Automatic 3D Building Detection and Modeling from Airborne LiDAR Point Clouds

by

Shaohui Sun

B.S. Sun Yat-sen University, China, 2006

M.S. Sun Yat-sen University, China, 2008

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

December 4, 2013

Signature of the Author ______________________________

Accepted by ______________________________
Coordinator, Ph.D. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

Ph.D. DEGREE DISSERTATION

The Ph.D. Degree Dissertation of Shaohui Sun
has been examined and approved by the
dissertation committee as satisfactory for the
dissertation required for the
Ph.D. degree in Imaging Science

Dr. Carl Salvaggio, Dissertation Advisor

Dr. Anthony Harkin

Dr. Nathan Cahill

Dr. Jinwei Gu

Date

DISSERTATION RELEASE PERMISSION

ROCHESTER INSTITUTE OF TECHNOLOGY

COLLEGE OF SCIENCE

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

Title of Dissertation:

Automatic 3D Building Detection and Modeling from Airborne LiDAR Point Clouds

Automatic 3D Building Detection and Modeling from Airborne LiDAR Point Clouds

by

Shaohui Sun

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

Urban reconstruction, with an emphasis on man-made structure modeling, is an active research area with broad impact on several potential applications. Urban reconstruction combines photogrammetry, remote sensing, computer vision, and computer graphics. Even though there is a huge volume of work that has been done, many problems still remain unsolved. Automation is one of the key focus areas in this research. In this work, a fast, completely automated method to create 3D watertight building models from airborne LiDAR (Light Detection and Ranging) point clouds is presented. The developed method analyzes the scene content and produces multi-layer rooftops, with complex rigorous boundaries and vertical walls, that connect rooftops to the ground. The graph cuts algorithm is used to separate vegetative elements from the rest of the scene content, which is based on the local analysis about the properties of the local implicit surface patch. The ground terrain and building rooftop footprints are then extracted, utilizing the developed strategy, a two-step hierarchical Euclidean clustering. The method presented here adopts a "divide-and-conquer" scheme. Once the building footprints are segmented from the terrain and vegetative areas, the whole scene is divided into individual pendent processing units which represent potential points on the rooftop. For each individual building region, significant features on the rooftop are further detected using a specifically designed region-growing algorithm with surface smoothness constraints. The principal orientation of each building rooftop feature is calculated using a minimum bounding box fitting technique, and is used to guide the refinement of shapes and boundaries of the rooftop parts. Boundaries for all of these features are refined for the purpose of producing strict description. Once the description of the rooftops is achieved, polygonal mesh models are generated by creating surface patches with outlines defined by detected vertices to produce triangulated mesh models. These triangulated mesh models are suitable for many applications, such as 3D mapping, urban planning and augmented reality.

Acknowledgements

Pursuing Ph.D. at R.I.T has become one of the most important life experiences of mine. I enjoyed every single moment here. I am so proud of being a member of R.I.T alumni community.

This thesis would not have been possible without the help, support, and advice from my advisor, Dr. Carl Salvaggio. Dr. Salvaggio has unsurpassed knowledge of a variety of topics in imaging science and grand vision that has been helping my research stay on the right track. Apart from being a great academic advisor, Dr. Salvaggio is also a wonderful career and life advisor. I also thank him for being such a trustworthy friend.

I wish to thank Dr. Nate Cahill for helping me conduct research on problems of computer vision during our short-term collaboration in my first year at R.I.T. Dr. Cahill taught me not only the knowledge of advanced image analysis but also the time management skill as a researcher.

I am extremely grateful to my other research committee members: Dr. Tony Harkin and Dr. Jinwei Gu for providing insightful advice without which I would not have been able to make any improvement.

I am also grateful to all the professors from whom I took classes. The knowledge I have gained through their wonderful teaching has been helping me significantly in my new career.

I would like to acknowledge the financial and academic support of R.I.T, Center for Imaging Science and the DIRS group throughout my study here. Thanks to Sue Chan for being incredibly patient and professional when I have all kinds of questions.

It has been a great honor to work with fellows in the DIRS group. I feel like being a member of a big family here. There is a huge amount of research data available that I can have easy access to anytime. Special thanks to Chris De Angelis for providing me with DIRSIG simulated data. Thanks to Cindy Schultz for always keeping everything well organized. Thanks to Mike Richardson for helping me understand the important paperwork when I was about to get the job. Thanks to my fellow officemates, Mike Harris, David Nilosek, and Katie Salvaggio. With those casual conversations related to study, research and life, I received a lot of joy and happiness, which helped me a lot on adjusting myself under great pressure.

I also wish to thank many classmates of Class 2010 with whom I shared office rooms on the second floor in my first year, including Bikash Basnet, Javier Concha, Aly Artusio-Glimpse, Shagan Sah, Siyu Zhu, Jiashu Zhang, Viraj Adduru and Poon Kanburapa. We helped each other on homework, projects. It has been a huge pleasure to study with all of them.

Bin Chen, Javier Concha, Kelly Canham and I had a great time at IGARSS 2012 in Munich, Germany. They were all there supporting me when I was presenting my research to researchers from places all over the world. The smile of Dr. John Kerekes who was the session chair at IGARSS 2012 helped me calm down before the speech, put myself together and embraced the challenge.

I would like to thank the lovely family from Irondequoit. Michele and Charlie are like parents to me. Tom and Felice are like brother and sister to me. I enjoyed every holiday we spend together. They have made me feel like I always have a home in Rochester.

Thanks to Dr. Lingfei Meng for sharing his own experience of how he managed to survive through the final preparation for his defense. Thanks to A.J. Altman, the founder of Hover, Inc, for being very supportive on my school work and providing partial tuition assistance. Thanks to Ravali Kankanala for being my dearest and closest friend and supporting me unconditionally.

In the end, I thank everyone that directly or indirectly helped me for the past a few years. I would never forget you wherever I go.

To my mom, dad, sister in China.

Thanks for always having great faith in me, even when I go through incredibly tough time.

Contents

List of Figures

Chapter 1

Introduction

Three dimensional building reconstruction has been a highly active research domain for decades.Various applications such as urban planning, virtual tourism, computer gaming, emergency response and robot navigation have been having increasing demands on 3D urban models. Popular commercial products like Google Earth and Apple Flyover have already deployed 3D building modeling techniques as a vital component. Those realistic models are usually made by texture mapping both aerial and ground-level imagery onto 3D geometric models. Traditionally, models are created manually. There are some very capable tools like Trimble SketchUp that allow the layperson to accomplish this, however, it does require a huge amount of human effort. It remains a very challenging and arduous task, especially when a large cityscape needs to be modeled.

In the remote sensing community, several types of data sources are suitable as inputs for the urban reconstruction task. Optical imagery is the conventional and most available data source. Research on extracting 3D information from ground or aerial imagery has been conducted for years. Recent advances in sensors have enabled techniques to directly capture 3D data over large scale areas. With the emergence of LiDAR (Light Detection and Ranging) technology, the form of point clouds becomes a powerful 3D representation, and can been created to improve the generation of 3D scenes in a more efficient and cost effective fashions.

In the aerial data of an urban setting, the three scene components that dominate this data are the building rooftops, vegetation, and the background terrain. The detection and extraction of these rooftops from the terrain is hence a crucial step in building reconstruction. Often times, the outlines of these building rooftops are quite complicated making the building modeling process a very challenging task.

In this research, the key contribution is an automatic workflow that exploits useful information from airborne LiDAR point clouds, effectively and robustly conducting the task of scene classification and 3D geometric model construction. The workflow detects building footprints and generates simplified, 3D models using solely LiDAR data from a large urban scene. One of the major challenges throughout this research is how to efficiently and accurately separate building regions from the rest of the background regions in the scene, particularly vegetation, without the assistance of multispectral or hyperspectral optical imagery. A graph cuts optimization scheme is used for vegetation detection and removal. In addition, due to the purposeful omission of a generic rooftop "template", an effective region growing based method for the extraction of major rooftop features is adopted. The algorithm presented for describing rooftops maintains the geometric integrity of the variety of shapes that exist in an urban landscape while utilizing certain regularities in the rooftop geometry usually exhibited by man-made structures. Rigorous quality evaluation of the developed approach is also conducted.

In Chapter 2, some basic terms that are used throughout this document are briefly described. The difference between photogrammetry and LiDAR will be discussed. High level introductions of specific algorithms that support this research will be presented as well. Related previous works will be examined, and some major limitations of these works will be targeted, and the main objective of this research will be addressed. In Chapter 3, the developed workflow is described in detail. Chapter 4 shows the results of this workflow when using the real data. A quantitative analysis of the result using simulated data is discussed in Chapter 5. Chapter 6 draws conclusions, addresses the limitations of this workflow, and the potential future work.

Chapter 2

Background

2.1 DEM, DSM, and DTM

The usage of the terms Digital Elevation Model (DEM), Digital Terrain Model (DTM) and Digital Surface Model (DSM) is not strictly defined in the photogrammetry and remote sensing literature. In most cases, the term Digital Surface Model (DSM) represents the earth's surface and all the other objects on it. In contrast, the Digital Terrain Model (DTM) only describes the bare ground surface without any objects such as trees and man-made objects (Fig.2.1).

The Digital Elevation Model (DEM) is often referred to as a generic term for DSM and DTM. It represents the height information without further specification on the earth surface. A DEM could be acquired by techniques such as stereo photogrammetry, LiDAR, IfSAR, and may also be built by land surveying, etc. [1]. DEMs are widely used in remote sensing community for many geographical or geological application purposes. To a certain extent, 3D urban reconstruction could be considered as a subset problem of DEM production.

2.2 Camera Model

Image-based stereo or multi-view systems have reached a relatively mature state in recent times. Sometimes accurate camera information is available, which allows higher density and accuracy and makes geo-referenced models possible.

The most widely adopted camera model is for the pin-hole camera which carries out

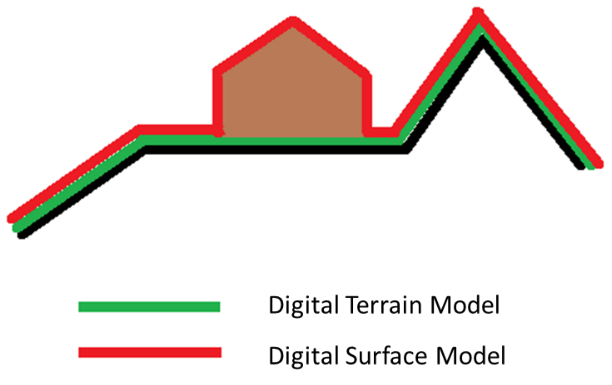


Figure 2.1: A digital surface model includes buildings and other objects on the ground. A digital terrain model represents the bare ground.

a linear central projection [2]. The 3×4 projective matrix in a homogeneous coordinate system is given by:

$$P = KR[I| - C] \tag{2.1}$$

where C is the camera center, I is the identity matrix, and R is the 3×3 rotation matrix derived from the orientation (pitch, roll, and yaw) of the camera. The matrix K is the interior calibration matrix and defined as:

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where f_x and f_y are the focal lengths in the x and y directions, (x_0, y_0) are the coordinates of the principal point, and s is the skew factor.

The way to project a 3D point onto a 2D plane is given by:

$$x = PX \tag{2.3}$$

where X is the 3D coordinates of the point, and x is the 2D coordinates of the projection, they are both in homogeneous coordinate system.

2.3 Point Clouds: Photogrammetry vs. LiDAR

2.3.1 Photogrammetric Technology

Photogrammetry is defined by the ASPRS (American Society for Photogrammetry and Remote Sensing) as "the art, science, and technology of obtaining reliable information about physical objects and the environment through the process of recording, measuring, and interpreting photographic images and patterns of recorded radiant electromagnetic energy and other phenomena"[3].

The conventional way of generating 3D information in remote sensing has always been the photogrammetric method. Traditionally, it requires delicate human effort, and is a time consuming process. However, several innovations help to develop novel automated photogrammetry. Firstly, digital imagery collection enables large overlap between images with negligible cost. The high overlap increases the chance of successful control point matching and minimize the undetected errors. A typical sixty percent forward-lap and thirty percent side-lap place a higher reliance on ground control points matches than imagery with higher piecewise overlap do. Secondly, the progress in automatic feature detection and matching in the field of computer vision provides greater opportunities to automate photogrammetry with no ground truth [4]. In conventional photogrammetry, an human stereo operator can create matches between two images with the accuracy of a few pixels at best [4]. The automatic dense matching can achieve a level of accuracy as small as one pixel. Thirdly, the development of sensor technology provides very high resolution airborne or satellite imagery as well, which makes the high density point cloud generation process possible.

2.3.2 Airborne LiDAR

LiDAR (Light Detection and Ranging) is a remote sensing technology that operates in a similar fashion to RADAR sensing, but uses laser light instead of radio waves. LiDAR scanners, which can be either ground-based or airborne/spaceborne, generate 3D point clouds of their environment by emitting pulses of light and precisely timing their reflections from a target, or in other words, the sensor-target-sensor round trip distance (see Fig.2.2). This timing information is used to create a point cloud, a large set of 3D points that reflects laser-target interactions. The data often requires extensive processing to filter noise and produce a meaningful 3D point cloud.

