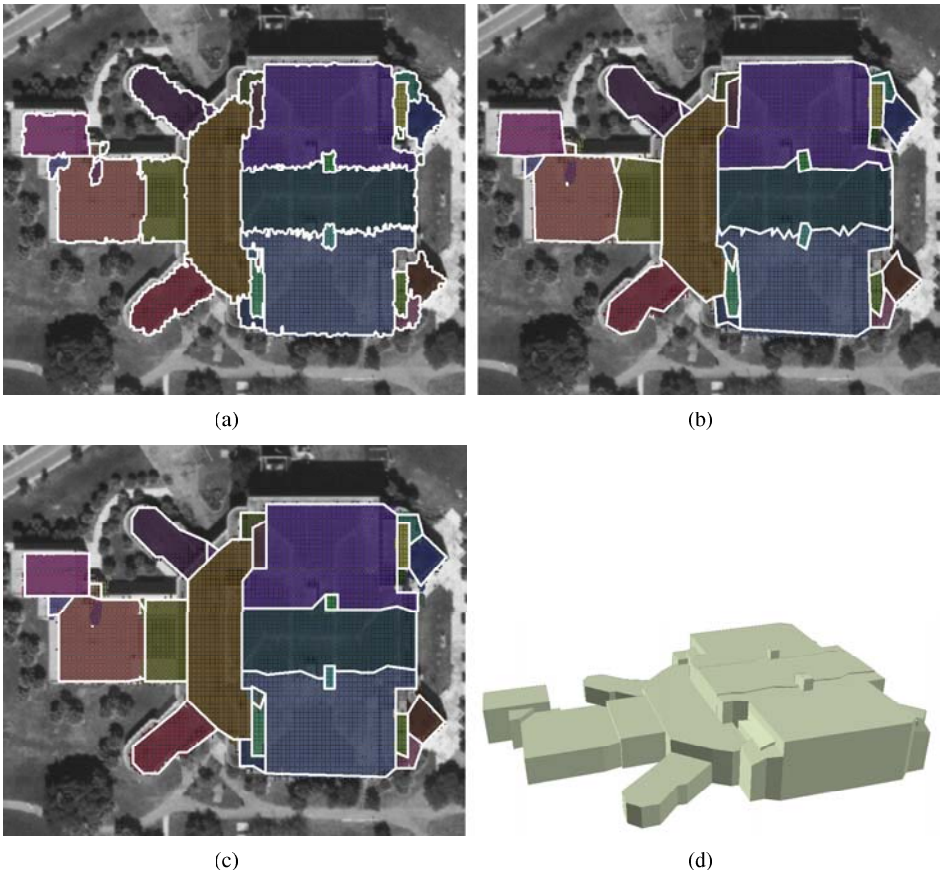
Figure 44 demonstrates one example to adjust a contour by using the snake algorithm based on these two energy functions. The contour shown in Figure 44(a) corresponds to the footprint outline of the building shown in Figure 39(e) and is adjusted to be the contour shown in Figure 44(b). We can see that most of the edges are adjusted to be either horizontal or vertical while the footprint is still close to its original position.

Traditional snake algorithm can only be applied to 1-D topology, such as a contour. A 2-D snake algorithm based on the proposed energy functions is required to refine a 2-D topology for a building. Unfortunately, general 2-D snake algorithm is a NP complete problem and is time consuming for adjusting a complicated 2-D topology. However, a subset of 2-D snake problems without complicated topology can be solved in linear time using an algorithm based on the graph theory [32]. Our experiment demonstrates that more than 95% of building topology can be adjusted by the graph-based algorithm. Figure 45 demonstrates an example to adjust the 2-D topology of a complicated building using the graph-based algorithm. The building roof consists of 46 vertices and 67 boundaries, and all the roof planes are flat. After applying the 2-D snake algorithm based on the proposed energy functions, the 2-D topology is adjusted and shown in Figure 45(c). The reconstructed 3-D building model is demonstrated in Figure 45(d).



(a) (b)

Figure 44: Demonstrates (a) Simplified footprint outline of the building shown in Figure 39(e). (b) The adjusted footprint by applying the snake algorithm to the simplified footprint outline



(a) (b)

(c) (d)

Figure 45: Demonstrates the procedures to reconstruct a 3-D building model (a) raw 2-D topology overlaid by LIDAR measurements (b) simplified 2-D topology by applying Douglas-Peuker algorithm to the raw 2-D topology (c) 2-D topology adjusted based on 2-D snake algorithm (d) Reconstructed 3-D building model

Topology Adjustment II

Some buildings, such as residential houses, mainly consist of non-horizontal roof planes which form many intersection edges. Since height continuity constraints are not enforced in the snake-based topology adjustment algorithm, errors are often produced in reconstructed 3-D building models. For instance, Figure 46(a) shows the adjusted 2-D topology by applying the proposed 2-D snake algorithm to the building in Figure 39. Artificial step walls are generated between adjacent roof planes in the reconstructed building model is shown in Figure 46(b).

To overcome these problems, we replace the edges from the snake based adjustment within the outline of the footprints with intersection segments of the neighboring planes. Before the replacement operation is performed, each edge is classified into intersection or step one. An edge is classified as the step edge as shown in Figure 40(a) if it separates two flat planes. Otherwise, we determine an intersection edge using the following value:

$$DH(e(v,w)) = \sum_{p \in \overline{vw}} |h_1(p) - h_2(p)|/n \qquad (40)$$

Where $\overline{vw}$ represents the set of grid points closest to or on the edge e. n is the number of grid points in the set and p is one grid point in the set. $h_1(p)$ and $h_2(p)$ are the elevation of p on its two neighboring planes respectively. If *DH(e)* is less than a predefined threshold *T_Edge*, edge e is classified as the intersection edge as shown in Figure 40(c). Otherwise, it is a step edge as shown in Figure 40(b), such as all the edges on the footprint outline. In our framework, *T_Edge* is conservatively set as $2*\Delta h_T$ since in the worst situation the measurement error from one roof plane can reach Δh_T , and then height difference between two neighboring roof planes with height continuity can reach $2*\Delta h_T$.

After each edge is classified, the 3D coordinates of each vertex P can be determined according to the number of intersection edges incident to it. If incident to two intersection edges, P is the joint point of three planes its 3-D coordinates can be derived directly. For example, edges connecting vertices 9-10, 9-12 and 9-4 in Figure 39(e) are classified as intersection edges. Thus the 3-D coordinates of the vertex 9 are same as the joint point of the planes 4, 10 and 12. Similarly, we can determine the 3-D coordinates of vertices 10, 4, and 5. If P connects with only one intersection edge, P is located on and can move along the corresponding intersection line segment. Most of vertices on the footprint outline also connect to one intersection edge and their 3-D coordinates can be determined by the intersection of the footprint outline and the intersection edge. In our framework, the raw footprint outline is adjusted first based on snake algorithm and then 3-D coordinates of vertices on the footprint outline can be determined by intersecting the adjusted footprint outline and the related intersection edge. 3-D coordinates of the vertex 11 in Figure 46(c) can be derived in this way, which is located on the footprint outline and incident to the intersection edge joining vertices 10 and 11. Figure 46(d) demonstrates the reconstructed 3-D building model after the 3-D coordinates of

vertices are derived. Comparing Figure 46(d) with Figure 46(b), we can see that edges connecting 4 and 5 are non-horizontal right now and the fake walls are removed. The 3-D building model looks more realistic.

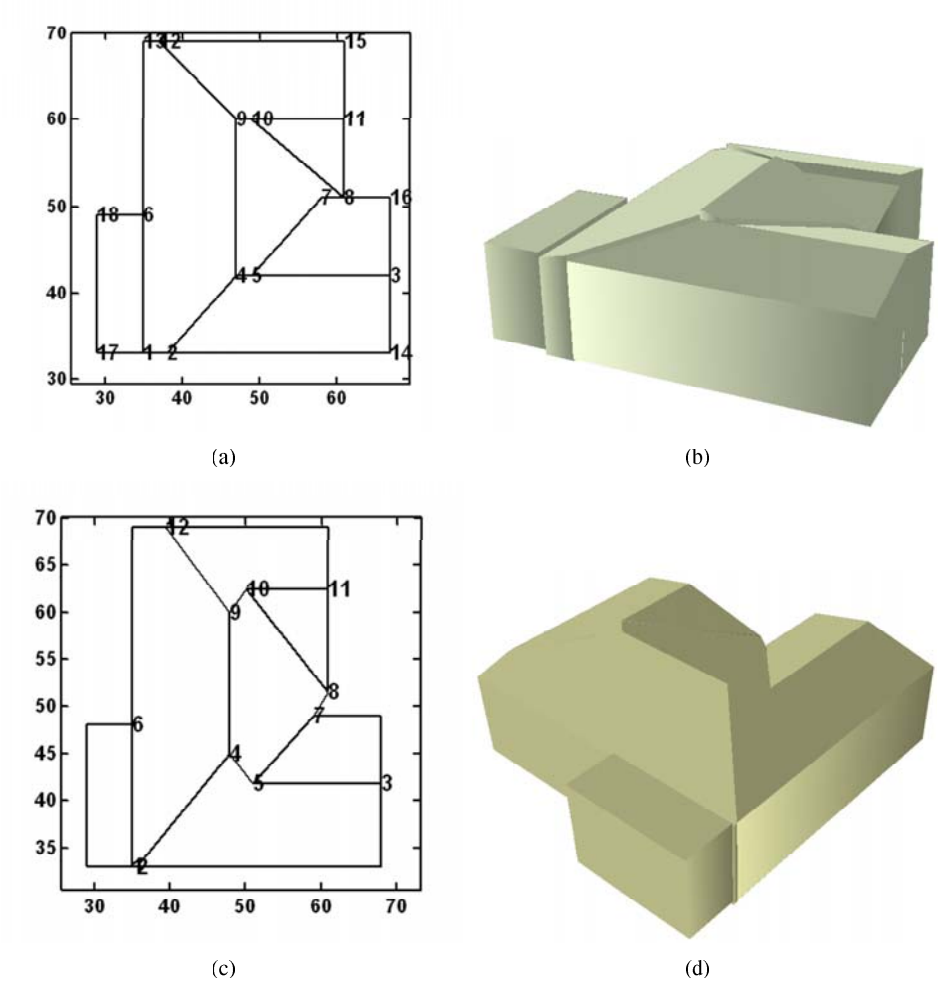


Figure 46: Compares the 2-D topology adjusted by (a) 2-D snake algorithm and (c) replacing intersection edges and the reconstructed 3-D building models (b) and (d) respectively.

5.3 Data Processing

The study area is located at and around the campus of Florida International University (FIU), covering 6 km2 of low relief topography. Surveyed features include residential houses, complex buildings, individual trees, forest stands, parking lots, open ground, ponds, roads, and canals. The data were collected on August 2003 with Optech ALTM 1233 systems operated by FIU. The Optech system recorded the coordinates (x, y, z) and intensity of the point measurements corresponding to the first and last laser returns. The data set consists of five overlapping 340-m-wide swaths of 13-cm-diameter footprints spaced approximately 1 m apart. 3-D building models were reconstructed from two test LIDAR data sets for the FIU campus and adjacent areas to examine their effectiveness. The thresholds used in our experiments for these datasets are listed in 6. It took 9, 2, and 2 min for a personal computer with a 2.8-GHz processor and a 2-GB RAM to perform morphological

filtering, building measurement identification, and 3-D building reconstruction for the FIU campus dataset. A two-dimensional array with about 7.2 million elements was employed to represent raw and interpolated points covering an area of 1.8 km2. Aerial photographs and field investigation were used to qualitatively and quantitatively evaluate the reconstructed building models. The aerial photographs were collected in 1999 at a resolution of 0.3 m.

Table 6: The parameters involved in the framework

Cell size (c_s) for snake algorithm	1 m
Douglas Distance *T_Douglas*	1.5 m
Threshold for edge adjusting *T_Projection*	2 m
Height difference to aggravate points (Δh_T)	15 cm (2.5*σ)
Threshold for plane adjusting *T_Var*	9cm (1.5*σ)
Threshold for ratio area *T_Ratio*	80%
Weight for energy E_{Dir} C_{Dir}	1
Weight for energy $E_{Dis}C_{Dis}$	5
Threshold for edge classification *T_Edge*	30cm(5*σ)
Window size for vertex adjusting W_v*W_v	5*5

5.4 Results

Both qualitative and quantitative methods were employed to measure the errors committed in 3-D building reconstruction in this study. A qualitative method checks the quality of segmentation for roof planes, estimated dominant directions, derived 2-D topology and reconstructed 3-D building models by visually comparing the extracted building models with those in maps and aerial photographs. The quantitative method examines the accuracy of the estimated domain directions.

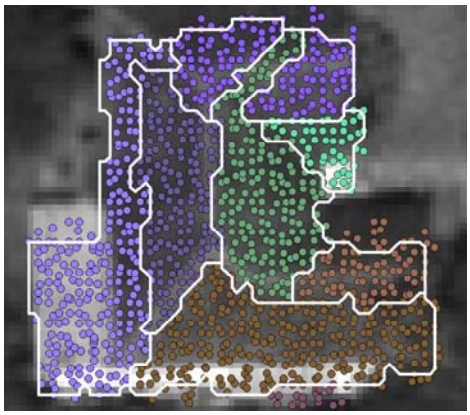


Figure 47: Roof plane segmentation result based on two overlapped strips of LIDAR measurements. Δh_T is 15cm (2.5σ)

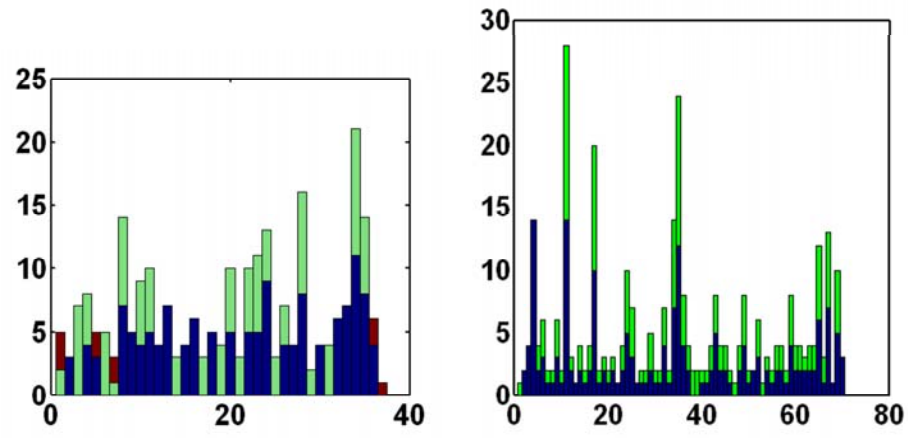
The density of LIDAR measurements has a great impact on the segmentation result. [35] increased the density of LIDAR cloud by overlapping two strips of LIDAR measurements and the region growing result to segment individual buildings is greatly improved. However, this technique is not helpful for the region growing result to segment roof planes. Figure 47 demonstrates the segmented roof planes of the same building

Figure 48: Extracted 2-D topology of the buildings in the residential dataset

shown in Figure 37 based on overlapped strips of LIDAR measurements. The data density increased from 0.5 m to 1 m. Δh_T is set as 15cm. Compared with Figure 37, there is more commission and omission error in segmenting individual roof planes. After visually checking segmentation result in the whole dataset, we conclude that single strip of LIDAR measurements provides more robust roof plane segmentation result than overlapped strips of LIDAR measurements. The reason is that elevations from different strips of LIDAR measurements deviate from each other, which makes the segmentation not robust.

The effectiveness of the algorithm to estimate dominant directions was evaluated quantitatively. The accuracy of the domain direction estimation algorithm based on three data sources, footprint outline, 2-D topology and roof planes, are compared. The residential dataset is convenient for checking the algorithm based on the these data sources since most of the buildings (140/143) are known to be horizontal and consist of intersection roof planes which can be used to estimate domain direction. 37 buildings among the total 140 horizontal ones are estimated to be non-horizontal by at least one data source and the estimated domain direction form small intersection angles with the horizontal axis, which represents the estimation errors. Errors from data sources footprint outline, topology and roof planes are displayed by green, blue and red bars in Figure 49(a) respectively. We can see that most of errors come from footprint outline and 2-D topology and the estimation result based on roof planes is much better that that based on other two data sources. Figure 49(b) compares the errors from the footprint and 2-D topology for buildings on the FIU main campus. Errors from roof planes are not provided since all buildings in this dataset consist of flat roof planes which cannot be used to estimate

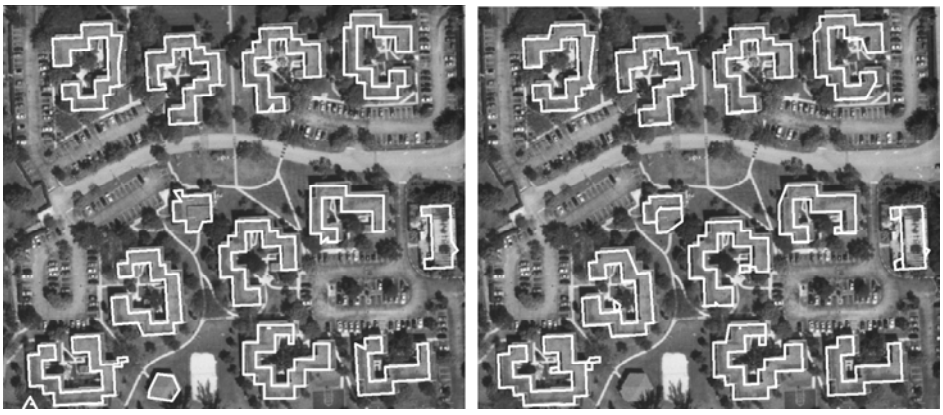
the domain direction. The result show that estimation result based on footprint outline is similar to that based on 2-D topology



(a) (b)

Figure 49: Error analysis of the domain direction estimation in (a) residential dataset (b) FIU main campus based on footprint outline, 2-D topology and roof planes, which are represented by green, blue and red bars respectively. Vertical axis in the bar graph represents errors (the intersection angle between the estimated directions with horizontal direction) and horizontal axis represents the corresponding building.

The effectiveness of the snake algorithm based on these proposed energy functions was evaluated by comparing the footprint adjusted by the snake algorithm and the adjusting operations in [35]. Figure 50 compares the footprints of some buildings on the FIU main campus extracted based on these two methods. These two methods produce very similar geometric shapes. After careful visual checking, we find that the snake based algorithm generates more accurate details. Meanwhile, the commission and omission error from the snake algorithm are decreased.



(a) (b)

Figure 50: Footprint of bilidng in FIU main campus adjusted by (a) snake algorithm (b) adjusting operations

The reconstructed 3-D building models for the dataset around FIU main campus are demonstrated in Figure 51. The quality of these models is evaluated by visual checking since no practical 3-D building models

are provided. Meanwhile, it is very difficult to extract 3-D building models from LIDAR measurements manually. We guess the real building model by visualizing LIDAR measurements and checking the photography at the same time. Most of buildings (130/140) are reconstructed well.

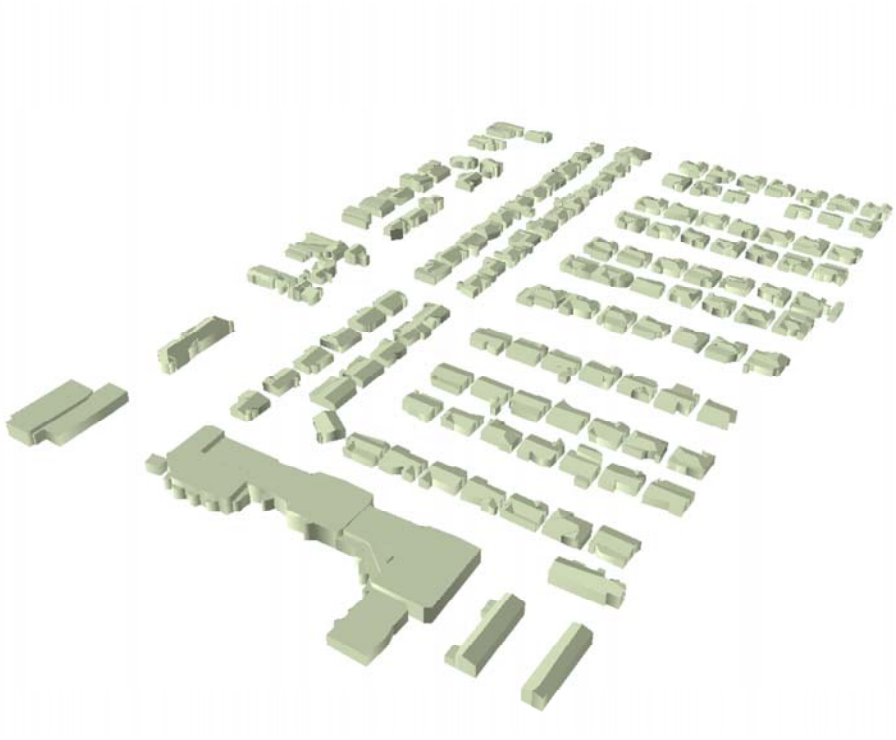


Figure 51: Reconstructed 3-D building models in the dataset

As we realized, some useful geometrical constraints are not enforced in our framework, such as "intersection constraint" which enforce more than three roof planes intersect at the same point. For example, our framework can only detect and adjust vertex which is the intersection point of three neighboring roof planes like the point enclosed by the red oval shown in Figure 52. The point enclosed by the green oval, which is actually the intersection point of six roof planes, cannot be detected by our algorithm. To detect and adjust such kind of vertex, we can snap vertices close to each other together. However, it is not easy to propose a robust snap operation which works well in most cases.

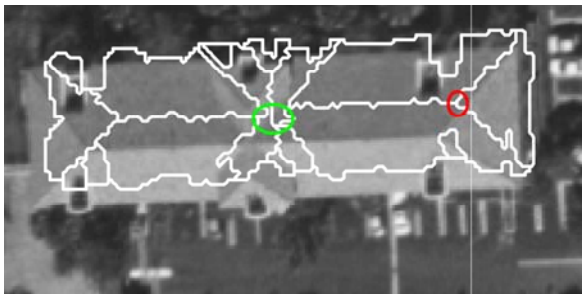


Figure 52: A building with two kinds of intersection point.

5.5 Conclusion

A framework including a series of algorithms has been developed to automatically reconstruct 3-D building models from LIDAR measurements. The framework includes five major components: 1) a progressive morphological filter for separating the ground and non-ground measurements; 2) a region-growing algorithm based on a local plane-fitting technique for segmenting building measurements; 3) a method for 2-D topology extraction; 4) an algorithm for estimating the dominant direction of a building; 5) an algorithm for adjusting the 2-D topology based on the building type. The entire process is highly automatic and requires little human aid, which is very useful for processing voluminous LIDAR measurements.

The region-growing algorithm to segment roof planes is critical to the performance of the extracted 2-D topology and reconstructed 3-D building model. The segmented roof planes based on single strip of LIDAR measurements has proven to be much more robust than overlapped strips of LIDAR measurements. However, the segmentation result in some buildings is still not satisfactory and further improvement is required. The algorithm for 2-D topology extraction does not involve any threshold and is very concise and robust. The algorithm for direction estimation utilizes roof plane information and makes the estimation more accurate for buildings with non-flat roof planes. The snake algorithm can adjust most of buildings well. The application of the framework to the FIU campus and a residential area shows that the algorithms reconstructed 3-D buildings effectively.

CHAPTER 6
2D SNAKE ALGORITHM

In this chapter, an algorithm is introduced to extend the traditional snake algorithm from problems with 1D topology to 2D topology. It can be used to adjust 2D topology, multi-objects segmentation or tracking.

6.1 Introduction

Significant research effort has been placed on developing deformable models to determine a surface or a contour having optimal properties in the past decade [29]. Applications of deformable models include medical image analysis, geometric modeling, and tracking of non-rigid objects. The deformable models which are also called "snakes" or active contours were first introduced by Kass et al. in 1988 [20]. Snakes usually start with an initial one dimensional (1D) contour, two dimensional (2D) surface, or even three dimensional (3D) volume [20] [13] [56] close to a target model, and then gradually deform contour/surface while minimizing energy functions so that the resulting contour/surface best matches the target boundary/topology of the object [13] [57]. The solution to the snake problem often involves the derivation of an energy function and minimization of the energy function. Methods that can be used to minimize the energy function include simulated annealing (SA) [9], variational techniques [26] and stochastic relaxation techniques [28].