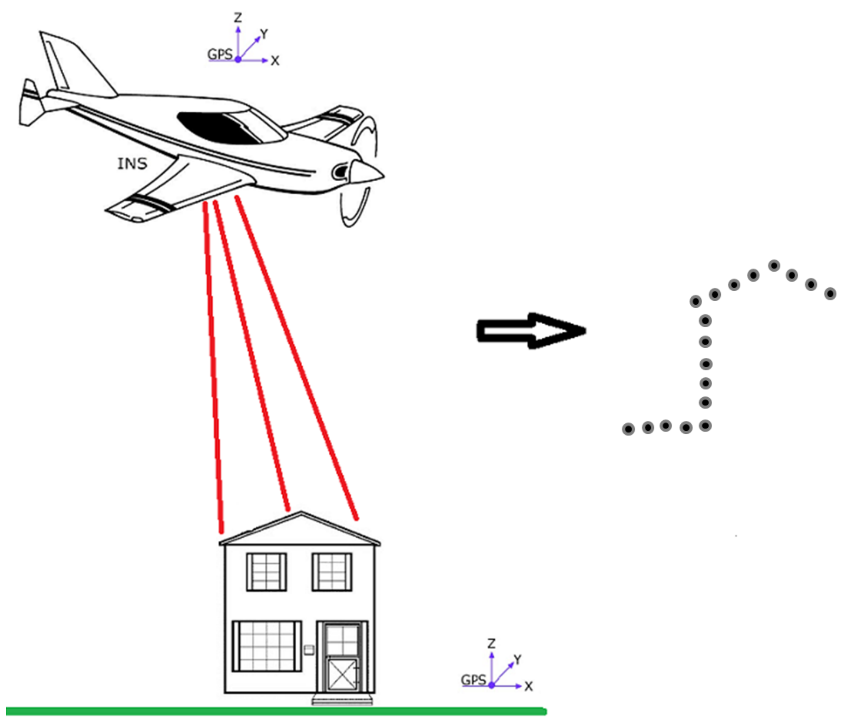


Figure 2.2: Data collecting by airborne LiDAR. The air plane carries a laser scanner to measure the time traveling distance (on the left), generating a point cloud (on the right).

2.3.3 Which is better?

There is no definitive statement that claims which technology is better, image-derived or LiDAR point clouds. During the past decade, LiDAR has gain enormous attention in industries for practical applications. Leberl *et al* [4] conducted two test projects to compare point clouds generated from airborne and ground based LiDAR systems with those created from optical images. Their result shows the accuracy of the photogrammetric method is still comparable when compared to the LiDAR based method. The density is actually much higher from the images than from the laser scanned data. Fifteen additional advantages are identified by their work in order to support the vision that photogrammetric methods are still going to be very valuable even though under the pressure of the rapidly growing usage of LiDAR.

2.4 RANdom Sample And Consensus (RANSAC)

2.4.1 Overview

Most building model reconstruction problems require extracting rigorous geometric features from the input data in the form of 3D irregularly spaced points. These geometric features are commonly recognized as lines, planes, and other types of surfaces in 3D. This estimation can be done by several well-known minimization approaches, such as least squared minimization. However, outliers in the original input are always a significant obstacle to the estimation of these different geometric features, and can jeopardize the final results dramatically. So, in most modern applications, when it comes to problems like estimating parameters for a well-defined mathematical model, researchers first think of the RANSAC algorithm that could deal with an input contaminated by a significant number of outliers.

The RANSAC algorithm was first introduced by Fischler and Bolles in 1981 [5]. An outlier is an observation or data point that does not fit the model defined by a set of parameters within some error threshold. Despite many improvements and modifications, the RANSAC algorithm essentially consists of two stages that are conducted iteratively [6]. The first stage is called the hypothesis stage. Minimal sample data are randomly chosen and used to compute the model parameters. The second stage is called the testing stage. The algorithm checks other elements in the entire dataset to see if they are consistent to the temporary, or current estimate of, model. During each iteration, the algorithm examines if the current consensus set is better than any previous one. It terminates when it is highly unlikely to find a better consensus set.

2.4.2 RANSAC Options

RANSAC has become a crucial tool in the computing society. Since the original idea came out, a great number of variations have been developed. Torr and Zisserman [7] proposed two improved versions of RANSAC called MSAC (M-estimator SAmple and Consensus) and MLESAC (Maximum Likelihood Estimation SAmple and Consensus), respectively. MLESAC tries to find the solution that maximizes the likelihood rather than the number of inliers.

2.4.3 Estimation of 3D Plane

The specific RANSAC implementation utilized in this work is mainly used for plane estimation.

Parameter Estimation

The model of the plane is defined as:

$$p_1 \cdot x + p_2 \cdot y + p_3 \cdot z + p_4 = 0 \tag{2.4}$$

where (p_1, p_2, p_3, p_4) represent the four coefficients defining a plane $\mathcal{M}$ in three dimensional space.

A number of potentially coplanar 3D points $(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)$ could be used to find the plane, we can write:

$$\begin{cases} p_1 \cdot x_1 + p_2 \cdot y_1 + p_3 \cdot z_1 + p_4 = 0 \\ p_1 \cdot x_2 + p_2 \cdot y_2 + p_3 \cdot z_2 + p_4 = 0 \\ \vdots \\ \vdots \\ p_1 \cdot x_n + p_2 \cdot y_n + p_3 \cdot z_n + p_4 = 0 \end{cases} \tag{2.5}$$

Let's rewrite it as:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ : & : & : & : \\ x_n & y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = A\bar{\mathbf{p}} = 0 \tag{2.6}$$

The optimal parameters are given by:

$$\bar{\mathbf{p}}_{opt} = \arg\min \|A\bar{\mathbf{p}}\|^2 \; s.t. \; \|\bar{\mathbf{p}}\|^2 = 1 \tag{2.7}$$

for which the solution can be computed using Singular Value Decomposition (SVD). A plane is uniquely defined by three points, so the cardinality of the minimal sample data is 3.

Error Estimation

The fitting error is calculated as the distance between each point and the plane $\mathcal{M}$ instantiated by $\bar{\mathbf{p}}$. The error between a point (x, y, z) and its orthogonal projection onto the plane $\mathcal{M}$ is given by:

$$e^2 = \frac{([\ x \quad y \quad z \quad 1\ ]\bar{\mathbf{p}})^2}{p_1^2 + p_2^2 + p_3^2} \tag{2.8}$$

assuming the data is affected by Gaussian noise, e^2 is χ^2 distributed [6].

2.5 Graph Cuts in Segmentation

The work presented here, a very critical requirement is the ability to identify trees, buildings, and the other background objects in a large scale scene, which can be considered as an object segmentation problem. Among the fruitful N-dimensional image segmentation methods, the graph cuts has emerged as a very powerful tool to separate components in imagery for researchers in the computer vision community. What the graph cuts algorithm does is it takes several classification decisions jointly, given a set of discrete variables (nodes in the graph), and it labels each variable while taking into account dependencies between variables. There are two reasons that motivate the use of graph cuts [8]. Firstly, the graph cuts is easily interpreted in the geometric form.

Secondly, the graph cuts algorithm also works as a powerful energy minimization tool in many vision problems. In particular, the graph cuts approach defines the minimal cut (smallest sum of link weights) of a discrete graph representing the pixels of 2D images or points of 3D data. The cut can also be seen as a hypersurface in N-dimensional space.

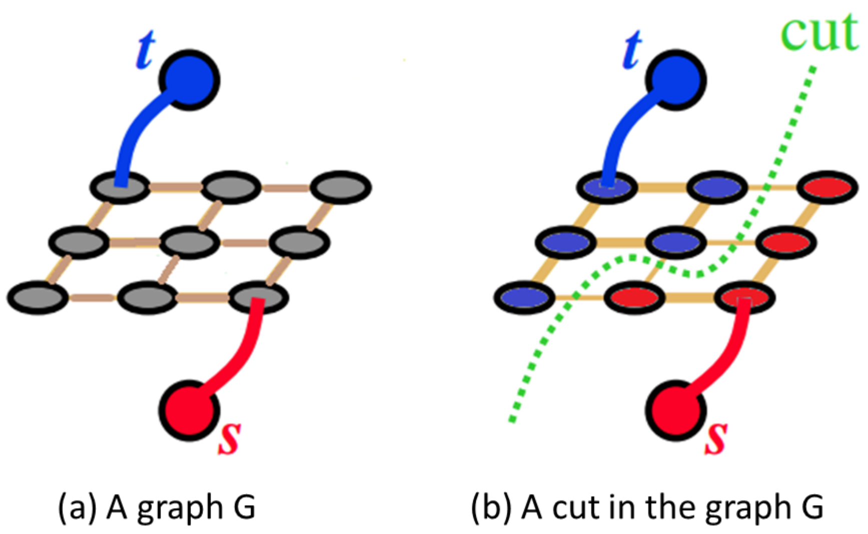


Figure 2.3: A demo graph with two terminal nodes s and t. Each node is connected with its four neighbors. A minimal cut is found and then the graph is divided into two parts.

2.5.1 Description

In this section, the basic terminology and a review of minimum cut/max flow problem is introduced. Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a graph which includes a set of nodes $\mathcal{V}$ and a set of directed edges $\mathcal{E}$. The node set $\mathcal{V}$ has two special terminal nodes that are the source s and the sink t. Fig.2.3 shows a simple case of a graph with the terminal nodes s and t, and an s/t cut is conducted.

All directed edges could be assigned some weight w which is nonnegative and can be interpreted under some certain context. All edges can be broken into two groups in the graph, namely, n-links and t-links. A t-link connects a non-terminal node with a terminal node (blue and red links in Fig.2.3). An n-link connects a pair of non-terminal nodes (brown links in Fig.2.3).

2.5.2 The Min-Cut and Max-Flow Problem

A cut is a partitioning that divides the graph into two disjoint subsets $\mathcal{S}$ and $\mathcal{T}$. $\mathcal{S}$ is defined by s, and $\mathcal{T}$ is defined by t. In combinatorial optimization, the cost of a cut is obtained by adding up the costs of all "boundary" edges (p, q), where $p \in \mathcal{S}$ and $q \in \mathcal{T}$. The minimum cut problem is to find a cut with the minimum cost among all possible cuts. Actually, in combinatorial optimization, the solution of the minimum s/t cut can also be determined by finding a maximum flow from the source s to the sink t. Ford *et al*[9] stated that a maximum flow from s to t is equivalent to a minimum cut. As matter of a fact, the value of the maximum flow is equal to the cost of the minimum cut.

For most computer vision problems, people usually deal with 2D, 3D, or even higher-dimensional data. Boykov *et al*[10] [11] developed a fast algorithm that outperformed other previous methods. With the development of GPUs, many accelerated methods have been implemented, which enable a variety of real-time applications.

2.5.3 Binary Optimization by Graph Cuts

The graph cuts algorithm is a powerful tool in optimization. It was inherently designed for binary problems. Speaking informally, the input is always supposed to be transformed into a graph and then divided into two desirable subgraphs after the optimal cutting. In Fig.2.3(b), all non-terminal nodes are grouped into two sets, $\mathcal{S}$ and $\mathcal{T}$. Each node in the graph is labeled with the binary value (0 or 1) according to a given minimum cut. If node $p \in \mathcal{S}$, it is labeled as 0, and if node $p \in \mathcal{T}$, it is labeled as 1. The cost of each cut can be given by:

$$c(\mathcal{S}, \mathcal{T}) = \sum_{p \in \mathcal{S}, q \in \mathcal{T}} w(p, q) \tag{2.9}$$

where w is weight value associated with the edges. This cost could directly be represented by the energy function of the minimization. The graphs cuts algorithm can achieve the minimization goal in polynomial time. This is very similar to dynamic programming. However, dynamic programming only applies to the tree structure, and it determines that the energy functions that the graph cuts can solve are limited. Kolmogorov and Zabih [12] presented a thorough discussion on the kinds of binary energy functions that can be minimized by graph cuts.

Unfortunately, most computer vision problems are ill-posed. The energy function defined as Eq.2.9 is not adequate. Constraints or regulations derived from the data are required to obtain a reasonable solution. Piecewise smoothness is a natural constraint or regularized term in most vision problems. In a graph, each node is connected by a number of nearby nodes which are considered as "neighbors". Labeling this query node should be consistent to the labels of its neighbors in order to maintain the piecewise smoothness. Thus, like many computer vision problems, the graph cuts approach can formulate the energy function as Eq.2.10 that extends the cost function Eq.2.9.

$$E(l) = E_{data}(l) + E_{smooth}(l) \tag{2.10}$$

where l is a label in some finite label set $\mathcal{L}$ that only has two labels in binary case. E_{data} is typically called the data term which can be defined as

$$E_{data} = \sum_{p \in \mathcal{V}} D_p(l_p) \tag{2.11}$$

where D_p penalizes labeling l_p as the element p. E_{smooth} is called the smoothness term to keep the piecewise consistency. The general form is shown in Eq.2.12, where $\mathcal{N}$ is the small neighborhood where p and q are connected. The choice of the smoothness term is quite open, whereas it is very critical and needs to be guaranteed to be as robust as possible.

$$E_{smooth} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(l_p, l_q) \tag{2.12}$$

Generally, the data terms are embedded into the t-link (linkage to terminals) weights, and the smoothness terms are encoded into the n-link (linkage between nodes) weights while implementing the graph.