Dynamic programming is an optimization approach that finds global minima by analyzing a collection of admissible solutions. In dynamic programming, constraints are often placed on the set of allowable solutions, thus reducing the computational complexity. For example, in the case of 1D active contours, the set of admissible solutions are in fact the set of all allowed curves that connect the start point with the end point. Unlike the variational method, dynamic programming can be directly applied to a discrete grid without approximations. Amini et al. [1] devised a time-delayed discrete dynamic programming algorithm to minimize the energy for 1D active contours. The discretization of the contour energy $E(C)$ is represented by $E(C) = E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$, where $C=\{v_{1,\ldots,}v_n\}$. Each contour point v_i is allowed to only take on m possible values. Instead of using exhaustive enumeration to find the minimum of $E(C)$, discrete dynamic programming method computes the global minimum in an efficient way. However, dynamic programming method is inherently restricted to problems with 1D topology such as a contour [29] [1] [12]. The dynamic programming method outlined for 1D topology cannot be directly extended to the bipartite case [64]. The

2D snake problems, such as the reconstruction of surfaces cannot be solved efficiently through the current dynamic programming method.

This chapter presents an algorithm to minimize the energy function associated with 2D snakes which represent 2D surfaces by using connected deformable graphs controlled by vertices and edges. The first objective is to develop the algorithm including a set of graph operations which are capable of reducing certain types of 2D snakes to single vertices. The time complexity of the proposed algorithm will be polynomial with guaranteed optimality if 2D snakes are reducible. The second objective is to apply the proposed method to refining building topology extracted from airborne light detection and ranging (LIDAR) measurements to examine the effectiveness of the algorithm. The third objective is to analyze the time complexity of the general 2D snake problem and prove it is NP-complete.

The chapter is organized as follows: Section 6.2 describes the proposed approach by first giving a formal definition of the 2D snake problem and analyzing the discretized form of the energy function, then describing the details of the proposed graph reduction based algorithm, and finally proving polynomial time complexity of the algorithm. Section 6.3 presents the application to refining building topology from LIDAR measurements. Section 6.4 presents a formal proof of the NP-completeness of the general 2D snake problem, and Section 6.5 concludes the chapter.

6.2 Graph Reduction Approach for 2D Snakes

Snake-based method to detect a complex graph structure (topology) involves minimizing the energy of deformable topology. A domain-specific energy function whose minimum represents the target topology is constructed first. Then, the energy function is gradually minimized starting with an approximate topology which is usually derived from low level image processing and pattern recognition operations. Finally, the topology corresponding to the minimum of the energy function is derived to represent the target topology. The key of this procedure is to develop an effective algorithm for energy minimization. We present a graph reduction algorithm in the following section to find a global minimum for a 2D snake.

6.2.1 The Cost Function

We represent the deformable topology (2D snake) by a weighted graph G=(V, E). Each vertex v is associated with an uncertainty list UL_v= $\{s_v^m = (x_m,y_m) \quad | \quad m=1, 2\ldots \ |UL_v|\}$, which is a list of points whose distance to this vertex is less than a pre-specified distance d. The number of points in the uncertainty list represents the number of possible states of the vertex v. For each state s_v of v($s_v \in UL_v$), there is a corresponding energy value, denoted by $EV(v, s_v)$. Correspondingly, for an edge e= *(v, w)* connecting two vertices v and w, there are $|UL_v| \times |UL_w|$ allowable states and each state is associated with an energy value, $EE(e = (v, w), s_v, s_w)$, $s_v \in UL_v$ and $s_w \in UL_w$ Figure 53. shows an example of vertices v and w with three possible states and

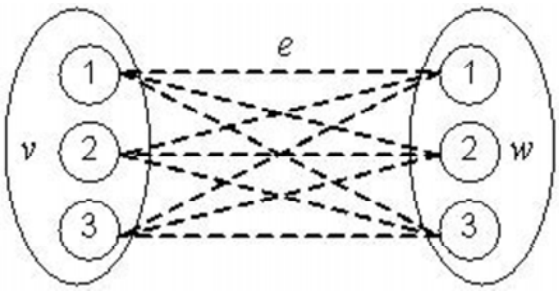


Figure 53: The edge $e = (v, w)$ and its nine possible states. Each vertex (v or w) has three possible states

the total number of states of the edge connecting these two vertices is nine. Note that the energy of an edge only depends upon two vertices to which the edge is connected. Assuming that a list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}), s_k \in UL_k\}$ represents a state of all vertices in G, we define the cost (energy) for the deformable topologies as a sum of cost for each vertex and edge in G which is given by

$$EG(G) = \sum EE(e = (v, w), s_v, s_w)_{s_v \in UL_v, s_w \in UL_w} + \sum EV(v, s_v)_{s_v \in UL_v} \quad (41)$$

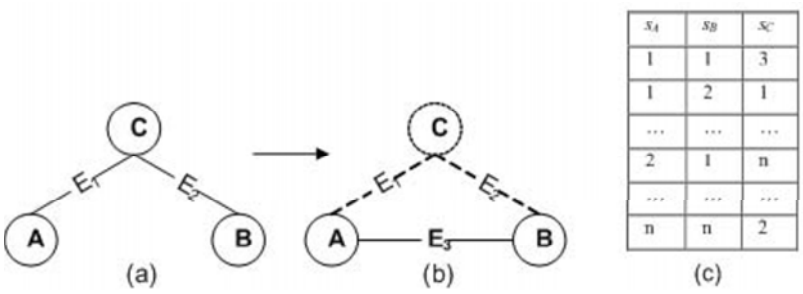
The specific form of the energy function EV and EE for edges relies on the application need. Minimizing the total cost (energy) generates the optimal topology that best fits the given object. In order to facilitate the presentation of the algorithm in the next section, we also assume that there is only one minimum for EG and corresponding state of all vertices in G is unique.

6.2.2 A Graph Reduction Based Implementation

Let us consider the problem of finding the optimal topology that has the same graph structure as the given initial topology and fits best the target topology. The corresponding energy minimization is denoted by $\min_S EG(G)$, where S is a state list for all vertices in G. A brute force implementation of searching the minimum will try all possible combinations of state lists for vertices and edges, which involves $\prod_{v \in V} |UL_v|$ steps, making the time complexity exponential. In Section 6.4, we proved that the general 2D snake problem is NP-Complete, which means no algorithm can resolve general 2D snake problems in polynomial time. However, it is still possible to resolve a special subset of 2D problems in polynomial time. This is similar to the case of 3SAT (Boolean satisfiability problem) in which 3SAT is NP-complete, but its subset 2SAT is not NP-complete. In this section, a method is proposed to derive the global minimization by progressively simplifying the graph using the following four graph reduction operations.

Type I operation

Given a vertex **C** which connects to two other vertices **A** and **B** via edges $\mathbf{E}_1$ and $\mathbf{E}_2$ (i.e., the degree of **C** is two) as shown in Figure 54, the vertex **C** and the two edges $\mathbf{E}_1$ and $\mathbf{E}_2$ can be reduced to a new edge $\mathbf{E}_3$



s_A	s_B	s_C
1	1	3
1	2	1
...	...	...
2	1	n
...	...	...
n	n	2

Figure 54: Graph and associated state retrieval table for Type I operation.

that connects **A** and **B**. The energy of the new edge $\mathbf{E}_3$ is determined by the energy of the removed vertex **C** and edges $\mathbf{E}_1$ and $\mathbf{E}_2$through the following equation:

$$EE(E_3 = (A, B), s_A, s_B)_{s_A \in UL_A, s_B \in UL_B}$$
$$= \min_{s_C \in UL_C} \{EE(E_1 = (A, C), s_A, s_C) + EE(E_2 = (B, C), s_B, s_C) + EV(C, s_C) \quad (42)$$

Equation (42) indicates that the energy of $\mathbf{E}_3$, given a pair of states s_A and s_B for the vertices **A** and **B**, is determined by the minimum sum of energies of $\mathbf{E}_1$ and $\mathbf{E}_2$ and **C**. The state s_C of **C** that minimizes Equation (42), for a given pair of s_A and s_B, is recorded in a **State Retrieval Table** as shown in Figure 54c when Type I operation is implemented. For example, the first row in Figure 54c indicates that the energy *EE* in Equation (42) reaches its minima when s_C equals 3, given s_A=1 and s_B=1.

For the reduced graph $G'(V', E')$ where $V' = V$-**C** and $E' = E$-E_1-E_2 +E_3, the following lemma holds:

LEMMA I: *The reduced graph G' has the same minimal energy as that of G after a type I reduction operation is applied.*

Proof:

To facilitate the discussion, we denote the state list of vertices that minimizes the energy of a graph G as *minS*(G) and the corresponding minimal energy as *minE*(G). The first step to proof this lemma is to verify that s_C,and $\mathbf{E}_3$ must satisfy Equation (42), if s_C belongs to *minS*(G).

We assume that there is s'_C (s_C'≠ s_C) that belongs to *minS(G)*, but does not satisfy Equation (42). For fixed s_A and s_B, we can find a s_C that makes the energy related to vertex **C** minimum in terms of Equation (42) since the role of **C** in the global minimization is only dependent on the states of **A** and **B**, i.e.

$$EE(E_1 = (A, C), s_A, s_C) + EE(E_2 = (B, C), s_B, s_C) + EV(C, s_C) <$$
$$EE(E_1 = (A, C), s_A, s'_C) + EE(E_2 = (B, C), s_B, s'_C) + EV(C, s'_C)\}$$

If other vertices remain unchanged in the state list of *minS(G)* and s'_C is replaced by s_C, resulted $E(G)$ will

Figure 55: Type II operation.

have a value less than $minE(G)$, which contradicts the fact that $minS(G)$ which include s'_C is the state list that minimizes $E(G)$.

The second step of proof the lemma is to demonstrate $minE(G') = minE(G)$. Given that $minS(G)$ which includes s_A, s_B, and s_C that generates $minE(G)$. We can construct the minimum energy related to E_3 in G' using s_A, s_B, and s_C in terms of Equation (42). Remaining vertices in G' are assigned with the corresponding state values in $minS(G)$, which forms a new state list s' where $s' = minS(G) - s_C$. In such a condition, the energy of G' is the same as $minE(G)$. Since s' is only one state of $S(G')$, there could be other state lists that make $E(G')$ smaller. Therefore, we have $minE(G') \leq minE(G)$.

On the other hand, let s_A, s_B belong to $minS(G')$ and s_C be the state of the vertex **C** derived from Equation (42) given fixed s_A and s_B. The energy of G corresponding to the state list $s = minS(G') + s_C$ is the same as $minE(G')$. Since s is only one of the states of $S(G)$, there could be other state lists that make E(G) smaller. Therefore, we have $minE(G) \leq minE(G')$. Based on the above analysis, we can conclude that $minE(G)$ is equal to $minE(G')$.

Type II operation

Type II atomic graph reduction operation is shown in Figure 55. Given a pair of vertices **A** and **C,** if there is more than one edge connecting these two vertices in G, those edges can be reduced to one single edge between **A** and **C**. The energy of the new edge is the sum of the energies of all edges between **A** and **C**, as given in Equation (43). For this operation, there is no need to keep a state retrieval table because the energy of the new edge $\mathbf{E}_3$ is determined solely by the vertices **A** and **C** and no other vertex is involved in this operation.

$$EE(E_3 = (A, C), s_A, s_C)_{s_A \in UL_A,\ s_C \in UL_C} = \sum_{i=1}^{k} EE(E_i = (A, C), s_A, s_C) \tag{43}$$

where k is the total number of edges connected **A** and **C**, and k is equal to 2 in Figure 55.

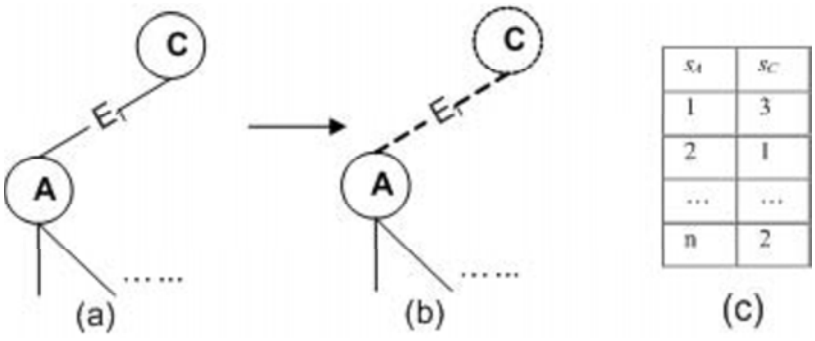


Figure 56: Graph and associated retrieval table for Type III operation.

Type III operation

Type III atomic graph reduction operation is illustrated in Figure 56. where the vertex **C** connects to one single vertex (**A**) only (i.e., the degree of **C** is one). In this case, the vertex **C** and the edge $\mathbf{E}_1$ which connects **C** to **A** can be reduced to **A.** After reduction, the energy of **A** is updated as follows.

$$EV(A, s_A)_{s_A \in UL_A} = EV(A, s_A) + \min_{s_C \in UL_C} \{EE(E_1 = (A, C), s_A, s_C) + EV(C, s_C)\} \quad (44)$$

By Equation (44), the minimum sum of energies of **C** and $\mathbf{E}_1$, for each state of **A** (s_A), will be added to the old $EV(A, s_A)$. Similar to the situation in Type I operation, the s_C that minimizes the sum of the energies of **C** and $\mathbf{E}_1$, will be recorded in the **State Retrieval Table** in Figure 56c. As the states of **C** and $\mathbf{E}_1$ in the global minimization are only dependent on the state of **A**, if s_A belongs to the list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}), s_k \in UL_k\}$ that minimizes the total energy of G, s_C must also in S. Therefore, the minimization of the total energy $EG(G)$ can be reduced to minimize the total energy of the reduced graph $G'(V', E')$, where $V' = V$-C and $E' = E$-E_1.

By applying the above three atomic graph reduction operations recursively, graphs with simple structures can be reduced to one single vertex $\mathbf{F}_G$, and the minimum energy of the original graph $\min\{EG(G)\}$ becomes $\min_{s_{F_G} \in UL_{F_G}} \{EV(F, s_{F_G})\}$. For the traditional 1D snake problem, the original graph is an open polygon as shown in Figure 57. Denote $n = |V|$ and $m = |UL|$. We can recursively apply the type III operation to remove the leftmost vertex. After $n-1$ operations, the original graph can be reduced to a single vertex. For each removed vertex, the state retrieval table has m entries and the space complexity is $(n-1)\times m$. In addition, to determine an entry in the state retrieval table, m calculations are needed. Thus, the total time complexity is $(n-1)\times m^2$, which is the same as reported in [1]. Not many types of connected graphs for 2D snakes can be reduced to one single vertex by just applying the above three atomic operations. In order to enable the proposed method to reduce more connected graphs for 2D snakes, we introduced the type IV operation.

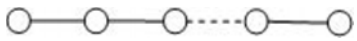


Figure 57: Topological graph G for a classical active contour problem.

Type IV operation

For a given pair of vertices **A** and **C**, as shown in Figure 58a, we first find a connected subgraph $G_{AC} = (V_{GAC}, E_{GAC})$ in G, which only connects to the vertices **A** and **C**, but not to any other vertex in G. For example, the subgraph within an oval in Figure 58a is such a connected subgraph between **A** and **C**. Then the three atomic reduction operations (Types I, II, and III) are applied to subgraph G_{AC}. If G_{AC} can be reduced to a single vertex (e.g., the vertex **E** in Figure 58c), the type IV operation will be applied to replace G_{AC} with a new edge connecting **A** and **C** directly (e.g., the edge E_{AC} in Figure 58d).

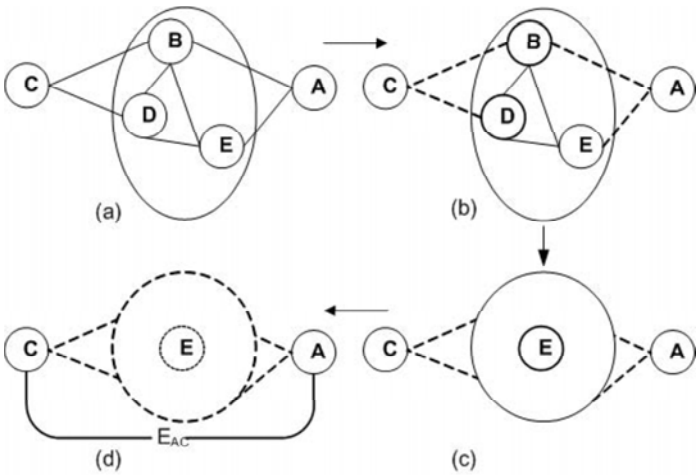
The minimized energy of G_{AC} as a portion of the minimum energy of the whole graph G is not only determined by the vertices of G_{AC}, but also the pair of s_A ands_C to which it connects. In operation IV, we assign the energy of each edge connecting **A** or **C** in subgraph G_{AC} to that of the vertex in G_{AC} which connects the edge. In the example shown in Figure 58b, E_{BC}, E_{DC}, E_{EA}, and E_{BA} are such edges connecting the subgraph with **A** and **C**. Given a pair of states (s_A,s_C) for **A** and **C**, for each vertex v which belongs to the subgraph and connects to **A** and/or **C**, we change its energy into

$$EV(v, s_v)_{v\in V_{G_{AC}}, s_v\in UL_v} = EV(v, s_v) + \sum_{(w,s_w)\in\{(A,s_A),(C,s_C)\}} EE(e = (v, w), s_v, s_w) \tag{45}$$

Let $\mathbf{F}_{GAC}$ be the single vertex reduced from G_{AC} by Types I to III atomic operations for given a pair of states (s_A,s_C) for **A** and **C**. The minimum energy of $\mathbf{F}_{GAC}$ is the same as that of G_{AC}, which is transferred to the energy of the new edge E_{AC}as follows:

$$EE(E_{AC} = (A, C), s_A, s_C)_{s_A\in UL_A, s_C\in UL_C} = \min_{s_{F_{G_{AC}}}\in UL_{F_{G_{AC}}}} EV(F_{G_{AC}}, s_{F_{G_{AC}}}) \tag{46}$$

The state list of vertices in G_{AC}that minimizes the energy of G_{AC}, given (s_A,s_C) for **A** and **C,** is denoted as $S_{G_{AC}}$and shown in the table of Figure 58e. Similar to the discussions for the three atomic operations, if s_A and s_C are in the list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}), s_k \in UL_k\}$ that minimizes the total energy of G, the states in $S_{G_{AC}}$are guaranteed to be also in S. After applying the type IV operation to G, the problem of minimizing $EG(G)$ can be reduced to the minimization of $EG(G')$ where G' is the reduced graph.



s_A	s_C	s_B	s_D	s_E
1	1	3	2	1
1	2	1	3	4
…	…	…	…	…
2	1	2	3	1
…	…	…	…	…
n	n	1	2	4

(e)

Figure 58: Graph and associated retrieval table for Type IV operation.

6.2.3 Finding Connected Subgraphs

By using operation IV together with operations I, II, and III, we can reduce the computation in finding the global minimum for some complicated graphs. The important step for applying operation IV is to find a reducible subgraph G_{AC}. We have developed the following algorithm which finds a subgraph for the type IV operation.

1. Set *scope* as 1. Initialize a set S_{ij} as empty for each pair of vertices i and j, $i, j \in V$(V is the set of graph G's vertices)

2. For each pair of vertices i and j

   - If($scope$ = 1)

     Find the set S_{ij} of vertices directly connected to vertices i and j

     Else

     Find the vertices directly connected to vertices in S_{ij} and add them into set S_{ij} except vertices i and j.

   - Examine whether S_{ij} can be partitioned into subsets which forms connected subgraphs connected only to vertices i, j. If any connected subgraph is found and can be reduced to a single vertex by the three atomic operations, the algorithm terminates. Otherwise, continue checking the next pair of vertices

3. increase *scope* by 1

This algorithm gradually expands the search scope from the vertices with direct connection to i and j to the vertices with indirect connection to i and j, and stops whenever a reducible subgraph is found. The advantage of this progressive algorithm is its efficiency – we start with the smallest search scope and expand it only when it is needed. In many cases, we can find the connected subgraph by searching only the direct neighbors of i and j. For example, let i=2 and j=6 be two vertices in Figure 59a. After Step 1, S consists of {1, 3, 4, 5, 7, 8, 11} and will be partitioned into S_1 = {1, 3, 4, 5}, S_2 = {7, 8} and S_3 = {11} at step 2. Since the connected subgraph for set S_1 connects to i and j only, it will be retuned as a connected subgraph G_{26} for i=2 and j=6. However, indirectly connected vertices for 1 and 2 need to be searched to find a subgraph G_{12} in Figure 60. In this example, S={3, 4, 7, 8, 9, 10, 13, 14} after Step1 and will be partitioned into S_1 = {3, 4}, S_2 = {7, 8}, S_3 = {9, 10}, and S_4 = {13, 14} at Step 2. Since S_1, S_2, S_3, $S_4$are not only connected to 1 and 2, but also other vertices, the algorithm will continue and go back Step 1, where S is expanded to the set {3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}. Then S is partitioned into S_1 = {3, 4, 5, 6, 7, 8} and S_2={9, 10, 11, 12, 13, 14}. S_1 or S_2 is found to be the connected subgraphs for i=1 and j=2.

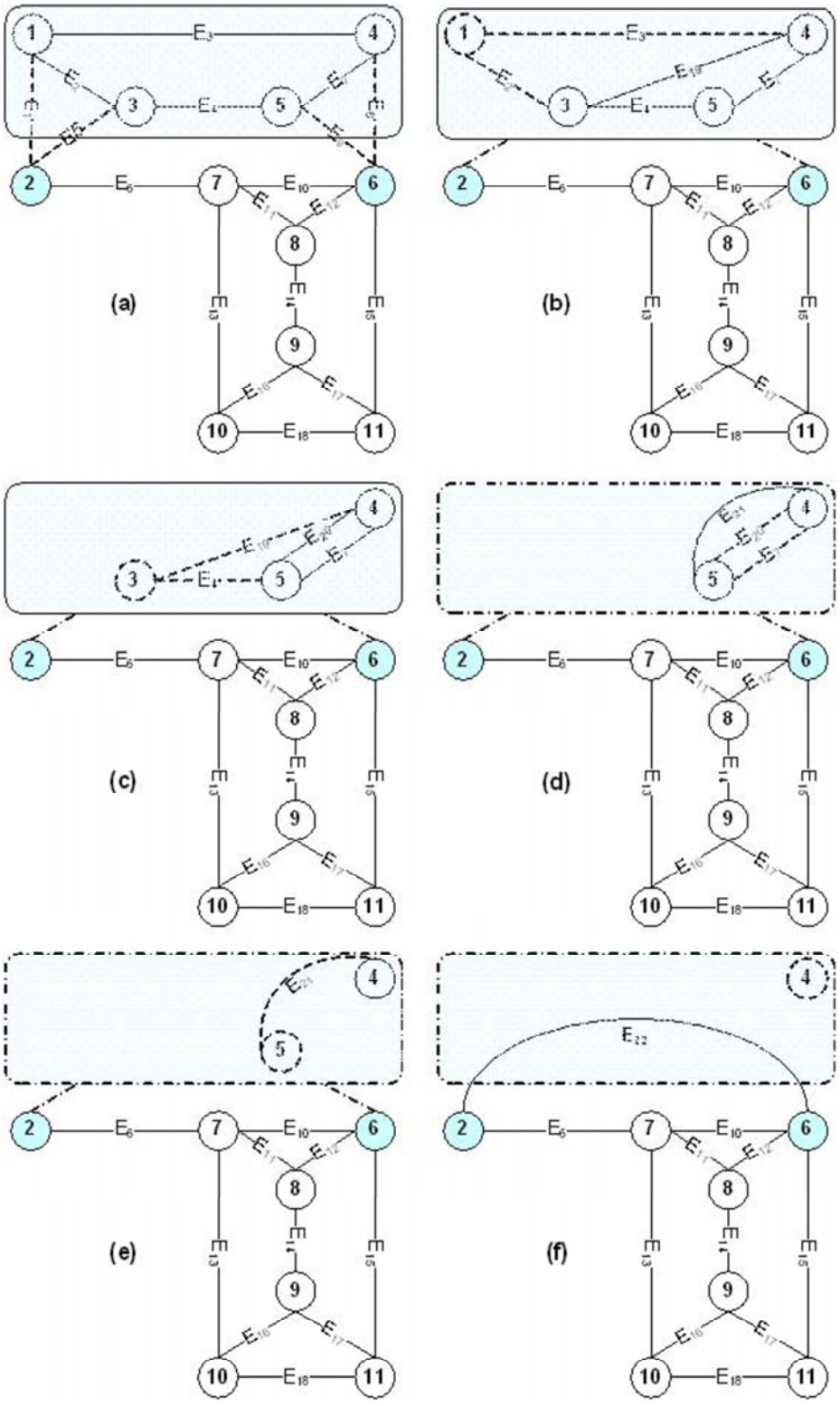


Figure 59: An example of graph reduction.