2.5.4 Multiple Label Optimization by Graph Cuts

Even though the graph cuts algorithm provides an inherently binary solution to optimization problems, it can also be used for multiple labeling problems. At this time, the finite label set $\mathcal{L}$ consists of more than two labels. The minimization is usually approximate as a matter of fact.

2.5.5 α-Expansion Move and α-β Swap Move

Both the expansion and the swap algorithms in discrete optimization try to find a local minimum for any defined energy function. The number of expansion or swap moves from each labeling action is exponential with relation to the number of sites in the graph. However, it is possible to compute the optimal α-expansion or the optimal α-β swap on a graph. Given a label α, a move called an α-expansion move occurs upon some site in the graph that was not labeled α is now labeled α. Similarly, given a pair of labels α and β, a move called an α-β swap move occurs upon some site in the graph that were labeled α are now labeled β, and some sites in the graph that was labeled β is now labeled α.

Boykov *et al* [10] developed efficient graph-based methods to find the optimal α-expansion or α-β swap given a labeling action l. The algorithms are quite similar in their structure and are guaranteed to terminate in a finite number of cycles.

2.6 Region Growing

Region growing is well known as a simple, low-level region-based, image segmentation method. The dimensionality of the image here can be generalized to n-dimension. It involves the selection of initial seed points.

Let each region be referred to as R_i, where $i = 1, 2, ..., n$. There are a couple of rules formulating region growing base segmentation approach:

- The region must fulfill the completeness criteria. $\bigcup_{i=1}^{n} R_i = R_{whole}$.
- Points in a region R_i must be connected based on some predefined criterion, for instance, the points that have the same color are grouped as one region.
- Each individual region R_i is disjoint to others. $R_i \bigcap R_j = \emptyset$ for all $i, j = 1, 2, ..., n$.

- Each region R_i has its own unique property. Every two adjacent regions can not have the same property.

In practice, region growing segmentation is mostly applied to 2D gray-scale images. It exploits the important information that neighboring pixels have similar intensity values. The basic steps are:

- Select the seed pixel.
- Examine the neighboring pixels and add them to the region if they are similar to the seed point.
- Repeat the second step for every newly added pixel, and stop if no more pixels could be added to the current region.
- For the remaining unprocessed pixels, repeat the previous steps.

How to choose the seed depends on the nature of the problem. It could involve some prior knowledge if it is available. Randomly picking one works fine in most cases without any prior knowledge. The initial region starts as the location of the seed point. The criterion of a region membership could be, for example, pixel intensity value, texture, color. Four-connected neighborhoods or eight-connected neighborhoods are widely used as the adjacency relationship to grow regions. Adjacent points of the seed point are kept if after examining and classifying they fulfill all the criteria. The whole process is iterative and stops until when significant change is observed between successive iterations.

2.7 Polyline Simplification Methods

The production of rooftop outlines is also a significant aspect of this research. In most urban areas, the shapes of many rooftops are relatively simple geometric polygons consisting of lines, right angles, and so on. Algorithms addressing polyline simplification can shed light on this particular problem. There are several polyline simplification algorithms that are worthy of being investigated. In this work [13], the popular Douglas-Peucker polyline simplification method is used to generate the first estimation of the rooftop outline from LiDAR points. Lach and Kerekes [14] utilized a sleeve-fitting approach as a key step to produce building roof boundaries from LiDAR points. Luebke [15] provides a though survey of polygonal simplification algorithms from a developer's perspective. This article identifies issues related to the strengths and weaknesses of different approaches.

2.7.1 Douglas-Peucker Algorithm

The most widely used, high quality simplification algorithm is the heuristic method known as the Douglas-Peucker algorithm [16]. Its premise is to reduce the number of points in a curve and approximate the curve with fewer points. The simplified curve has a subset of the points that constructed the original curve. At each step, the Douglas-Peucker algorithm attempts to fit a sequence of points by an approximate line segment connecting the first point and the last point. The furthest point, of the collection, from this line segment is found, and if its distance from the line segment is smaller than some predefined threshold, the line approximation is kept and this point is discarded. If this furthest point is too far from the approximated line, this point becomes a new end point of two line segments that connect the previous two end points. The algorithm runs recursively, dividing the line into the appropriate set of segments.

2.7.2 Sleeve-fitting Algorithm

Zhao and Saalfeld [17] present a linear time sleeve-fitting polyline simplification algorithm. The basic version of the algorithm utilizes a variable angle tolerance measure to compute maximal subsequences of points of the polyline that could be replaced by a single line segment. A term called ϵ-buffering is presented in their paper. It is proven that the angle-testing procedure locally equals to ϵ-buffering. The maximum sleeve, that is a 2D rectangular strip, can be found by iteratively angle-testing and covers the largest number of consecutive vertices. The center-line of each sleeve is a one-segment approximation.

2.8 Related Work

The automatic reconstruction of urban building models has become a critical interest of both photogrammetric and computer vision research during the past two decades. This section will discuss the previous research into three dimensional urban building modeling methods as well as provide a summary of the methods. It is difficult to categorize all the existing reconstruction approaches, since they can be identified by several properties, such as the type of input data, the levels of detail, and the degree of automation. At an early time, Jinhui *et al* [18] conduct a survey on large scale urban modeling technologies using a variety of sensors and data acquisition techniques. They categorize the existing approaches based on photogrammetry, active sensors, and hybrid sensor systems and examined them with respect to several performance criteria, such as data acquisition source, level of user interaction, geometric fidelity, model completeness, and potential applications.

There has been a plethora of work conducted for urban modeling from airborne aerial images, LiDAR data, or the combination of the two. Musialski *et al*[19] recently provided a more comprehensive overview of urban reconstruction from different perspectives that are not restricted to aerial data inputs. The complexity and difficulty of this problem has been approached in many ways, but the synergistic use of muli-modal datasets has become a prominent pedagogy in this research area. Haala and Kada[20] conduct a review of a number of state-of-art reconstruction methods and their principles. They point out ”the difficulties of aerial image interpretation also motivated the increasing use of three-dimensional point clouds from laser altimetry as an alternative data source”.

2.8.1 Building Footprint Detection

Man-made building structures are typically the main objects of interest in this research area. Many works have focused on or start with detecting and isolating buildings from the entire scene. The extraction of man-made objects is essentially an object recognition or segmentation problem and the primary step of the reconstruction process that requires are to find each rooftop from 2D imagery or irregular point clouds. Maas and Vosselman [21] pointed out that segmentation of laser altimetry data may be obtained in several ways from an earlier point of view. In many cases, available GIS information can be used as reliable and accurate auxiliary information. For those regions with limited terrain roughness, a reasonable threshold value and the boundaries of buildings could be indicated by detecting the local maxima and analyzing the histograms of regions around the maxima. The derivation of digital terrain models from laser scanning data could be applied because it could be considered the inverse of the building detection task. Integration of an airborne laser scanner and a multispectral or hypersectral scanner would also be option if this state-of-art technology is available.

Some very early work started looking at how to extract buildings from high-resolution Digital Surface Model. Weidner [22] introduced a binarization method using a building related threshold. An approximation of the topographic surface was computed using a morphological opening. The difference between the original DSM and the approximated topographic surface contains the information describing the buildings. Brunn *et al*[23] described an approach for building extraction using DSM. They used the height and differential geometric information to discriminate building structures and vegetative areas. Two approaches, binary classification and Bayesian networks, were introduced. Both require high quality data. Morgan *et al*[24] proposed a reconstruction procedure that began by re-sampling the irregular points from the laser scanned data into a regular grid. They also applied a morphological filter with an adaptive window size for distinguishing between terrain and non-terrain parts. This is the core idea for building footprint detection. The non-terrain segments are further classified into building area or vegetative area.

Wang *et al*[25] developed a Bayesian method for detecting building footprints automatically from the LiDAR data. The point cloud has to be first segmented into buildings, trees, and grass as a pre-processing step.

Lafarge *et al* [26] extracted building footprints from Digital Elevation Models (DEMs) that are generated from high resolution satellite images. The first step is to extract a rough description of the shape of buildings (rectangular approximation). A method based on marked point processes is used to achieve this purpose. This method is to minimize an energy function by applying a Reversible Jump Markov Chain Monte Carlo (RJMCMC) sampler embedded into a simulated annealing scheme. The rough rectangular layouts are then regularized by connecting neighboring rectangles and detecting roof height discontinuities. So, the final obtained building footprints are non-overlapping structured footprints. In the end, a very simple 3D reconstruction process is introduced.

Matei *et al* [27] presented a building segmentation system for dense urban areas, where the ground is 20% or less of the area. They deal with 3D unorganized points by employing the 3D grid. Each processing element is then a voxel instead of a point. In the first stage, each point is classified into ground and non-ground, and the terrain is modeled. In the second stage, building segmentation is done by estimating surfels at voxels locations containing non-ground points. Roof surfaces are refined using the expectation-maximization (EM) algorithm. A loopy belief propagation framework is used to enforce neighborhood constraints and sharp boundaries between regions.

Carlberg *et al* [28] introduced a multi-category classification system for identifying water, ground, rooftop, and trees from airborne LiDAR data. The whole system acts as a cascade of binary classifiers. 3D shape analysis and region growing are used to recognize ”planar” and ”scatter” regions which most likely correspond to terrain/rooftops and vegetations respectively.

2.8.2 Image-based Reconstruction

Moons *et al*[29] proposed a method to automatically reconstruct 3D polyhedral models of generic house roofs from aerial images of residential and urban areas. It addresses the limitation of using predefined roof models for their reconstruction due to the wide variety in shapes. This method requires that accurate camera model information is readily available. The process is formulated as a feed-forward scheme involving four stages: 2D edge detection and region selection, line segment matching and 3D reconstruction, 3D grouping and polygonal patch formation, and rooftop model generation and model fitting.

Kim *et al*[30] presented an approach to automatically describe a complex rooftop from multiple images. Image derived unedited elevation data (DEM) is used to assist feature matching, and to produce rough cues of the presence of 3D structures. Once all of the 3D line features are obtained, the rooftop hypothesis generation process introduces the next important issue, time complexity. A level-of-detail technique is used to reduce time.

Werner and Zisserman [31] investigated an approach for reconstructing buildings from multiple close-range ground images.The method is targeted at architectural scenes that usually contain planes with three dominant orientations that are perpendicular to each other. The approach is to fit generic models, such as planes and polyhedra, and proceeds in two stages which are coarse model fitting and refinement. The one parameter plane-sweep method is extended with uncalibrated setting for both plane fitting and polyhedral model fitting.

Nevatia *et al*[32] introduced an automatic and interactive modeling method from aerial images. For the automatic approach, 2D features such as lines, junctions and parallel relationships, are extracted from multiple images. 3D features are then derived from groups of 2D features matched over images. A rough DEM produced by stereo matching is used to aid feature matching. The last step is to use rigorous Bayesian learning to select from many potential models in a hypothesized and test paradigm. Rau *et al*[33] presented a Split-Merge-Shape algorithm for 3D building modeling in which an accurate scheme for 3D roof-edge measurements is proposed, however, this idea is semi-automatic, still requiring manual intervention.

The above image-based methods all require some level of extra input besides images. A widely studied process of recovering 3D structure without calibrated information from a set of images that have been taken by a camera that was in motion is called structure from motion (SFM). Snavely *et al*[34][35] developed a state-of-art system for generating sparse point cloud of urban environments based on SFM. However, point clouds from SFM are rather sparse and do not indicate any solid geometry. Dense matching is usually used to obtain more dense structure. Multiview stereo is the technique used for dense matching. Seitz *et al*[36] provided a detailed overview on multiview stereo reconstruction algorithms.

2.8.3 LiDAR-based Reconstruction

Elaksher *et al*[37] developed an approach to utilize the geometric properties of buildings for the reconstruction of building wire-frames from LiDAR. The finding of building candidate points is done by convolving the LiDAR DEM with a minimum filter. What this filter does is to classify points above a certain height as building points. It is a weak assumption in general, but works well in their limited data set. The building points are then used to populate a plane parameter space, which is an application of the generalized Hough transform. Once planes representing the roof surfaces are found, roof regions are then extracted, and the parameters are refined. The boundaries are used to form the wire-frames of the buildings.

Rottensteiner [38] presented a method for automatically creating polyhedral building models without involving ground planes. The whole method consists of two steps: building detection and building reconstruction. After obtaining regions of interest for the geometric reconstruction of the buildings, the system handles those ROIs individually. The system performs building region detection by thresholding the height differences. The geometric reconstruction of the buildings consists of four steps: detection of roof planes, grouping of roof planes and model generation, consistent estimation of the model parameters, and model regularization. This techniques handles polyhedral buildings of arbitrary shape. It does not require assumptions about the regularity of the footprints.

Verma *et al* [39] detect and reconstruct 3D geometric models of an urban area with complex buildings using aerial LiDAR. In their approach, the complex roof types refer to as saddle-back rooftops could be represented as a combination of simpler roof types, such as rectangular, L-shape, and U-shape geometric primitives, that all have a uniform ridge and gutter height. Segmentation of the roof and terrain points is done first. All points in the point cloud that are not locally co-planar are abandoned assuming that non-flat points constitute mostly points that fall on trees, fences, and utility poles, etc. Once all the roof points are identified, they fit local planar patches to the roof points and group them based on their surface normals. Next, a roof-topology graph is constructed that not only establishes geometric constraints between faces, but also helps recognize sub-roofs models by simpler shapes. The final step is geometric fitting. The roof refinement process fits a watertight polyhedral model to the LiDAR points.