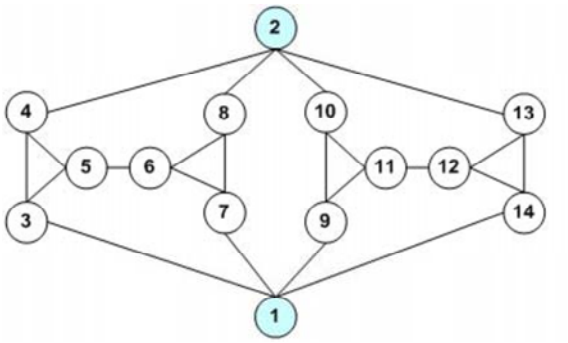


Figure 60: Another example of graph reduction.

6.2.4 The Algorithm of Graph Reduction

In combination with operations I, II, III, IV, and finding connected subgraph algorithm, we have developed the following procedure to reduce a graph G and derive the state list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}), s_k \in UL_k\}$ that minimizes the total energy of G.

Step 1: Reduce the graph by Types I to III operations

Without loss of generality, we scan the vertices in their numerical order, starting from vertex 1. First, each vertex in the current graph G is checked to see whether any of the three atomic reduction operations can be applied to the vertex. Then atomic reduction operations are performed to qualified vertices to reduce the current graph G. The information related to operations is recorded into an array **AppliedOperations.** If the current graph Gcan be reduced to a single vertex $\mathbf{F}_G$, the procedure goes to Step 3 to construct the state retrieval table. Otherwise, go to Step 2 to further reduce the graph by a type IV operation.

Step 2: Reduce the graph by Type IV operation

Try to find one reducible subgraph $G_{ij}$by using the algorithm introduced in Section 6.2.3 and perform a type IV operation to reduce the graph. The information related to this operation will be recorded and added to array **AppliedOperations**. Then the procedure goes back to Step 1 to reduce the current G by the three atomic operations again. If no reducible subgraph can be found, the algorithm will exit and report a message "This graph is irreducible."

Step 3:Construct the state retrieval table

Scan the operations recorded in array **AppliedOperations** from its start and apply each reduction operation to G as follows.

- For each Type I/II/III atomic operation, the energy of edges and/or vertices involved will be updated according to Equations (42)-(44). The state retrieval table for each operation is constructed as well.
- For each Type IV operation, we store the atomic operations used to reduce the subgraph G_{ij} into a temporary array. Given each pair of s_i and s_j, the energy of internal vertices in G_{ij} will be first updated

according to Equation (45). Then we scan the atomic operations in the temporary array forward from the start, and update the energy of edges and/or vertices involved in each operation and construct the state retrieval table if applicable. After all the operations in the temporary array are performed, go to Step 4 to compute the minimum energy of the subgraph G_{ij} and retrieve the state list minimizing the energy of G_{ij}. This state list, together with s_i and s_j, is stored as one row in the state retrieval table for the Type IV operation.

After all the operations in the array **AppliedOperations** are performed, go to Step 4 to compute the minimum energy of G and retrieve the state list minimizing the energy of G. Finally, the minimum energy value together with the state list and the **AppliedOperations** is returned as the result and the procedure stops.

Step 4:Determine the minimum energy and the state list minimizing the energy of graph G or a subgraph G_{ij}

- For a subgraph G_{ij}, we first determine the minimum energy of single vertex F_{Gij} reduced from G_{ij} and the corresponding state that minimizes the energy of F_{Gij}. Then the corresponding states of other vertices in G_{ij} are retrieved by scanning the temporary array backward from the end which contains reduction operations associated with G_{ij}. For each atomic operation, its state retrieval table is looked up and the states of the participating vertices can be determined in a retrospective way. The state list that minimizes the energy of G_{ij} is returned and the procedure goes back to Step 3.

- For a graph G, we first determine the minimum energy of the single vertex $\mathbf{F}_G$ reduced from G and the corresponding state s_{F_G} that minimizes the energy of $\mathbf{F}_G$. Then the corresponding states of all the other vertices are retrieved by scanning the array **AppliedOperations** from its end. For each reduction operation, its state retrieval table is looked up and the states of the participating vertices can be determined in a retrospective way. The state list that minimizes the energy of G is then returned and the procedure goes back to Step 3.

6.2.5 Example for Graph Reduction Algorithm

The following example illustrates the overall workflow of the proposed approach. Let us consider the graph $G(V, E)$ as shown in Figure 59, where V={1, 2, . . . , 11}, E={E_1, E_2,. . . , E_{18}} and $\underset{v\in V,\ s_v\in UL_v}{EV}(v, s_v) = 0$. The proposed algorithm works as follows:

In Step 1, we find that each vertex has a degree of at least 3 and there is no appropriate atomic reduction operation that can be applied. Therefore, the procedure goes to Step 2 where potentially reducible subgraphs are identified. The first subgraph identified is $G_{26}(V_{G26}, E_{G26})$, where V_{G26}={1, 3, 4, 5} and E_{G26}={E_2, E_3, E_4, E_7}. Before G_{26} can be further reduced, the energies of the vertices {1, 3, 4, 5} that connect G_{26} to the outside vertices 2 and 6, are updated according to operation IV. During the reduction of G_{26}, we first apply a

Type I operation to edges E_2 and E_3 which join at vertex 1. As the result, a new edge E_{19} connecting vertices 3 and 4 is inserted (Figure 59b), while E_2, E_3, and vertex 1 are removed from G_{26}. Then, we continue to apply Type I operation to G_{26} which reduces edges E_{19} and E_4 and vertex 3 to a new edge E_{20}, as shown in Figure 59c. Now two edges E_{20} and E_7 connecting the same pair of vertices 5 and 4; Therefore, a type II operation is used in the next step to reduce E_{20} and E_7 to a new edge E_{21} (Figure 59d). At this point, since there is only one edge E_{21} and its two ending vertices 4 and 5 left in G_{26}, a Type III operation can be applied to reduce vertex 5 and edge E_{21} to vertex 4 (Figure 59e). After G_{26} is reduced to a single vertex 4, the final step of this type IV operation is to insert a new edge E_{22} in place of this vertex, connecting vertices 2 and 6, as shown in Figure 59f. The current graph after the reduction of G_{26} as shown in Figure 59f is subject to further reductions in a recursive fashion.

The complete sequence of operations stored in array **AppliedOperations** is listed below:

AppliedOperations: (

1. **Type IV**(i=2, j=6, connecting_graph={1, 3, 4, 5}, new_edge=E_{22}, Oper_list: Opers_Type); **Oper_list**: (
   (a) **Type I**(e_1=E_2,e_2=E_3,joint_vertex=1, new_edge=E_{19});
   (b) **Type I**(e_1=E_{19},e_2=E_4,joint_vertex=3, new_edge=E_{20});
   (c) **Type II**(e_1=E_{20},e_2=E_7,new_edge=E_{21});
   (d) **Type III**(e=E_{21}, vertex_removed=5, vertex_remained=4));
2. **Type I**(e_1=E_{22}, e_2=E_6, joint_vertex=2, new_edge=E_{23});
3. **Type II**(e_1=E_{23}, e_2=E_{10}, new_edge=E_{24});
4. **Type IV**(i=6, j=11, connecting_graph={7, 8, 9, 10}, new_edge=E_{26}, Oper_list: Opers_Type); **Operation_list**: (
   (a) **Type I**(e_1=E_{11}, e_2=E_{13}, joint_vertex=7, new_edge=E_{25});
   (b) **Type III**(e=E_{25}, vertex_removed=8, vertex_remained=10));
   (c) **Type III**(e=E_{18}, vertex_removed=10, vertex_remained=11));
5. **Type III**(e=E_{26}, vertex_removed=6, vertex_remained =9)).

The state retrieval tables constructed after each operation are shown in Figure 61. Assume there are three states for each vertex. The first operation in **AppliedOperations** is a type IV operation for subgraph G_{26} (Figure 59). For each pair of states (s_2, s_6), we construct the state retrieval tables for the operation IV and associated atomic operations I and III. For example, Figure 61a, Figure 61b, and Figure 61c show state

retrieval tables after atomic Operations 1.1, 1.2, and 1.4, given s_2=1 and s_6=1. G_{26} is reduced to the single vertex 4 and its minimum energy is the same as that of vertex 4, which is assumed to be reached at s_4=1 in this example. To construct a state retrieval table for the operation IV, other vertices in G_{26}have to be determined in addition to s_2=1, s_4=1, and s_6=1. To do so, we start from the end of the **Operation_list** of the subgraph atomic operations and check the corresponding state retrieval tables backwards. First, from the state retrieval table in Figure 61c, we find that s_5=2 if s_4=1. Thus, (s_5=2) is recorded in the state retrieval table for the type IV operation (Figure 61d). Next, we look up the row where s_4=1 and s_5=2 is located in the state retrieval table in Figure 61b to identify the state of vertex 3 (s_3=1 in this example). Thus, (s_3=1) is recorded in the table in Figure 61d. Finally, we find the state of vertex 1 in the row where s_3=1 and s_4=1 is located in the state retrieval table in Figure 61a, and record the state of vertex 1 (s_1=1) is to the table in Figure 61d. The entry (s_2=1, s_6=1, s_1=1, s_3=1, s_4=1, s_5=2) in the state retrieval table of G_{26} indicates that given (s_2=1, s_6=1), the energy of G_{26} reaches its minimum when (s_1=1, s_3=1, s_4=1, s_5=2).

The state retrieval tables of Operations 2, 4, and 5 are shown in Figure 61e, Figure 61i, and Figure 61j. The state retrieval table of Operation 4 (Figure 61i) which is also a type IV reduction is constructed the same way as we did for that of Operation 1. Graph G is reduced to the single vertex 9 and its minimal energy G is the same as that of vertex 9, which is assumed to be reached at s_9=2. To determine the state of other vertices in G which minimize the energy of G, we start from the state retrieval table of Operation 5. s_6 is identified as 1 by querying the table in Figure 61j. From Figure 61i, we can derive that s_7=1, s_8=2, s_{10}=1,s_{11}=1 given (s_6=1, s_9=2), which is then be used to derive s_7=1 by Figure 61e. Finally, the row (s_2=1, s_6=1, s_1=1, s_3=1, s_4=1, s_5=2) in Figure 61d is located and the state list of vertices (s_1=1, s_2=1, s_3=1, s_4=1, s_5=2,s_6=1, s_7=1, s_8=2,s_9=2, s_{10}=1, s_{11}=1) is returned.

6.2.6 Complexity Analysis

Let $n = |V|$ and $m = |E|$. Denote the time complexity to reduce a graph (Steps 1 and 2 in Section 6.2.3) as $T(n, m)$. In Step 1, each vertex is checked to see if any atomic reduction operation (Type I/II/III operation) can be applied to it. After each atomic reduction operation, the graph shrinks by at least one vertex (operations I & III) or one edge (operation I & II). Therefore, with at most $(n + m)$ times of reduction operations, the given (reducible) graph can be reduced to one single vertex. In the worst situation, $(n + m)$ iterations are required to scan the vertices in the graph and each scan will check all the n vertices. Therefore, the time complexity for this step is at most $n \times (n + m)$.

For a Type IV operation, we observe that the simplest form of a subgraph G_{ij} contains at least two vertices and one edge, as shown in Figure Since the graph after Step 1 cannot be further reduced by any atomic operation, the degree of each vertex should be at least 3 and there are at least four edges that connect the subgraph to the outside vertices i and j, as shown in Figure Therefore, the reduced graph will shrink by

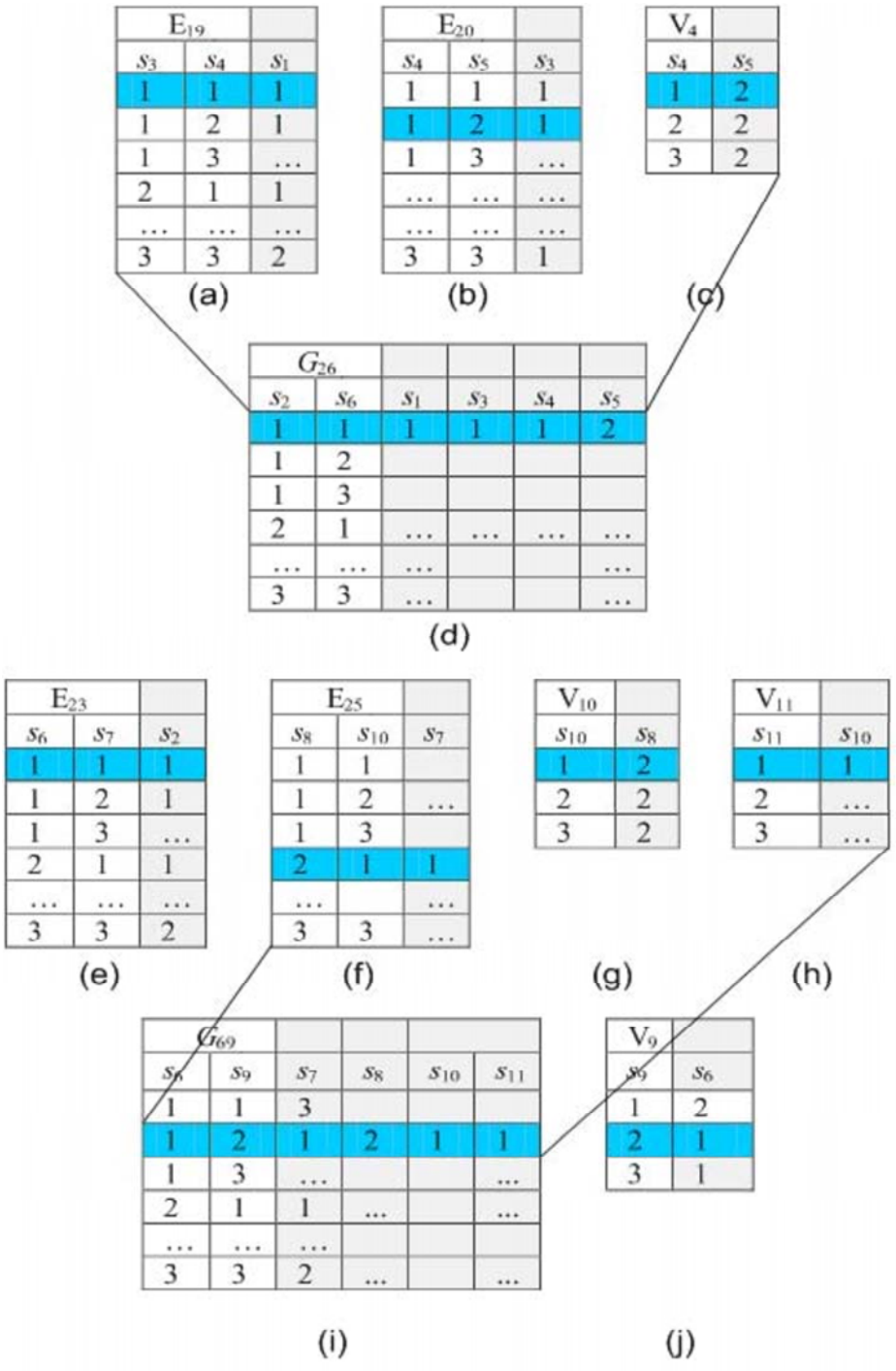


Figure 61: The state retrieval tables for the example in Figure .

at least two vertices and four edges. Assume that the reduced graph has n' vertices and m' edges, we have $n' \leq n$-2 and $m' \leq m$-4.

In Step 2, $C(n, 2)$ connected subgraphs may be processed in the worst situation. It will take at most $(n + m)$ steps to find such a G_{ij} because all the vertices and edges in the graph will be searched in the worst situation. Since each connecting subgraph G_{ij} has at most $(n$-2) vertices and $(m$-4) edges, it will take at most $(n$-2)$\times(n$-2+m-4) steps to reduce $G_{ij}$by the atomic operations. In total, it will take $n + m$+$(n$-2)$(n$-2+m-4) steps to process one connecting subgraph. In the worst situation, only one Type IV operation is performed after processing all subgraph G_{ij}. Therefore, the following relationship holds:

Since the time needed to reduce an empty graph is 0, $T(0, 0)$ is equal to 0. Based on the inequality in equation 11 , the upper bound of $T(n, m)$ is $O(\min(n/2, m/4)\times(n^3 \times m + n^4))$, which is a polynomial in n.

In Step 3 and Step 4, the minimal energy is determined and the state list which minimizes the energy is retrieved. Since each reduction operation introduces at most one new edge and there are at most $n + m$ reduction operations, at most $n + m$ new edges will be created. Taking the original edges into account, there are at most $n + m + m$ edges and n vertices involved in the energy calculation. Thus the minimal energy can be also determined in polynomial time. As a summary, the entire procedure, including graph reduction and the retrieval of the state list that minimizes the total energy, will take polynomial time if the given graph is reducible.

Figure 10. Simplest connected subgraph G_{ij} (the shaded area) with two vertices and one edge. The four dashed edges connect the subgraph through vertices i and j.

6.2.7 Cases Cannot Be Reduced

The proposed algorithm for graph reduction is not effective for all graphs. There are some graphs which cannot be reduced in polynomial time by this algorithm. An example is shown in Figure 62a. In this example, given any pair of vertices i and j, their connecting subgraph G_{ij} contains all the rest of the vertices. Figure 62b illustrates the connecting subgraph G_{ij} for vertices i and j which are shown as two black nodes in the Figure 62a. By applying the three atomic reduction operations, G_{ij} can only be reduced to Figure 62c, instead of a single vertex. Therefore, G_{ij} is irreducible. After checking the connecting subgraph G_{ij} for each pair of vertices i and j in Figure 62a, we find that none of them is reducible. Therefore, the graph shown in Figure 62a is not reducible. Consequently, any graph that contains such an irreducible subgraph is also polynomial irreducible.

6.3 Experiment for Graph Reduction Method

The proposed algorithm has been applied to refining building topology from airborne LIDAR measurements to examine the effectiveness. The LIDAR technology provide an effective way to derive 2D footprints

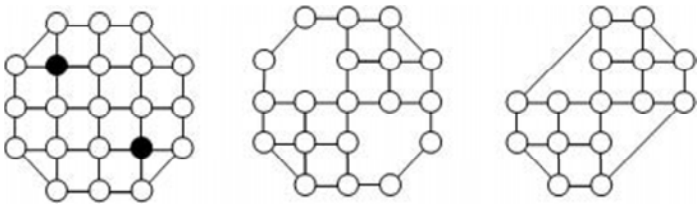


Figure 62: (a) An example of a graph that cannot be reduced to a single vertex by the three atomic reduction operations, (b) The connecting subgraph for the two black nodes shown in (a). (c) The reduction result after applying the three atomic reduction operations to (b).

and 3D shapes of buildings by measuring building elevation directly [35]. The test data site is located at the campus of Florida International University (FIU) at Miami, Florida, covering 6 km^2 of low relief topography. The LIDAR data for building extraction with an average point spacing about 1 m were collected in August 2003. The buildings in the test site include residential houses, commercial buildings, and institutional buildings.

A building topology is represented by a set of connected roof plane surfaces (polygons) projected onto a 2D space, which matches our proposed "2D snake" very well. Initial footprints and internal topology of buildings are extracted from LIDAR measurements through a plane-fitting technique and regional growing algorithm [35]. The boundaries (edges) between different roof planes are noisy and critical corner vertices might not be in correct positions in initial footprints due to the influence of irregularly spaced point LIDAR measurements. The process for refining building topology involves adjusting the initial topology by changing the admissible states of vertices through minimizing a defined energy function. The allowable states of vertices are determined by the spatial resolution and errors of LIDAR measurements. Therefore, the refinement of building topology from LIDAR measurements provides an ideal case to test the proposed algorithm.

The energy function for building topology refinement can be defined in any format as long as it satisfies the search constraints in Section 6.2.1. The energy function for this test depends upon the length of an edge and the angle between the edge and the dominated building direction. A detailed discussion about the energy function is beyond the scope of this chapter since the purpose of the experiment is to investigate the effectiveness of graph reduction algorithm. The focus of the experiment is to examine whether a connected building topology can be reduced by the algorithm. The percentage of building topologies that are successfully reduced by the algorithm is used to measure the effectiveness of the algorithm. A successful reduction of a building topology means that the raw topology of that building can be reduced into a single vertex using the proposed algorithm.

One data set including 67 institutional buildings on the FIU campus and one data set including 211 residential and commercial buildings next to the FIU campus have been used to test the algorithm. Figure 63

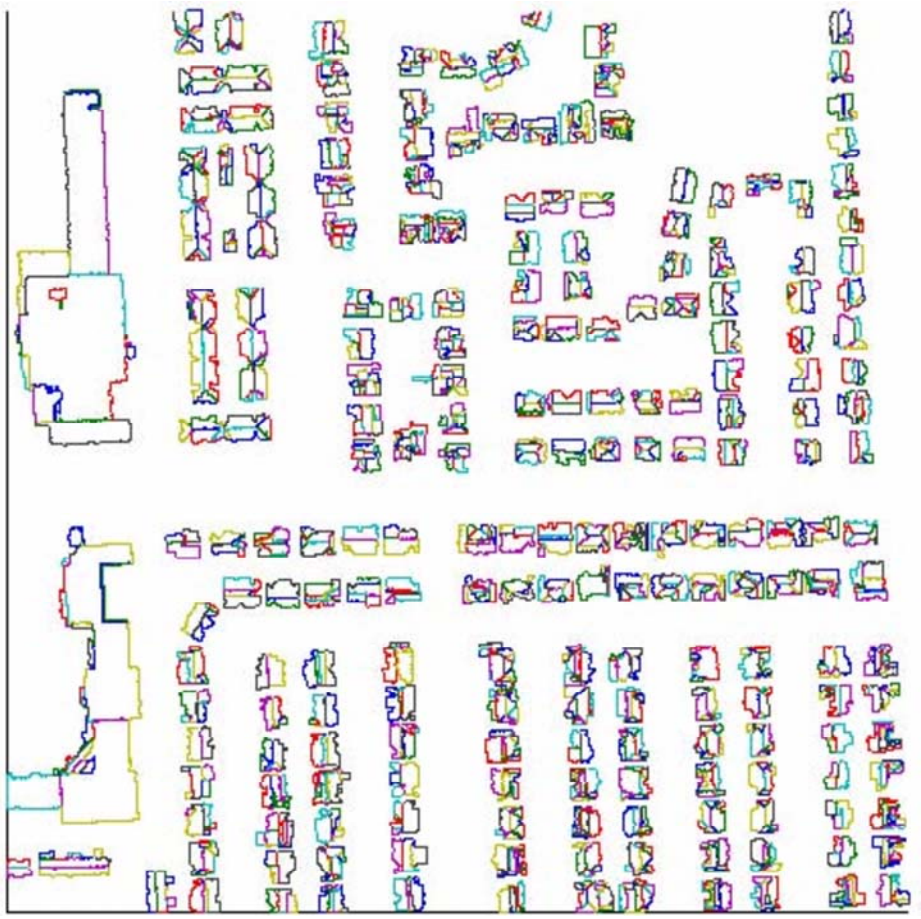


Figure 63: The raw topologies and footprints for residential and commercial buildings next to the FIU campus.

displays the raw footprints and topologies for buildings derived from LIDAR measurements for the area next to the FIU campus.