Zhou and Neuman[40] presented an automatic approach for reconstructing building models from airborne LiDAR data of urban areas. The workflow highlights vegetation detection, boundary extraction, and roof orientation determination by a data-driven algorithm. The output of the workflow are polygonal models. For classification of vegetation from other urban objects, a Support Vector Machine algorithm requiring one-time training is introduced that utilizes several differential geometric properties instead of using information such as height and intensity. Points on rooftops are separated from the ground points by extracting planar patches. Roof boundary extraction is an extension of work from a 3D octree contouring algorithm in volumetric geometry processing[41] preserving the topology of original data. At this last stage, principal directions of building layouts are determined handling arbitrary angles and requiring no presumptions or preferences. At the end of the workflow, they claim non-flat surfaces could also be supported involving minimum human interaction. Zhou and Neuman [42] afterwards presented a streaming framework for seamless building reconstruction from huge aerial LiDAR point clouds. The building modeling pipeline introduced in [40] could be adapted into this streaming framework, and it gives the ability to deal with hundreds of millions of points in an uniform manner. It is the first attempt to employ streaming techniques for building modeling from LiDAR data though these techniques that have been widely used in computer graphics. A series of streaming operators and the order in which to perform them are defined, and then different modules, including classification, segmentation, and modeling in the building reconstruction pipeline are well adapted to this streaming process.

Dorninger *et al*[43] proposed a comprehensive approach for automated determination of 3D city models from LiDAR point cloud data, but their approach involves an interactive initialization referred to as the coarse selection of building regions. Poullis and You [44] presented a rapid, automated modeling approach for large-scale area reconstruction by using statistical considerations for the segmentation of the buildings. The goal is to create lightweight, watertight polygonal 3D models from LiDAR data. Three steps are involved, which are preprocessing, segmentation, and modeling. The raw 3D point cloud data is first projected on a 2D X-Y map by re-sampling. It is common knowledge that 3D raw data is very often too large to be processed at time. Subdividing the raw data into smaller parts and transforming it into 2D is an efficient representation. The proposed clustering approach is an underlying region growing method based on the statistical analysis of the geometric properties of the data. The modeling part involves common surface fitting and boundary refinement techniques.

Toshev *et al* [45] presented a method for detecting and parsing buildings from unorganized 3D point clouds that is gathered from a set of range measurement. They address the problem from the perspective of central scene understanding from raw, unorganized points. In their work, a building is viewed as a tree whose nodes are volumetric parts covered by roof and roof parts. All roof or roof parts are the children of the tree. The hierarchical tree description is claimed to be a natural organization since a building usually has a main body and other attached smaller parts. They start by parsing the whole point cloud of a city. Two supernodes, "building" and "non-building" are introduced. All nodes descending from the "building" supernode are supposed to represent city architecture. A simple grammar is defined consisting of a set of terminals, a set of non-terminals, a production set, and two supernodes. An efficient dependency parsing algorithm is used to generate the desired semantic description.

Sampath and Shan [46] proposed a solution framework for the segmentation and reconstruction of polyhedral building models from airborne LiDAR point clouds. The eigen-analysis is first carried out for each point within its Voronoi neighborhood instead of k-nearest neighbors. Planar points and non-planar points can be separated based on the surface normals. All planar points are considered as roof parts and the fuzzy k-means method [47][48] is used to cluster them. The potential-based clustering is used to estimate the number of cluster in order to feed the fuzzy k-means method. In the final step, the parallel and coplanar segments are separated based on the distance and connectivity.

Verdie *et al* [49] presented an automatic approach for producing accurate, watertight and compact building meshes under planar constraints that are especially designed for urban areas from aerial LiDAR data. The LiDAR points are classified through a non-convex minimization method for the purpose of labeling points belonging to building rooftops. A compact mesh generation is performed in two steps: mesh initialization and mesh simplification with topology preservation. A quadratic-edges collapse-decimation based algorithm [50] is iteratively used to reduce the size of the mesh.

Lafarge *et al*[51],[52] presented a novel and robust method for modeling cities from point clouds. The algorithm is able to construct simultaneously buildings, trees and terrains. They use LiDAR point clouds for experiments but claim it is not restricted to LiDAR data inputs.

2.8.4 Fusion of Optical Imagery and Range Data

Many recent achievements in the field of automatic generation of 3D building models are based on merging data from two or more sources, such as digital optical images and LiDAR data. The purpose of this is to overcome the drawbacks of particular sensor types. The ground resolution of LiDAR sensors is still a big limitation. Intuitively, it would be best to integrate aerial images into the workflow rather than using LiDAR solely. Though fusion of LiDAR and optical imagery is an efficient way to construct 3D virtual reality models, one difficulty is registering the optical images with the LiDAR point cloud, which is often regarded as a camera pose estimation problem.

Fr*ü*h *et al*[53] [54] presented a fast approach to automate generation of 3D textured building models with both ground level and airborne laser scans. The DSM created from the far-range laser scans is registered with facade models using Monte-Carlo-Localization.

In consideration of the combination of LiDAR and geometrically uncorrected image data for urban modeling, some work has been done on automatic registration of aerial images with LiDAR data. Wang *et al*[55] proposed a registration method based on 3D feature detection and matching. A novel feature called 3CS (Three Connected Segments) is proposed that is composed of connected line segments. 3CS features detected from both aerial images and LiDAR point clouds are putatively matched. A two-level RANSAC algorithm is used to remove outliers.

Ding *et al* [56] developed a fast automated algorithm for texture mapping oblique aerial images onto a 3D model derived from airborne LiDAR data. They first coarsely estimate the camera position and the rotation angles with the help of information from a global positioning system aided inertial system. The pitch and roll angles are estimated from the vanishing points of vertical lines. Vanishing points of non-vertical lines are also detected to be used for corner extraction from the images. The second step of the approach is to use 2D corners extracted from the LiDAR DSM as well as aerial images. The matches between DSM 2DOCs and image 2DOCs are generated based on steps of processing, including similarity comparison, Hough transform, and generalized M-estimator sample consensus. In the end, Lowe's camera pose recovery algorithm [57] is used to refine camera parameters for texture mapping.

Mastin *et al*[58] [59] introduced a novel idea for utilizing mutual information between the LiDAR and the 2D imagery, which analyzes the statistical dependency in scenes of optical appearance with measured LiDAR elevation. The LiDAR data adopted in their research has a probability of detection value that represents the number of measured photons along with the x, y, and z value. The definition of mutual information used is the Kullback-Leibler (KL) divergence of the joint distribution and the product of marginals. So, maximizing MI is equivalent to maximizing the KL divergence. Three methods of evaluating MI are proposed. The first is simply the MI between elevation in the LiDAR data and luminance in the image. The second is using the MI between the luminance in the image and probability of detection in the LiDAR data, which provides slight benefit in accuracy. The final method is a combination of the first two.

This information merger has advantages, however, finding correspondences between these two different types of data automatically is often problematic.

2.9 Objective of this Research

This research describes the development of an aerial LiDAR based reconstruction workflow. The major limitations existing in previous works on this problem include requiring a particular scene type to work on, heavily adopting primitive templates, inefficiency on multi-layer roof geometry recovery and the lack of ability to provide accurate and clean boundaries. In this research, the objective is to overcome these limitations. The developed workflow is expected to work for a wide range of scene types. No primitive templates are used for estimating the shape of rooftops within the scene. Features on the rooftop are detected and modeled individually, followed by roof polygon intersection refinement to provide finer details in the produced models. Boundaries of roof features are estimated and aligned to the principle orientation that is found using the minimum bounding box technique.

Chapter 3

Methodology

3.1 Overview

Given a LiDAR point cloud covering an urban area, the objective of this research is to automatically detect and model man-made building structures with complex rooftops. The output could be either polyhedral models or triangulated meshing models. In particular, the whole process aims to automatically remove vegetative areas from the scene and then single out each individual building footprint from the terrain. Geometric analysis is also conducted for each rooftop and the goal is to extract as many significant features from the rooftop as possible. The original points on the rooftop are thus divided into a number of clusters, each of which represents a component of the overall rooftop. Shape refinement is further applied to each cluster of points, including surface fitting and outline estimation.

No templates for roof shapes are used. The shape is automatically inferred directly from the LiDAR point cloud. One issue that needs to be pointed out is that the airborne laser scanner mostly "sees" the tops of the buildings. Very few points on the sides of buildings are captured unless the height of the building is large enough for the parallax to assist with this. Due to the lack of information about the geometry of the sides of buildings, it is not possible to portray the details. So, in this research, the roof outlines are extruded down to the ground level and watertight models are then produced. Please see Fig.3.1 showing the workflow for the method employed in this work. The details of the method are briefly summarized in Algorithm.3.1.1.

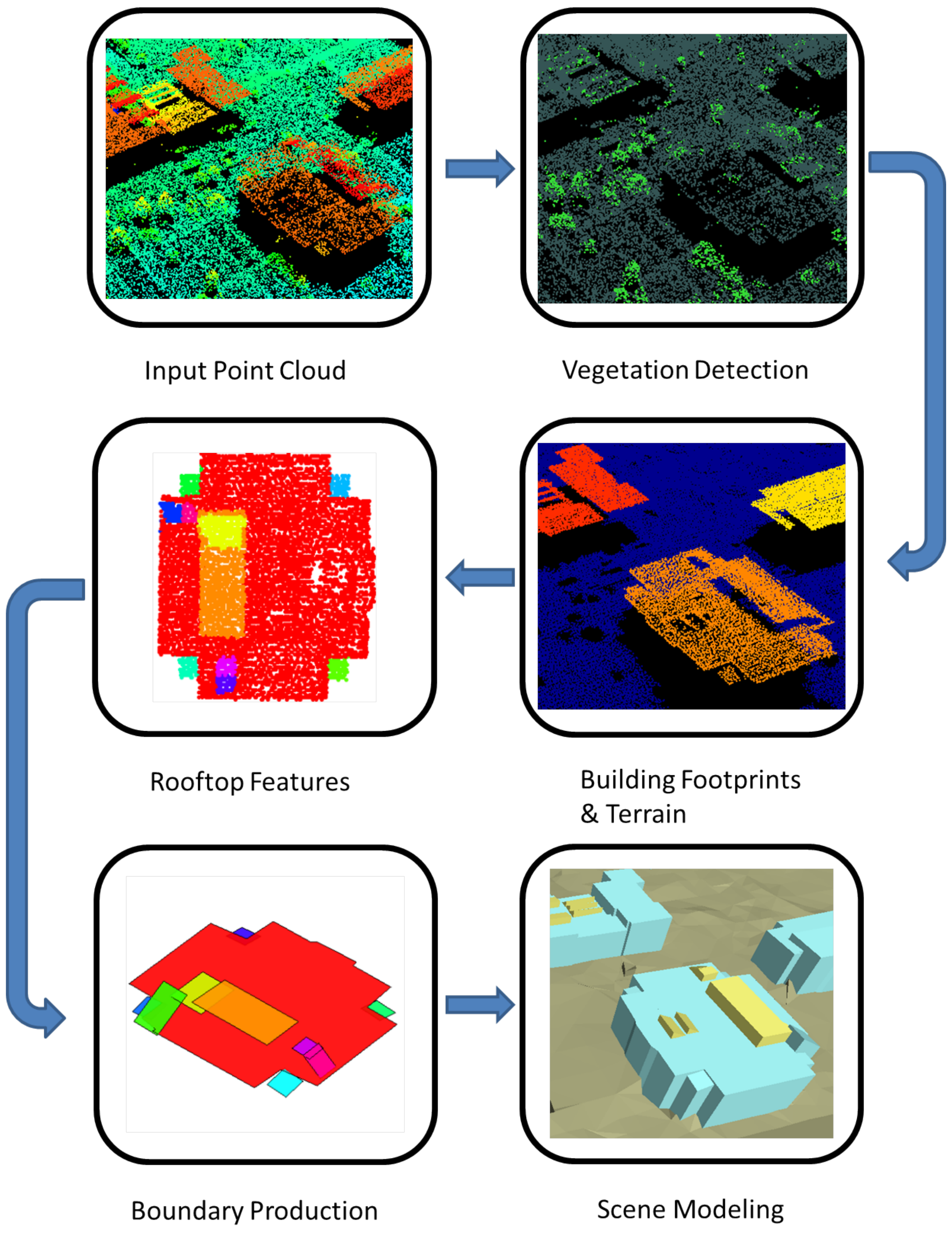


Figure 3.1: 3D scene modeling workflow. It involves a group of modules, such as scene classification, rooftop feature detection, boundary production, and mesh modeling.

Algorithm 3.1.1: BUILDINGMODELING(*pointcloud*, *configuration*)

```
comment: The presented method of building detection and modeling.
$pc\_denoised \leftarrow NoiseRemoval(pointcloud);$

$LocalAnalysis \begin{cases} normals, curvatures \leftarrow NormalEstimation(pc\_denoised); \\ normals\_dist \leftarrow NormalsDistribution(pc\_denoised, normals); \end{cases}$

$trees, nontrees \leftarrow VegetationDetection(pc\_Denoised, normals\_dist, curvatures);$

$terrain, others \leftarrow TerrainExtraction(nontrees);$

$buildings \leftarrow BuildingExtraction(others);$

for $i \leftarrow 1$ to $number\_of\_buildings$
$model(i) \leftarrow ProcessIndividualBuilding(buildings(i));$

return
```

3.2 Point Sampled Representation

One of the main objectives of this research is to reconstruct rigorous geometric models from the point cloud which represents a finite set of point samples. All operations are performed directly on the point cloud that is captured by the airborne laser range scanner (airborne LiDAR).

The input data is given as a point cloud, namely $\mathcal{P} = \{p_1, p_2, ..., p_n\}$, where n is the number of points. Each individual point is often represented merely by its Cartesian coordinates (x, y, z). Some attributes like intensity, number of returns, or color information can also be assigned to each point. However, no additional information associated with the surface properties is provided, such as surface normals and curvatures. Nonetheless, such information with regards to surface properties is the most important information in terms of 3D reconstruction. Building corners defining the outline of the rooftop can not be guaranteed to be captured, which also brings difficulty to 3D modeling. Thus, deriving the surface properties must be done before any further processing.

3.3 Local Analysis

3.3.1 Estimation of Point Density

The point density of the LiDAR point cloud is a very crucial quantity in point cloud processing. The knowledge of the density can assist many other calculations, such as nearest neighbor search and estimation of surface properties. If the density is high, a smaller local area is sufficient to get the needed surface features, on the contrary, a lower density requires a larger local area to do the same surface calculations which can introduce larger errors and more uncertainties.

In this research, since the majority of the points are picked up from the top of scene objects, side points can be simply ignored for point density estimation. All the 3D points can be then ortho-projected on a 2D horizontal plane to get a quick estimate of point density. A 2D convex hull can be found around the projected points. So, the point density is calculated by Eq.3.1.

$$\text{Density} = \frac{\text{Number of points}}{\text{Area of the convex hull}} \tag{3.1}$$

3.3.2 Concept of "Local Neighborhoods"

A local neighborhood can be defined by the spatial relationships of the points in the domain of 3D point cloud processing. Given a point p as the query point, the set $\mathcal{N}^k = \{p^1, p^2, ..., p^k\}$ are the neighbors of p according to some certain rule. The rule can be defined such that the set P^k is sufficient to represent a small surface patch for feature analysis. Pauly [60] defined this kind of 2-manifold surface formally, and also addressed that the definition of a surface is based on local neighborhoods. Pauly [60] also pointed out the local neighborhoods only relies on the geometric locations of the 3D points, not on some other structural information associated with the point cloud. The neighborhood of the given point p is independent on the neighborhood of its neighbor $p^i (i \in [1, k])$. In practice, there are two definitions (shown in Fig.3.2) for $\mathcal{N}^k$ of a given point p that are generally used, K nearest neighbors and radius searching.

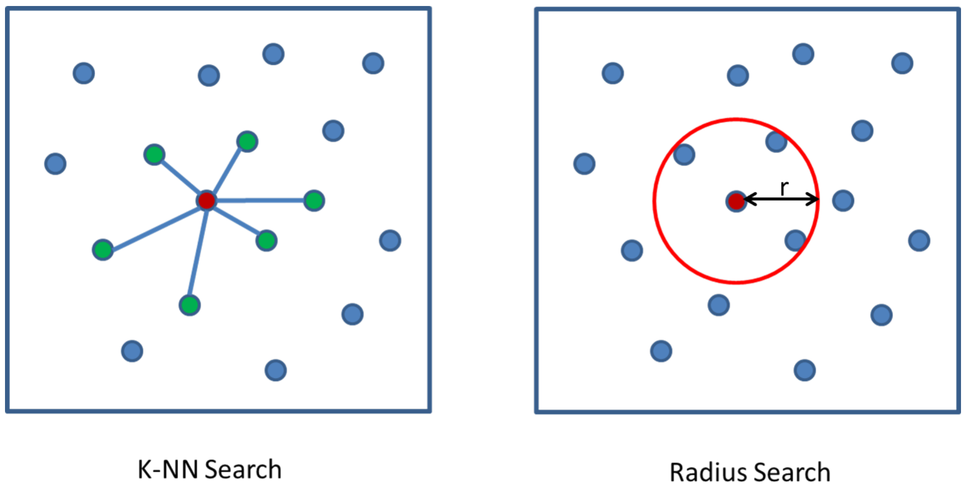


Figure 3.2: Two types of nearest neighbor searching (K Nearest Neighbor searching (K is 6 in the figure) and Radius searching (r is the radius in the figure))

3.3.3 K Nearest Neighbors Searching

K nearest neighbors (K-NN) are the K closest points to the query point based on the Euclidean distance (K = 6 in Fig.3.2). K-NN is an easy and simple to understand algorithm that works very well in practice. This method is adaptive in terms of the region of interest it operates in according to the density of points. Furthermore, K-NN always uses the fixed number of neighbors and avoids the degenerate case, such as a point having zero neighbor. K-NN is non-parametric since it does not require any assumptions about the underlying data distribution. More analysis could be found in Duda and Hart [47]. Brute force searching is not recommended in applications. Many nearest neighbor searching packages use optimization. The most popular one is FLANN (Fast Library for Approximate Nearest Neighbors) [61] [62] and also used in this research.

3.3.4 Radius Searching

Radius searching is essentially an extended version of K-NN. Only neighbors within the region defined by the radius r from the query point are accepted (Fig.3.2). It can be defined as $\mathcal{N}^{k'} = \{q_i | q_i \in \mathcal{P}, where d(p, q_i) < r, i \in [1, k']\}$. d is usually chosen to be

Euclidean distance as well. The number of neighbors selected depends on the density of points, so the number chosen may vary. This approach is not adaptive in terms of the region of interest. Rusu[63] pointed out the radius search is particularly useful for 3D feature estimation since it is not dependent on the number of point sampled, distance or rotation angle of the sensor.

3.3.5 Estimation of Point Normal and Surface Flatness

Once the neighboring point set $\mathcal{N}^k$ of a query point p is determined, it can be used to estimate the local underlying surface properties around p using statistical analysis. Normal estimation indicating the orientation of each point is an important problem in describing the geometric properties of the surface. Normals are heavily used in this research and also other related areas. There are two widely used approaches used for normal estimation, one is based on numerical analysis, and the other one is based on Voronoi diagrams. A thorough study of the two approaches discussing which approach is appropriate under which circumstances is presented by Dey *et al*[64]. The most common and simplest solution is based on plane fitting to the neighboring points [65]. Generally speaking, eigenanalysis of the covariance matrix of points in a local neighborhood provides an efficient algorithm for normal estimation and other related properties, such as curvature. Let $\bar{p}$ be the centroid of all point in the neighborhood $\mathcal{N}^k$.

$$\bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} q_i, q_i \in \mathcal{N}^k \tag{3.2}$$

The 3×3 covariance matrix $\mathcal{C}_p$ for the query point p is given by:

$$\mathcal{C}_p = \frac{1}{k} \cdot \begin{bmatrix} w_1 \cdot (q_1 - \bar{p}) \\ \cdot \\ \cdot \\ \cdot \\ w_k \cdot (q_k - \bar{p}) \end{bmatrix}^T \begin{bmatrix} q_1 - \bar{p} \\ \cdot \\ \cdot \\ \cdot \\ q_k - \bar{p} \end{bmatrix}, q_i \in \mathcal{N}^k \tag{3.3}$$

where w_i is the positive weight factor for q_i, and usually set to 1 when contributions from different points are treated equally. $\mathcal{C}_p$ contains the information of the points within the neighborhood of the query point p. Now the eigenvector problem is presented as:

$$\mathcal{C}_p \cdot \vec{e}_j = \lambda_j \cdot \vec{e}_j, j \in \{1,2,3\} \tag{3.4}$$

The eigenvectors $\vec{e}_j$ correspond to the principal components of $\mathcal{N}^k$. Since $\mathcal{C}_p$ is symmetric and positive semi-definite, all the eigenvalues λ_j are nonnegative real values. The total variation of the neighboring points is given by:

$$\sum_{i=1}^{k} |q_i - \bar{p}|^2 = \sum_{j=1}^{3} \lambda_j \tag{3.5}$$

The tangent plane going through the query point p is defined as:

$$(x - p) \cdot \vec{n} = 0 \tag{3.6}$$

where $\vec{n}$ is the normal vector of the point p. Assuming $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$, the eigenvector $\vec{e}_1$ corresponding to the smallest eigenvalue λ_1 therefore approximates the normal vector $\vec{n} = \{n_x, n_y, n_z\}$ or $-\vec{n}$ (Fig.3.3). Rusu[63] addresses that there is no mathematical way to solve for the sign of $\vec{n}$. The orientation of the normal vector obtained by Principal Component Analysis (PCA) is actually ambiguous. However, the solution to this is simple and easy to achieve. As long as the view point is known for the problem, all normals can be forced to point at the direction of where the view point is. The consistency of the orientation can also be achieved by the Euclidean Minimum Spanning Tree (EMST) of the point cloud proposed in [66].

In addition to the surface normal estimation, the eigenanalysis can also indicate some other properties of the small surface patch around the point p, such as the curvature or the flatness. There are also multiple ways to estimate these properties. The flatness is a quantitative measurement of the surface variation. Remember λ_1 quantitatively shows the surface variation along the direction of the surface normal vector since the eigenvector $\vec{e}_1$ is used as an approximation of the normal vector. The larger λ_1 is, with respect to λ_2 and λ_3, the more likely the points deviate from the tangent plane, and vice versa. So, the variation of the surface around the point p can be defined as:

$$\mathcal{F}_f = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{3.7}$$

This ratio approximates the variation in the flatness in the neighborhood $\mathcal{N}^k$ of p.

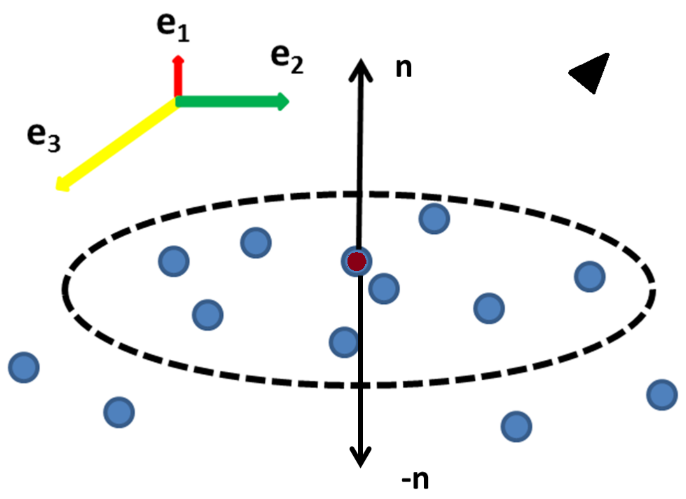


Figure 3.3: Normal estimation according to the eigenvector corresponding to the smallest eigenvalue. The orientation has an ambiguity in nature. The view point (shown in black triangle) can be added to cancel this ambiguity.

Informally, smaller values of $\mathcal{F}_f$ means all the points within the neighborhood are more likely distributed on the tangent plane.

Note that the maximum value of $\mathcal{F}_f$ is 1/3, which means all points are completely isotropically distributed. The minimum value of $\mathcal{F}_f$ is 0, which means all points fall in a plane.

At this point, both estimates are highly affected by noise. Instead of least squared minimization, an alternative approach to fitting a plane using a sample consensus framework. Only inliers in $\mathcal{N}^k$ are considered for the plane model fitting. A similar scheme is introduced by Rusu *et al* [67].

3.3.6 Variation of the Distribution of Normals

A similar eigenanalysis considering the covariance matrix of all the surface normals within a neighborhood is conducted. The covariance matrix is given by:

$$\mathcal{C}_p^n = \frac{1}{k} \cdot \sum_{q \in \mathcal{N}^k} \vec{n}_q^T \cdot \vec{n}_q \tag{3.8}$$

Let $\lambda_1^n < \lambda_2^n < \lambda_3^n$ be the three eigenvalues of the covariance matrix $\mathcal{C}_p^n$. λ_2^n measures the maximum variation of the surface normals on the Gaussian sphere that helps indicate the distribution of normal vectors [60]. So, the variation of the distribution of normals is approximated as:

$$\mathcal{F}_n = \lambda_2^n \tag{3.9}$$

$\mathcal{F}_n$ is another property which functions similarly to $\mathcal{F}_f$. Both of them tend to have larger values when the local surface tend to be rough. Fig.3.4 shows two real examples of the distribution of normal vectors on a Gaussian hemisphere. In the left image, the local surface is flat and horizontal, so all normal vectors cluster together within a tiny region on the top of the hemisphere. On the contrary, such as what is shown in the right image, the normal vectors are sparsely distributed over the surface of the hemisphere because the local surface exhibits significantly more roughness.