Test results indicate that 210 buildings from the data set next to the FIU campus and 66 buildings from the data set at the FIU campus can be successfully reduced by the proposed algorithm. The reduction rate is over 98% for both cases. It appears that the algorithm is capable of reducing the topology of a building with a complicated shape. For example, the topology of a building on the FIU campus with total 46 vertices and 67 edges (Figure 64a) is successfully refined (Figure 64b) through the graph reduction algorithm. In addition, connected subgraphs for almost all buildings (209 of 210) in graph reduction process are identified by only searching direct neighbors of vertex pairs using the algorithm in Section 6.2.3. This expedites the process of graph reduction. It took about 2 and 3 minutes for a PC with a 2.8 GHz processor and 2 GB RAM to complete the entire reduction process for the dataset for the FIU campus and the dataset next to the FIU campus, respectively.

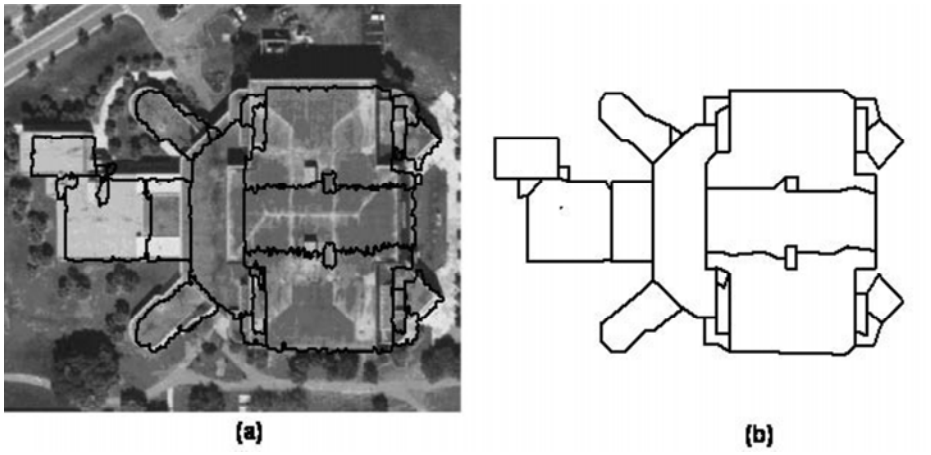


Figure 64: The raw (a) and refined (b) topology of a complicated building at the FIU campus. The roof surface edges from LIDAR measurements (dark line) do not seamlessly match those in the aerial photo because the aerial photo is not orthorectified perfectly.

6.4 NP-Completeness of the General 2D Snake Problem

We have demonstrated that a subset of 2D snake problems can be resolved in polynomial time. However, not all 2D snake problems can be solved in polynomial time. In fact, the general 2D snake problem presented is NP-complete. To prove this conclusion, we first prove that a simple case, "3-state 2D snake" of 2D snake problems is NP-complete. Some of vertices in the general 2D snake problem have more than three states, but all vertices in the 3-state 2D snake have at most three states. Since it is more complicated than the 3-state 2D snake problem, the general 2D snake problem should be also NP-Complete.

We start with the following standard NP-complete problem.

3SAT:

An instance of 3SAT contains a set U of variables, a collection C of clauses over U in which $|C| \leq 3$ for each $c \in C$. The variable-clause graph of the given instance is a graph $G = (V, E)$, where $V = U \cup C$ and E contains exactly those pairs $\{u, c\}$ such that either u or its negation belongs to the clause c.

Question: Is there a satisfied truth assignment for C?

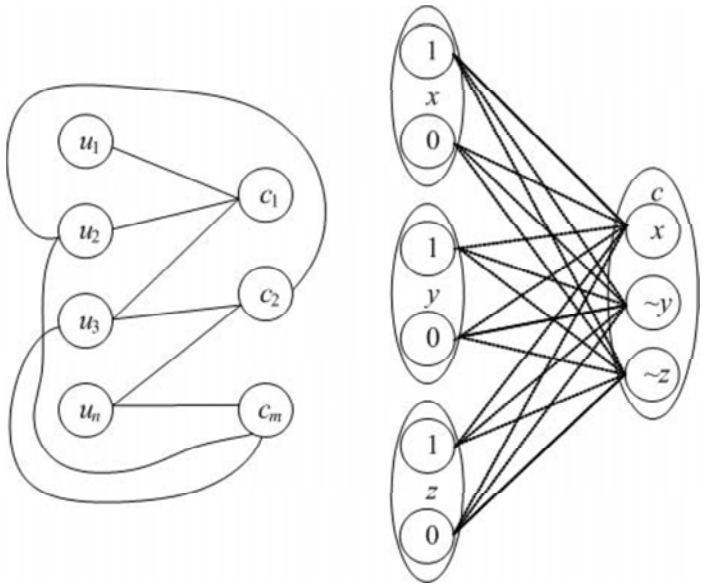
3-State 2D Snake:

An instance of 3-state 2D snake consists of a weighted graph $G=(V, E)$ and an integer k. Each vertex has at most three possible states. The energy EE on each edge e is determined by the states of its two endpoints. Define$EG(G) = \sum EE$.

Question: Is there a state list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}), s_k \in UL_k\}$ which makes $EG(G) \leq k$?

Given an instance of a 3SAT problem, it can be transformed to an instance of the 3-state 2D snake problem as shown in Figure 65.

Figure 65a shows the graph $G=(V, E)$ that corresponds to a 3SAT instance. The corresponding 3-state



(a) (b)

Figure 65: The procedure to transform a 3SAT instance to an instance of the 3-state 2D snake problem is illustrated with an example clause $\{x, \sim y, \sim z\}$. (a) The graph corresponding to the instance of a 3SAT problem. $u_1 \ldots, u_n$ and $c_1 \ldots, c_m$ correspond to the variable vertices and the clause vertices, respectively. (b) The energy value of the edge connecting each clause vertex and each literal vertex (related to that clause). Each clause vertex has at most three states and each variable vertex has two states. The edge energy value is 0 for each dash line and -1 for each solid line.

snake instance has the same structure as G. The two states assigned to each variable u correspond to the truth assignments 1 and 0, respectively. The states assigned to each clause c correspond to the literals in that clause. The energy value of the edge $e = (u, c)$ is determined by the states of its two endpoints. Its energy value is set to -1 if the current state of c is u and the current state u is '1', or the state of c is $\sim u$ and the state of u is '0'. Otherwise, the energy value is set to 0. It can be formally described by Equation (47).

$$EE(e = (u, c), s_u, s_c) = \begin{cases} -1 & s_c = u \& s_u = 1 \\ -1 & s_c = \sim u \& s_u = 0 \\ 0 & otherwise \end{cases} \tag{47}$$

Figure 65b shows a collection C with only one clause c:$\{x, \sim y, \sim z\}$, and the energy values assigned to its edges. The energy value of the edge (x, c) is set to -1 when the state of c is x and the state of x is 1. Otherwise, it is set to 0. Similarly, the energy value of the edge (y, c) is set to -1 when the state of c is $\sim y$ and the state of y is 0. Otherwise, the energy of the edge is 0. Since the clause c can be in only one state at a time, there is at most one edge of c which has an energy value of -1. Therefore, the sum of the edge energy values $\sum EE$ will be at least $-|C|$.

We can see that a 3SAT instance is satisfiable if and only if the sum of the edge energy values of its corresponding 3-state 2D snake instance equals $|C|$. If the 3SAT instance is satisfiable, there is a truth assignment

making each clause true. The state of each variable u is then set to its truth assignment. The state of each clause c is set to one of its literals which make it true. By such assignments, we can see that there is one and only one edge of each clause c with an energy value of -1. Thus, the sum of the edge energy values $\sum EE$ will be -|C| under such assignments.

If the sum of the edge energy values $\sum EE$ of a 3-state 2D snake instance equals -|C|, each clause c will have one and only one edge (u, c) with an energy value -1. In the corresponding 3SAT instance, u is set to 1 if u is in c or 0 if $\sim u$ is in c. Obviously, such truth assignments will make the corresponding 3SAT instance also true. Therefore, a 3SAT instance is satisfiable if its corresponding 3-state 2D snake instance is satisfiable. Since the 3-state 2D snake problem is easily shown to be in NP, we conclude that the 3-state 2D snake problem is an NP-complete problem. Furthermore, we can conclude that the general 2D snake problem is also NP-complete for the aforementioned reason. In fact, planar 2D snakes, one of the subsets of the general 2D snake problem, can be proven to be NP-complete problem also in the same way by reducing from planar 3SAT to it.

6.5 Conclusion

Discrete dynamic programming has been widely used to resolve snake problems. It can bypass local minima and guarantee global optimality of the solution. However, its application is limited to snake problems with 1D topology and cannot handle 2D snake problems. In this research, we have demonstrated that a subset of 2D snake problems can be resolved in polynomial time. The topology of such a 2D snake problem can be reduced to a single vertex by applying the set of proposed graph reduction operations. The proposed algorithm was applied to refining 2D building topology extracted from airborne LIDAR data. Various topologies for institutional, commercial, and residential buildings have been used for refinement. Over 90% of building topologies have been successfully reduced and their global optima have been found in polynomial time. This demonstrates that the proposed method is very effective for refining building topology although the algorithm only work for a subset of graph.

We also showed that the proposed algorithm cannot resolve 2D snake problems with very complicated topologies. We have proven that the reduction of general 2D snakes is an NP-complete problem by analyzing time complexity. For those irreducible 2D snake problems, the domain-specific knowledge could be used to derive the approximate optimal results. For example, we can remove the least important edges in a 2D snake gradually until the remained graph is reducible. The minimum energy of the remaining reducible graph plus the energies of those removed edges should be close to the minimum energy of the original problem. This remains as our most immediate goal for future work.

CHAPTER 7
CONCLUSION AND FUTURE WORK

In this dissertation, several approaches are proposed to filter out major feature objects such as terrain, building and footprint from LIDAR data. A progressive morphological filter was built to remove non-ground LIDAR measurements to generate bare-ground elevation models. Experimental results show that the proposed progressive morphological filter separated ground and non-ground LIDAR measurements in both the urban and mountain areas accurately and effectively by gradually increasing the sizes of the opening operation and elevation difference threshold. An accuracy analysis of filtering results for the urban data set shows that only 3% errors were committed by the filter in a random sample of 648 measurements. The selections of the filtering parameters have a great impact on the removal of non-ground objects. Appropriate parameters can be found based on analyzing terrain and non-terrain measurements in the study area. The filtering process is highly automatic and requires little human interference, which is desirable when processing voluminous LIDAR measurements. However, this method is not perfect, and a few commission and omission errors did occur during filtering. A human interactive filtering method may need to be developed in the future to refine the filtering results.

A region growing algorithm based on a local plane-fitting technique is developed to segment building measurements from non-ground measurements. Experiments demonstrated that the method is not sensitive to seed point selection, indicating that it is robust for identifying building measurements. A quantitative accuracy analysis for identified building measurements was performed using count and area-based metric methods. The result shows that all the buildings are correctly identified and 12% area errors are committed by the area growing methods. However, some small facets on the building roofs are lost and some trees are misidentified as building measurements, which has great negative effects on the subsequent operations, especially footprint extraction and complicated building model construction.

With the identified building measurements, simple building models are derived based on several methods. First, the Douglas-Peucker algorithm is utilized to remove noises on the boundary and generate a coarse footprint. Second, a weight-based method is developed to estimate the domain directions of each building. Finally, the existing assumptions for building footprint are extended to allow the existence of line segments

oblique to the domain directions. Several operations based on these assumptions are proposed to adjust the footprint, which forms the simple building model with the average height of the building. The method for domain direction estimation is proven to be robust and applicable for footprints if the sum of the lengths from line segments paralleling to either domain direction amounts to more than half of the perimeter of the footprint. The footprint adjusting algorithm makes the footprint look more realistic while almost keeping their original position information. Experiments proved that these methods achieve good result in most situations. However, the final footprints are still not perfect. In some situations, a footprint may contain self-intersection segments or over-adjusted segments deviating far from the original boundary positions.