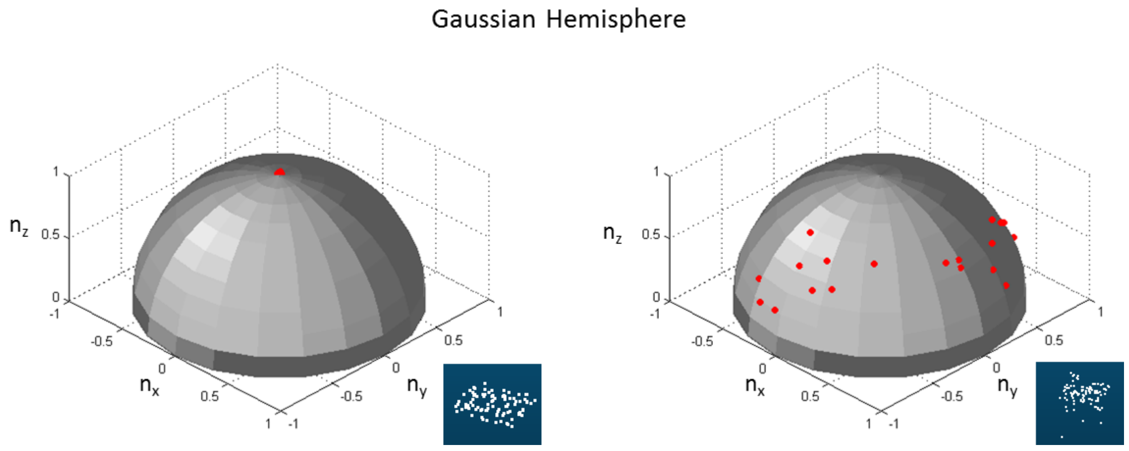


Figure 3.4: Gaussian normal hemisphere. The left figure shows all normal vectors within a small neighborhood are very consistent to each other, and the right figure shows the normal vectors within a small neighborhood distribute sparsely all over the hemisphere (each red dot represents a normal vector). Original points are also presented.

3.4 Classification of the Scene

Automatic 3D reconstruction of urban buildings from a point cloud relies on proper separation of the points of interest from the unwanted scene content (such as trees or small clusters on the ground). This is a scene classification problem. The ultimate goal of classification in this research is to divide the scene into three categories: vegetative areas, the terrain, and building footprints (Fig.3.5). The trees are to be removed initially. A smooth hole free terrain then needs to be modeled. It is commonly acknowledged that vegetation, mainly in the form of trees in urban or residential regions, is the most difficult part to be separated from the building footprints. Tree modeling is not a goal in this research, since trees are irrelevant objects with regards to the task of building reconstruction, mostly serving as an obscuration. Therefore, detection and elimination of trees is the first critical mission to be achieved before any further processing occurs.

Since building modeling is the major objective of this work, it is imperative that vegetative regions are accurately and efficiently identified and removed. Based on the discussion in the previous chapter about related works, almost all the building reconstruction works include scene segmentation or classification due to this requirement. A great number of classification works in the field of remote sensing have been conducted for a variety of purposes [68][69][70]. In this effort, an elevation filter was use to help to distinguish rooftops from trees[13]. The elevation filter is intuitive and simple, but lacks robustness if there is not much difference in height between the buildings and the trees. Sometimes the trees may be higher than the buildings in the scene. Anguelov *et al*[71] present a Markov Random Field based method for detection and segmentation of complex targets from 3D ground based range data. Sedlacek and Zara [72] introduce an interactive graph cuts based segmentation method on image derived point clouds. Golovinskiy and Funkhouser [73] propose a min-cut based method to separate the background and the foreground in point clouds.

In this research, two separate approaches are going to be introduced for vegetation detection and removal. The first approach is simpler and more straightforward, but exhibits larger errors. The second approach is more robust, reliable and generic, and was adopted for following processing.

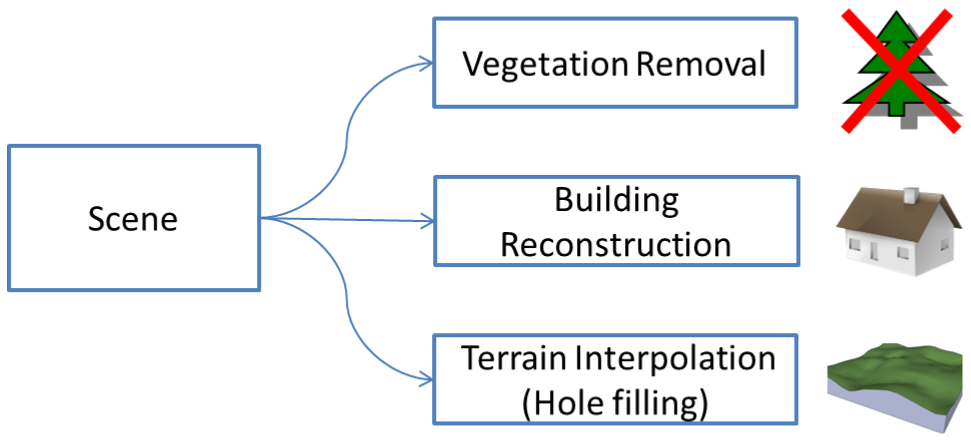


Figure 3.5: Classification of a point cloud representing a scene. Three categories are defined, which are trees, building footprints, and the terrain. Trees are to be eliminated from the scene. Buildings are to be reconstructed. The terrain also needs to be modeled.

3.4.1 Vegetation Detection by Shape Analysis on High Resolution Digital Surface Model (HDSM)

High Resolution Digital Surface Model

As mentioned before, the LiDAR points are irregularly spaced. One typical way to generate regular-spaced DSM is by triangulating the points, temporarily, to a triangulated irregular network (TIN), and then rastering the TIN onto a 2D plane in order to produce a digital surface model (DSM). One major disadvantage of this method is that it highly depends on the performance of the triangulation algorithm, which are typically computationally expensive. So, the resolution of the DSM is often kept low. Isenburg *et al* [74] introduce a streaming way to efficiently generate raster DEMs from mass points. Here, a novel, simpler interpolation method is used to convert irregular-spaced mass points to a regularly-spaced high resolution DSM.

Fig.3.6 shows a small 2D grid whose individual cells or pixels are either occupied or unoccupied by original points from LiDAR. Each blank pixel will be filled with some value which indicates the elevation at this location. For each target pixel, its five nearest neighbors are found and sorted in an increasing fashion according to their elevations. The mean elevation value of the last three is assigned to the target pixel. For instance, if the elevation values of five neighbors are 120.4 m, 121.0 m,150.5 m,150.5 m,150.8 m

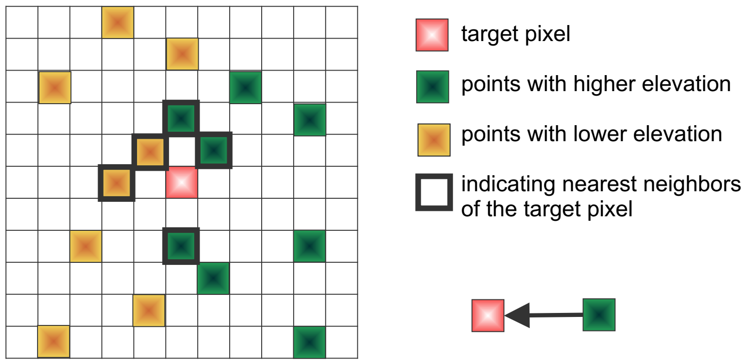

target pixel
points with higher elevation
points with lower elevation
indicating nearest neighbors of the target pixel

Figure 3.6: One target pixel (red solid square) with some original points (green and yellow solid squares) around it. Five nearest neighbors are highlighted with bold boundaries. After sorting, the three green squares become candidates for the elevation estimation of the target pixel. The mean value of their elevations is assigned to the target pixel.

respectively, the elevation of the target is likely to be 150.6 m.

Raw ungridded 3D points are projected onto a regularly spaced grid in which there is no loss of spatial resolution. As much information from the original LiDAR data as possible is retained, and missing information is reasonably estimated. In Fig. 3.7, the left hand image is an unprocessed version of the gridded 2D projection. The figure on the right side is the upsampled and hole-filled 2D HDSM. Note that the map is color coded according to the elevation value of each point. A morphological hole filling algorithm is performed to fill in holes.

Segmentation

The segmentation module takes the HDSM as input and tries to remove the vegetation and detect each individual building footprint. This is very crucial step for the upcoming single-building modeling process. The value of each pixel in the map corresponds to the height of that point. Different from typical optical imagery, the HDSM is already inherently grouped as a number of contiguous areas coded in various colors. The remaining task is to determine which components belong to rooftops, and which belong to the rest of the scene.

The underlying foundation on which this idea is based is that the shape of an individual tree object is similar to a two-dimensional Gaussian-shaped surface. Similar to

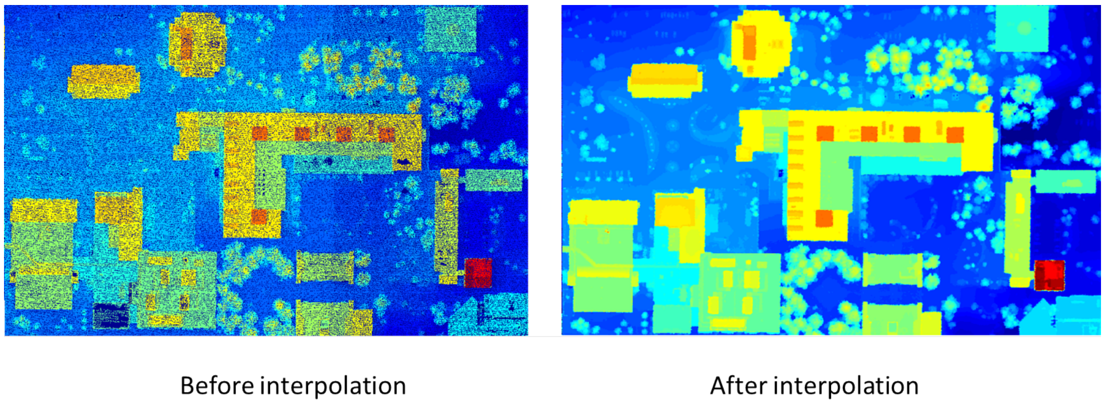


Figure 3.7: HDSM before and after nearest neighbor interpolation.

two-dimensional image correlation with a Gaussian filter, a correlation with a Gaussian surface is performed to the HDSM. Instead of the traditional approach of pixel-wise multiplication, a pixel-wise subtraction is performed. The summation of squared differences (SSD) within the neighborhood defined by the kernel would be the value of the query pixel in the "to-be-generated" tree map ($\mathcal{T}$-map), namely

$$\mathcal{T} = \mathcal{H} \ominus \mathcal{P}(r) \tag{3.10}$$

where $\mathcal{H}$ represents the high-resolution digital-surface map, $\mathcal{P}$ is the r-by-r standard Gaussian kernel, and the sign $\ominus$ represents a neighborhood-wise subtraction operation (SSD operation). Appendix.A.1 demonstrates that SSD operation and the computing of the cross correlation can achieve the same goal of getting $\mathcal{T}$-map. A small SSD value corresponds to a large cross correlation value, which both indicate the two comparing parties are similar to each other.

Fig.3.9 shows a $\mathcal{T}$-map in which most blue regions indicate those areas that are covered by trees. These trees might have the same height as the rooftops, which brings confusion to the elevation filter while conducting roof segmentation. In accordance with the clue provided by the $\mathcal{T}$-map, trees can be removed and the building footprints detection process can be made much easier.

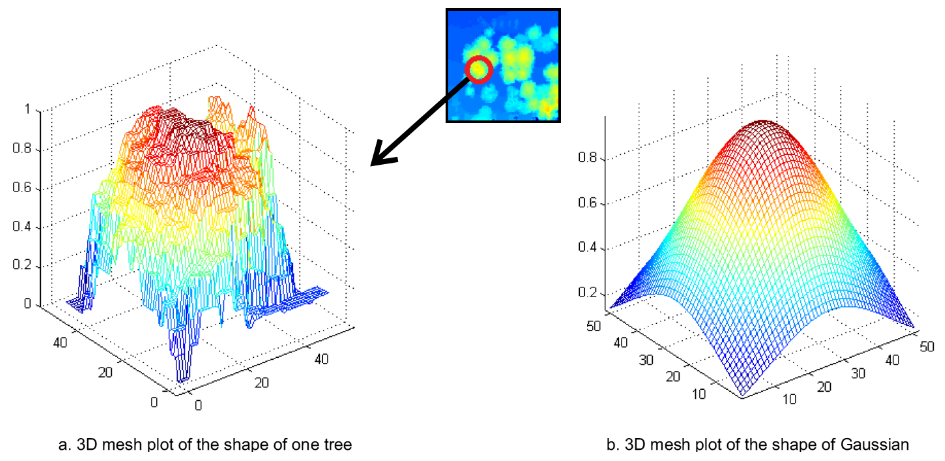


Figure 3.8: Comparison of the shape of a tree in the scene and the shape of a Gaussian kernel.

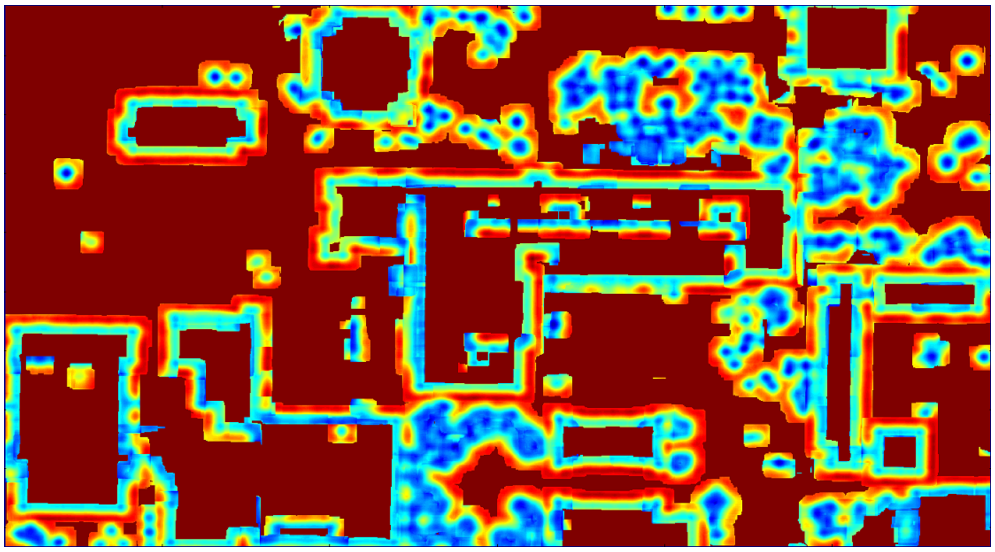


Figure 3.9: $\mathcal{T}$-map. In this map, tree areas are mainly colored in blue, which could provide a cue to assist segmenting them out. In the meanwhile, edges around rooftops are also somehow contaminated due to the limitation of this approach.

3.4.2 Vegetation Detection by Graph Cuts

In this section, a graph cuts based vegetation detection approach for point clouds is presented. The vegetation detection problem is transformed into the problem of finding a minimum cut in a graph whose nodes are the points in the point cloud. The problem at this stage can be simply regarded as a binary discrimination problem.

Graph Construction

The conventional way of dealing with 2D image segmentation using graph cuts is to construct a regular grid graph in which each node represents each individual pixel in the image, and the edges connecting each pair of adjacent nodes are equally weighted (Fig.2.3). The adjacency between two nodes is inherently defined. However, a LiDAR point cloud is an irregularly distributed point set. The points are unorganized in terms of spatial relationship between each other. Under these circumstances, a weighted graph, $\mathcal{G}$, containing all the 3D points, $\mathcal{V}$, from the input point cloud can also be created. Each node in the graph represents each individual 3D point. Each node is connected by its $k = 4$ nearest neighbors (Fig.3.10) obtained by K-NN search. Two imaginary terminal nodes (tree node $\mathcal{S}$ and non-tree node $\mathcal{T}$) are added. The strength of each connection is dependent on the Euclidean distance between the two end nodes. Closer nodes are supposed to be more strongly connected. The weight function can be defined in two forms. Either Eq.3.11 or Eq.3.12 is able to describe the connections between nodes adequately.

$$\mathcal{W} = s \cdot e^{-\frac{d^2}{\sigma_w^2}} \tag{3.11}$$

$$\mathcal{W} = s' \cdot \frac{1}{d} \tag{3.12}$$

where s or s' is a constant scaling factor, and σ_w can be set based on the average spacing of the testing points, and d is the Euclidean distance between two points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$ given by:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{3.13}$$

Energy Function

At this point, the graph has been reasonably constructed. The point set $\mathcal{V}$ will be separated into two disjoint sets belonging to either the terminal node $\mathcal{S}$ (tree) or the terminal node $\mathcal{T}$ (non-tree) by finding a minimum cut in the graph. Note that in the real world, most man-made building structures have flat or smooth rooftops. Also, in

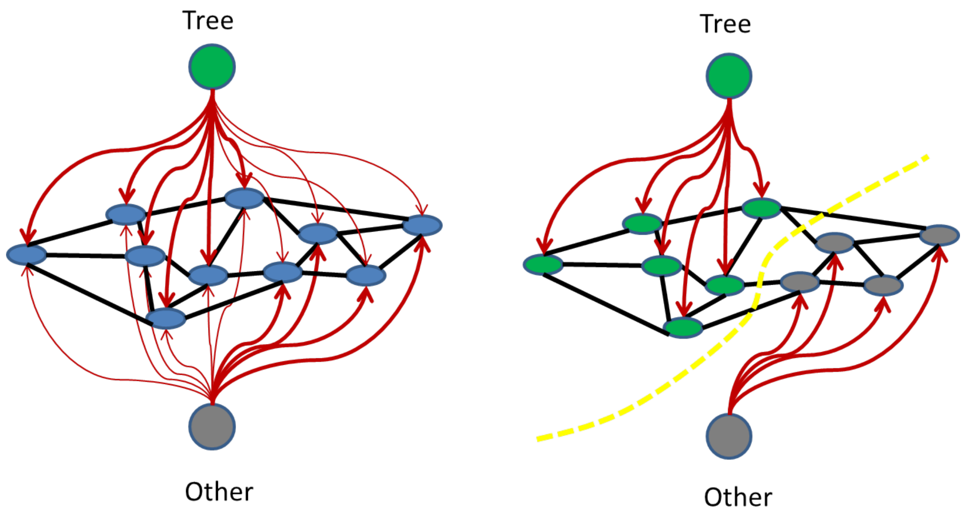


Figure 3.10: An non-grid graph constructed from a irregularly distributed point cloud. Each node is a 3D point in the point cloud. It is connected by its four nearest neighbors. The weights on the edges (black) are based on the Euclidean distances between them. A minimum cut is found in the right figure.

most urban areas, there are rarely abrupt changes to the ground plane. Speaking more formally, the ground surface varies gradually and smoothly across the whole scene, whereas, in vegetative areas, geometric irregularity plays a major role in describing the geometric surface of the vegetations, such as tree canopies. Within a small local surface region, presumably on a rooftop or the ground, all sampled points are supposed to have normal vectors whose directions are consistent, for the most part to each other. The degree of the geometric variation of the surface is expected to be as small as possible. The inverse expectation should be presumed when it comes to the vegetative areas where it's natural to have large surface irregularities. Hence, the variations of point normal vectors and the flatness or smoothness are supposed to be large as well. These properties of the implicit surface sampled by discrete points should be taken into account for the cut cost of a potential segmentation. The estimations of the variation of the normal vector distribution $\mathcal{F}_n$ and the surface flatness $\mathcal{F}_f$ are derived in Section.3.3. The energy function follows the generic form defined in Eq.2.10, Eq.2.11, and Eq.2.12. It is necessary to rewrite it here as:

$$E(l) = \sum_{p \in \mathcal{V}} D_p(l_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(l_p, l_q) \tag{3.14}$$

This form of energy function only depends on the pairwise interaction between neighboring points, and therefore the smoothness can only be designed to the first degree[8].

Performing Segmentation

The previous section discussed how the graph is made and what factors the energy function should take into account. The construction of the graph is adaptive to the density of the input point cloud. When the minimum cut is computed, the segmentation is assured to be smooth, which means neighboring points are very likely to be assigned to the same label. The developed segmentation can be operated with or without thresholding depending on how the data term is defined. Operating without thresholding achieves complete automation, whereas with thresholding requires some prior knowledge. However, they can both be considered as unsupervised strategies. Supervised methods ask users to pick some known points from the data and assign labels under complete confidence. This could be considered as hard constraints, on the contrary to, the soft constraints utilized by the energy function. This interaction increases the accuracy by sacrificing automation of the algorithm. Thresholding requires prior knowledge but not interaction because it is also involved in the energy function which only provides soft constraints. As matter of fact, the threshold could be estimated from the input data itself. Prior knowledge is not even required. Without loss of generality, let's assume labels are integers in $\mathcal{L} = \{0, 1\} = \{\mathcal{S}, \mathcal{T}\}$. Two types of definitions of the data term are given here corresponding to whether it is based on thresholding or not. So, the data term can be defined as:

$$D_p(l_p) = \begin{cases} \lambda & \text{if} \quad ((\mathcal{F}_n + \mathcal{F}_f) > t)\&(l_p == \mathcal{T}) \\ & \quad\;\; \mathit{or} \\ & \quad\;\; ((\mathcal{F}_n + \mathcal{F}_f) < t)\&(l_p == \mathcal{S}) \\ 0 & \text{if} \quad ((\mathcal{F}_n + \mathcal{F}_f) > t)\&(l_p == \mathcal{S}) \\ & \quad\;\; \mathit{or} \\ & \quad\;\; ((\mathcal{F}_n + \mathcal{F}_f) < t)\&(l_p == \mathcal{T}) \end{cases} \tag{3.15}$$

where λ is a constant value, and t is a threshold that is tunable. Or the data term could also be defined as:

$$D_p(l_p) = s_n \cdot e^{-\frac{\mathcal{F}_n^2}{\sigma_n^2}} + s_f \cdot e^{-\frac{\mathcal{F}_f^2}{\sigma_f^2}} \tag{3.16}$$

where s_n and s_f are two coefficients, and σ_n and σ_f are the standard deviations that can be statistically given based on the estimated values about curvatures and variations of normals in the whole scene. An example of the second type of data term is plotted in Fig.3.11.

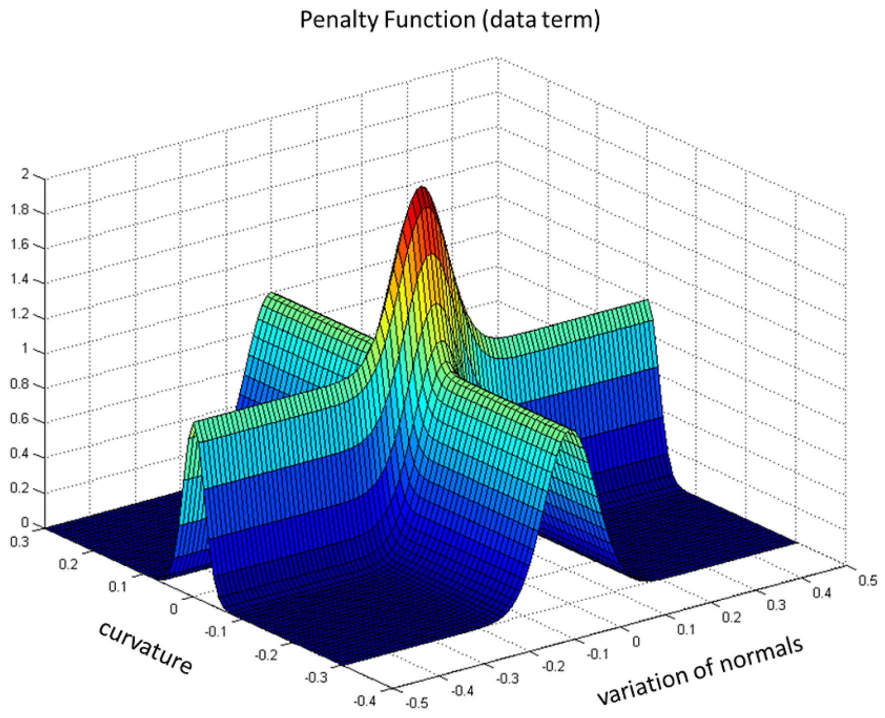


Figure 3.11: Plots of normalized penalty function (data term 3.16) of assigning a label to a node in the graph with some σ_n and σ_f values.

There are several choices for the smoothness term. A robust pairwise form based on Potts model [10] is chosen. Though energy optimization with Potts $V_{p,q}$ is NP-hard, graph cuts can still be used to find an answer that is within a factor of 2 from the optimum [10].

$$V_{p,q}(l_p, l_q) = s_v \cdot \delta(l_p, l_q) \tag{3.17}$$

$$\delta(l_p, l_q) = \begin{cases} 0 & if \quad l_p = l_q \\ 1 & if \quad l_p \neq l_q \end{cases} \tag{3.18}$$

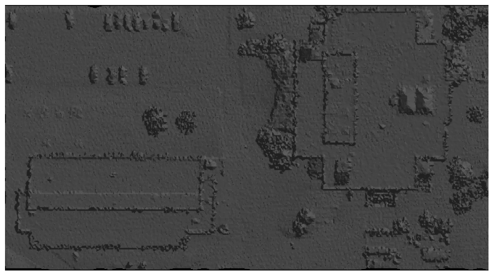
where p and q denote two points in the point cloud and l_p and l_q are either 0 or 1. s_v is a coefficient as well.

Both the swap move and the expansion move could be adopted providing equal segmentation results.

Experimental Results

The campus of the Rochester Institute of Technology is used as one of the major testing areas in this research. In this area, the vegetative portion is as significant in aerial coverage as the man-made objects. The height of trees canopies vary throughout scene. The ground elevation also shows significant variation. Point clouds with low density (about 4 pts/m^2) and high density (about 50 pts/m^2) are both available to be examined. Hence, while conducting the local analysis, the demanded number of neighboring points can not be fixed and depends on the density of the input point cloud. According to the experimental experience, the appropriate number of neighboring points would be as $2 \sim 3$ times as large as the point density.

Fig.3.12 shows the LiDAR image (about 3.5 pts/m^2) and the corresponding optical image for testing area NO.1, where the Chester F. Carlson Center for Imaging Science building is located on the R.I.T campus. Local analysis is conducted on the point cloud covering this area. In order to demonstrate how the size of the local neighborhood would affect the local analysis, the number of neighbors n is chosen as 10, 20, 30, and 40 for different groups of experiments. The number 10 is actually the appropriate choice for this data. Fig.3.13 shows the color coded normal maps using differently sized of local neighborhoods. The three elements $[n_x, n_y, n_z]$ composing a normal vector are represented by the values of RGB in three color channels.