A sophisticated building model can also be reconstructed automatically. A simple and concise method is proposed to extract the raw 2D topology of each building first. Then the raw 2D topology is adjusted and a 3D building model can be reconstructed based on the adjusted 2D topology and roof plane information. The key factor for the 3D building model reconstruction is to adjust the 2D topology. In our framework, the snake algorithm is utilized and the 2D topology with minimal energy will be returned as the targeted 2D topology. We defined two new energy functions for the 2D topology adjustment and the experiment results prove that they work well. Since the traditional snake algorithm is limited to 1D topology like contour, We also extended the 1D snake algorithm to resolve 2D snake problems based on graph operations. Experimental result prove that most (over 90%) of the buildings can be reconstructed based on the 2D snake algorithm.

The framework can be further improved in many aspects and they are listed as follows:

- The extracted feature objects such as terrain and buildings are stored in file system which is not convenient for tasks such as LIDAR data retrieving, analyzing and visualization. Most commercial GIS tools such as ArcGIS cannot process LIDAR data efficiently. For example, they cannot display sophisticated models efficiently due to the compound geometric structures. The response for the LIDAR data processing is very slow and cannot be tolerated by users. Normally it will take ArcGIS about thirty seconds on an advanced desktop computer to load and visualize the LIDAR dataset of a small area like the FIU main campus. Thus, a database system should be developed to support fast retrieval and visualization of LIDAR data and the filtered features. Some issues will be studied and resolved. For example, what are the querying patterns for LIDAR data and how to partition, index and organize the huge volume of LIDAR data efficiently? Similar projects in other fields are reported. [3] introduces a database system named “Sloan Digital Sky Survey” for astronomy. [33] presents a geospatial database system integrating images and different kinds of geometrical objects together. The techniques used in these projects may be useful for LIDAR data processing. However, LIDAR data measurements are irregularly spaced and much denser than the data sources used in these projects, which makes it more

difficult to build a database system for LIDAR data processing directly based on these techniques. New partition and index methods suitable for LIDAR data should be developed later.

- The area growing result to segment building roof planes is critical to the performance of the whole framework. The segmentation result is not robust enough in some cases and needs to be further improved in the future.

The long-term objective of the system is to automatically filter out other features such as trees, lakes, roads, etc. These features can be integrated together and provide an environment for real-time and automatic visualization of the real world.

[1] R. C. J. A. A. Amini, T. E. Weymouth. using dynamic programming for solving variational problem in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:855–867, 1990.

[2] J. S. B. A. F. Elaksher. Reconstructing 3d buildings from lidar data. *ISPRS Commission III Symposium, Photogrammetric and Computer Vision*, 102-107, 2002.

[3] A. T. J. G. D. S. R. J. B. Alexander S. Szalay, Peter Kunszt. Designing and mining multi-terabyte astronomy archives: The sloan digital sky survey. *SIGMOD*, 29:451–462, 2000.

[4] A. Alharthy and J. Bethel. Heuristic filtering and 3d feature extraction from lidar data. *ISPRS Commission III Symposium, Photogrammetric and Computer Vision*, pages 29–34, 2002.

[5] K. S. S. N. C. K. S. S. N. C. Atsuyuki Okabe, Barry Boots. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. Chichester: John Wiley & Sons, 1992.

[6] P. Axelsson. Dem generation from laser scanner data using adaptive tin models. *International Archives of Phtogrammetry and Remote Sensing*, 33:85–92, 2000.

[7] C. Brenner. Towards fully automatic generation of city models. *ISPRS*, XXXIII, 2000.

[8] C. Brenner. Modeling 3-d objects using weak csg primitives. *ISPRS*, 35, 2004.

[9] V. Cerny. A thermodynamical approach to traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[10] C. I. Cohen, L.D. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Trans. on Pattern Analysis and Machine Learning*, 15:1131–1147, 1993.

[11] R. G. Congalton. A review of assessing the accuracy of classifications of remotely sensed data. *emote Sensing of Environment*, 37:35–46, 1991.

[12] L. A. C. D. Geiger, A. Gupta and J. Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:294–302, 1995.

[13] A. W. D. Terzopoulos and M. Kass. Constraints on the deformable models: recovering 3d shape and nonrigrid motions. *Artificial Intelligence*, 36:91–123, 1988.

[14] D. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10:112–122, 1973.

[15] W. Eckstein and O. Munkelt. Extracting objects from digital terrain models. *Proc. Int. Society for Optical Engineering: Remote Sensing and Reconstruction for Three-Dimensional Objects and Scenes, Proceedings of the International Society for Optical Engineering*, 1995.

[16] S. O. Elberink and H. G. Maas. The use of anisotropic height texture measures for the segmentation of airborne laser scanner data. *International Archive of Photogrammetry Remote Sensing*, XXXIII:678–684, 2000.

[17] F. S. Ellen Schwalbe, Hans-Gerd Maas. 3-d building model generation from airborne laser scanner data using 2-d gis data and orthogonal point cloud projections. *ISPRS Workshop "Laser scanning 2005"*, 2005.

[18] S. Filin. Surface clustering from airborne laser scanning data. *ISPRS Commission III Symposium, Photogrammetric and Computer Vision*, pages 119–124, 2002.

[19] A. Fischer. Extracting buildings from aerial images using hierarchical aggregation in 2d and 3d. *Computer vision and image understanding*, 72:185–203, 1998.

[20] J. P. T. G. Subsol and N. Ayache. A scheme for automatically building three-dimensional morphometric anatomical atlases: application to skull atlas. *Medical Image Analysis*, 2:37–60, 1998.

[21] A. Grn and X. Wang. Automatic extraction of man-made objects from aerial and space images. pages 93–101, 2001.

[22] W. X. Grun, A. Cc modeler: 'a topology generator for 3-d city models. *ISPRS Commission IV Symposium on "GIS - Between vision and application"*, 32:188–196, 1998.

[23] H. M. M. H. Ruther and E. G. Mtalo. pplication of snakes and dynamic programming optimization technique in modeling of buildings in informal settlement areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 56:269–282, 2002.

[24] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. MA: Addison-Wesley Publishing Company, 1992.

[25] R. A. Haugerud and D. J. Harding. Some algorithms for virtual deforestation (vdf) of lidar topographic survey data. *nternational Archives of Photogrammetry and Remote Sensing*, 34:211–218, 2001.

[26] B. K. P. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[27] N. H. J. Kilian and M. Englich. Capture and evaluation of airborne laser scanner data. *International Archives of Phtogrammetry and Remote Sensing*, 31:383–388, 1996.

[28] S. M. J. Marroquin and T. Poggio. Probabilistic solution of iii-posed problems in computational vision. *Journal of American Statistical Association*, 82:76–89, 1987.

[29] H. D. J. Montagnat and N. Ayache. A review of deformable surfaces: topology, geometry, and deformation. *Image and Vision Computing*, 19:1023–1040, 2001.

[30] E. K. J. Overby, L. Bodum and P. M. Ilse. utomatic 3d building reconstruction from airborne laser scanning and cadastral data using hough transform. *Proceeding of the ISPRS 20th congress*.

[31] J. R. Jensen. *Remote Sensing of the Environment*. Upper Saddle River, NJ:Prentice Hall, 2001.

[32] Z. Z. S. C. Jianuan Yan, Keqi Zhang and G. Narasimhan. A graph reduction method for 2d snake problems. *CVPR*, to be submitted, 2007.

[33] N. K. K. t. B. N. J. B. N. H. K. R. R. L. C. E. J. K. S. G. J. L. D. D. Jignesh Patel, JieBing Yu and Naughton. Building a scalable geo-spatial dbms: Technology, implementation, and evaluation. *Proceedings of the 1997 SIGMOD conference*, pages 336–347, 1997.

[34] D. W. M.-L. S. J. Y. K. Zhang, S-C. Chen and C. Zhang. A progressive morphological filter for removing non-ground measurements from airborne lidar data. *IEEE Transactions on Geoscience and Remote Sensing*, 41:872–882, 2003.

[35] S.-C. C. Keqi Zhang, Jianhua Yan. Automatic construction of simple building models from airborne lidar data. *EEE Transactions on Geoscience and Remote Sensing*, 44:2523–2533, 2006.

[36] K. Kraus and N. Pfeifer. A new method for surface reconstruction from laser scanner data. *International Archives of Phtogrammetry and Remote Sensing*, 32:80–86, 1997.

[37] K. Kraus and N. Pfeifer. Determination of terrain models in wood areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry & Remote Sensing*, 53:193–203, 1998.

[38] L. J. Latecki and R. Lakamper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73:441–454, 1999.

[39] C. A. Lin and R. Nevatia. uilding detection and description from a single intensity image. *Computer Vision and Image Understanding*, 72:101–121, 1998.

[40] A. W. M. Kass and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[41] H. G. Maas. Fast determination of parametric house models from dense airborne laser scanner data. *ISPRS workshop on mobile mapping technology*, 32, 1999.

[42] M. Morgan and A. Habib. nterpolation of lidar data and automatic building extraction. *ACSM-ASPRS 2002 Annual Conference Proceedings*, 2002.

[43] M. Morgan and K. Tempfli. utomatic building extraction from airborne laser scanning data. *Proceeding of the 19th ISPRS Congress*, pages 616–623, 2000.

[44] P. S. N. Pfeifer and B. C. Derivation of digital terrain models in the scop++ environment. *roc. OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models*, 2001.

[45] S. Noronha and R. Nevatia. Detection and modeling of buildings from multiple aerial images. *IEEE Transactions on Pattern Analysis and Machine Learning*, 23:501–518, 2001.

[46] A. K. P. Lohmann and M. Schaeffer. Approaches to the filtering of laser scanner data. *International Archives of Photogrammetry and Remote Sensing*, 33:540–547, 2000.

[47] S. R. S. R. M. Haralick and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:523–550, 1987.

[48] M. Roggero. Airborne laser scanning: clustering in raw data. *International Archives of Photogrammetry and Remote Sensing*, 34:227–232, 2001.

[49] F. Rottensteiner and J. Jansa. Automatic extraction of buildings from lidar data and aerial images. *International Archives of Photogrammetry and Remote Sensing*, XXXIV:569–574, 2002.

[50] F. Rottensteiner and J. Jansa. utomatic extraction of buildings from lidar data and aerial images. *ISPRS Commission III Symposium, Photogrammetric and Computer Vision*, pages 119–124, 2002.

[51] T. P. C. S. P. Kaluzny, S. C. Vega and A. A. Shelly. *S+SpatialStats: User's Manual for Windows and UNIX*. New York: Springer-Verlag, 1997.

[52] A. Sampath and J. Shan. Urban modeling based on segmentation and regularization of airborne lidar point clouds. *Proceeding of the ISPRS 20th congress*, commission III, 2004.

[53] J. A. Shufelt and D. M. Mckeown. Fusion of monocular cues to detect man-made structures in aerial imagery. *Computer Vision Graphics and Image Understanding*, 57:307–330, 1993.

[54] G. Sithole. Filtering of laser altimetry data using a slope adaptive filter. *International Archives of Photogrammetry and Remote Sensing*, 34:203–210, 2001.

[55] B. C. T. Schenk. Fusion of lidar data and aerial imagery for a more complete surface description. *International Archives of Photogrammetry and Remote Sensing*, 34(3A):310–317, 2002.

[56] J. P. Thirion. Image matching as a diffusion process: an analogy with maxwell's demons. *Medical Image Analysis*, 2:243–260, 1998.

[57] G. S. V. Caselles, R. Kimmel and C. Sbert. Minimal surfaces based object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:394–398, 1997.

[58] M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recognition Letters*, 13:517–521, 1992.

[59] G. Vosselman. Slope based filtering of laser altimetry data. *International Archives of Phtogrammetry and Remote Sensing*, 33:958–964, 2000.

[60] G. Vosselman. Building reconstruction using planar faces in very high density height data. *ISPRS*, XXXIV:211–218, 2001.

[61] R. L. S. W. E. Carter and K. C. Slatton. Photon counting airborne laser swath mapping (pc-alsm. *SPIE Proceddings: Remote Sensing Applications of the Global Position System*, 2004.

[62] H. Wackernagel. *Multivariate Geostatistics*. Berlin:Spinger-Verlag, 1998.

[63] U. Weidner and W. Forstner. Towards automatic building reconstruction from high resolution digital elevation models. *ISPRS Journal of Photogrammetry Remote Sensing*, 50:38–49, 1995.

[64] O. W. Y. Boykov and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1222–1239, 2001.