(a) LiDAR Image (tessellated)

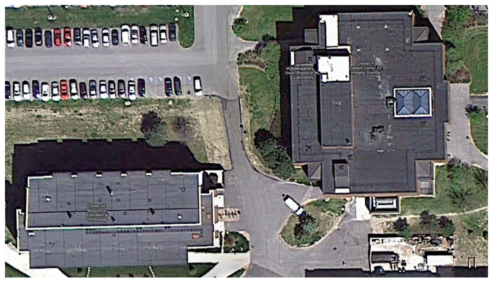

(b) Optical Image (courtesy to Google Map)

Figure 3.12: Testing area NO.1 on the northern part of R.I.T campus (including the CIS building).

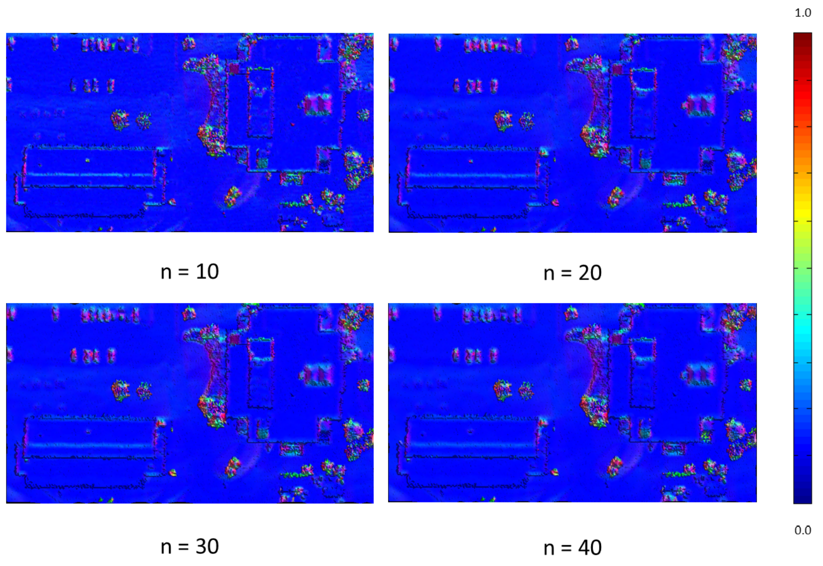


Figure 3.13: Comparison of color-coded normal map for increasing size n of local neighborhood (testing area NO.1).

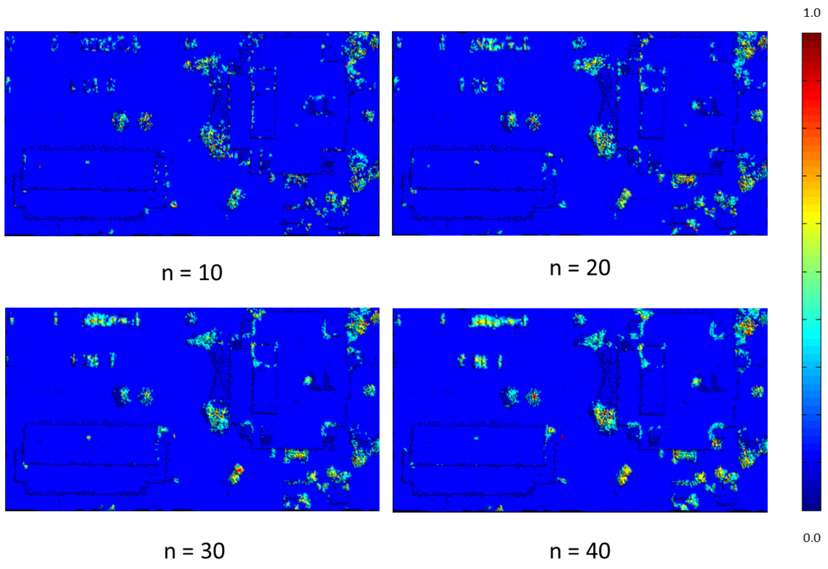


Figure 3.14: Comparison of color-coded surface flatness map for increasing size n of local neighborhood (testing area NO.1).

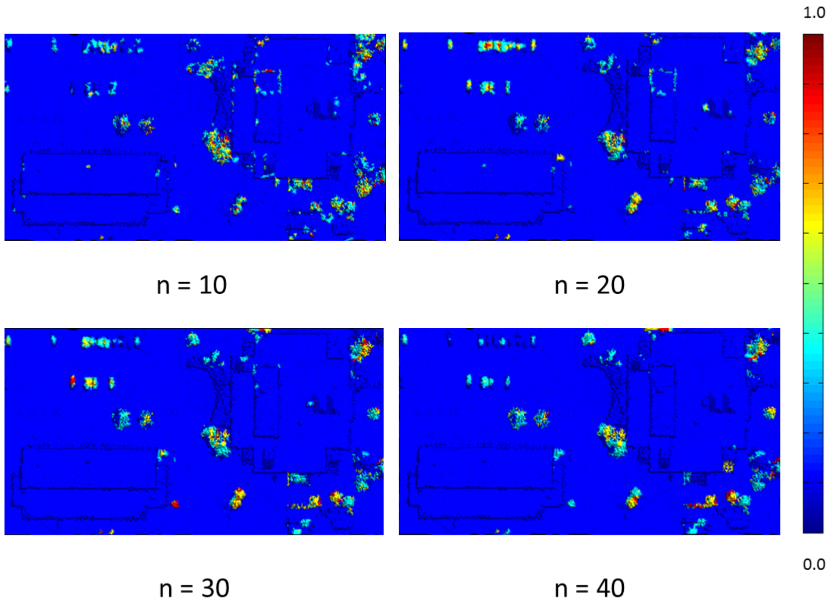


Figure 3.15: Comparison of color-coded normal distribution map for increasing size n of local neighborhood (testing area NO.1).

Fig.3.14 and Fig.3.15 show the corresponding surface flatness maps and surface normal distribution maps of testing area NO.1 using different sizes of local neighborhood. It is noticeable that regions covering trees, cars, and rooftop fences have much higher local flatness values and larger variation of the local surface normal distribution. These two different kinds of maps basically reflect the same property of the surface. In other words, areas with higher flatness values are also the areas with larger variation value of surface normal distribution, and vice versa. Fig.3.16 presents the vegetation result for testing area No.1 from a few of different visualization angles. The points colored in green are considered to be representing vegetation after detection. Since, at this stage, only two categories are to be identified, cars exhibiting as small clusters on the ground in the scene may also be or not be recognized as vegetations, which does not impose any significant negative influences on the future terrain and building footprints extraction.

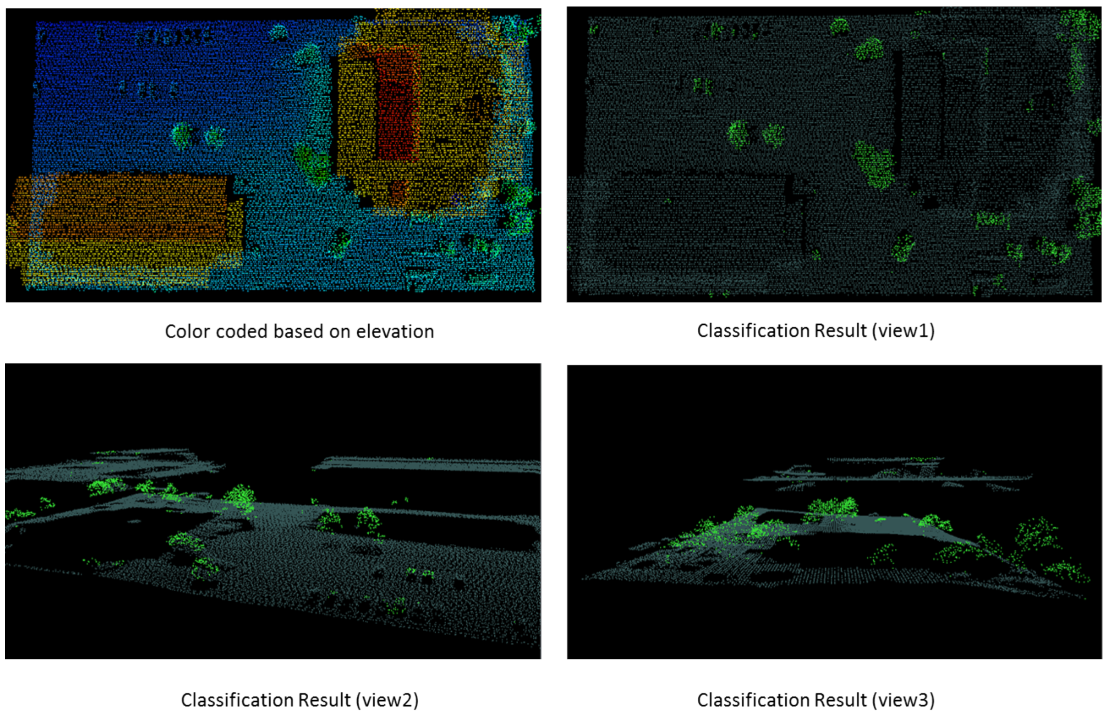
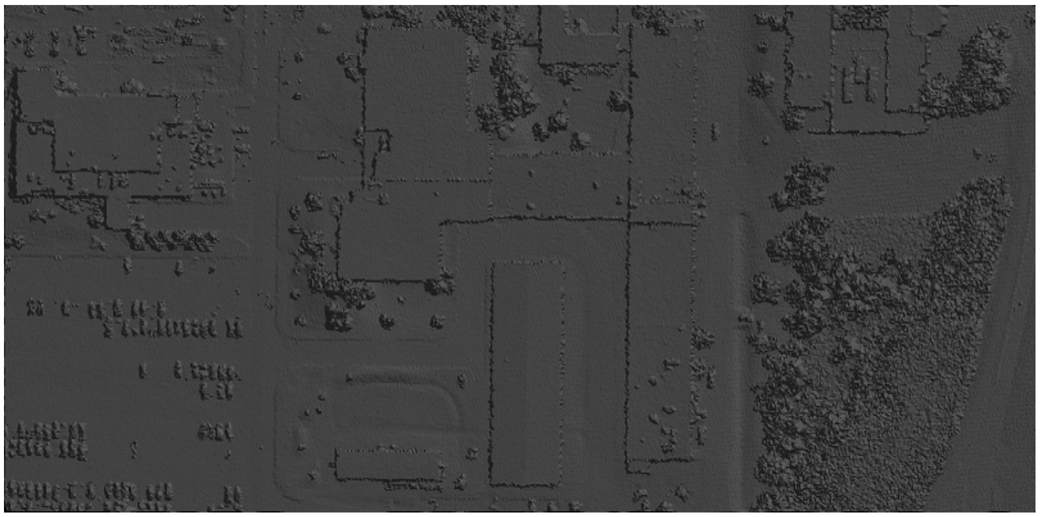
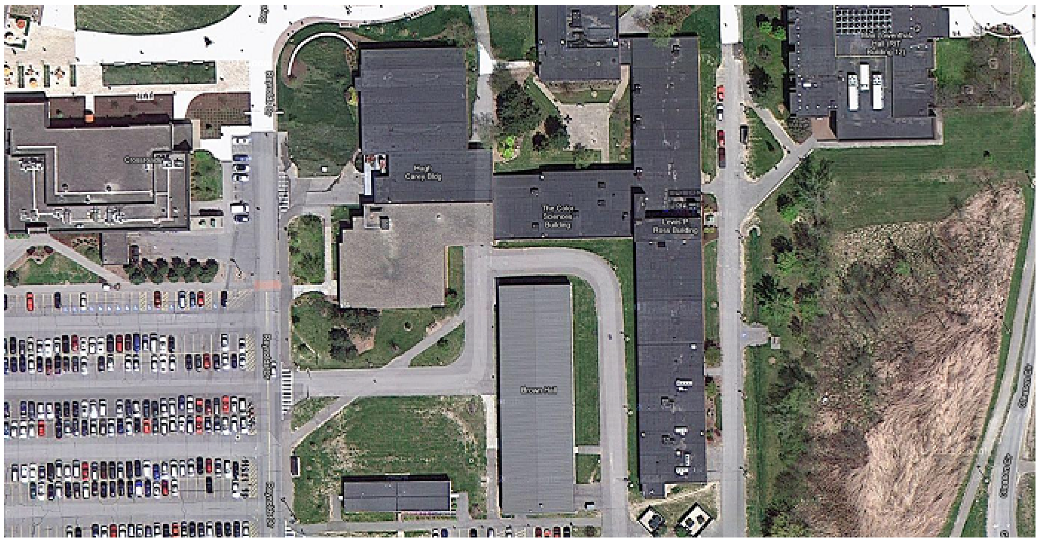

Color coded based on elevation
Classification Result (view1)
Classification Result (view2)
Classification Result (view3)

Figure 3.16: Vegetation detection result on testing area NO.1: the top left figure shows color-coded point cloud based on the elevation; the rest figures shows the separation of trees from the other content from three different views.


(a) 2D ortho-projected LiDAR Image (tessellated)



(b) Optical Image (courtesy to Google Map)

Figure 3.17: Testing area NO.2 on the southern part of R.I.T campus

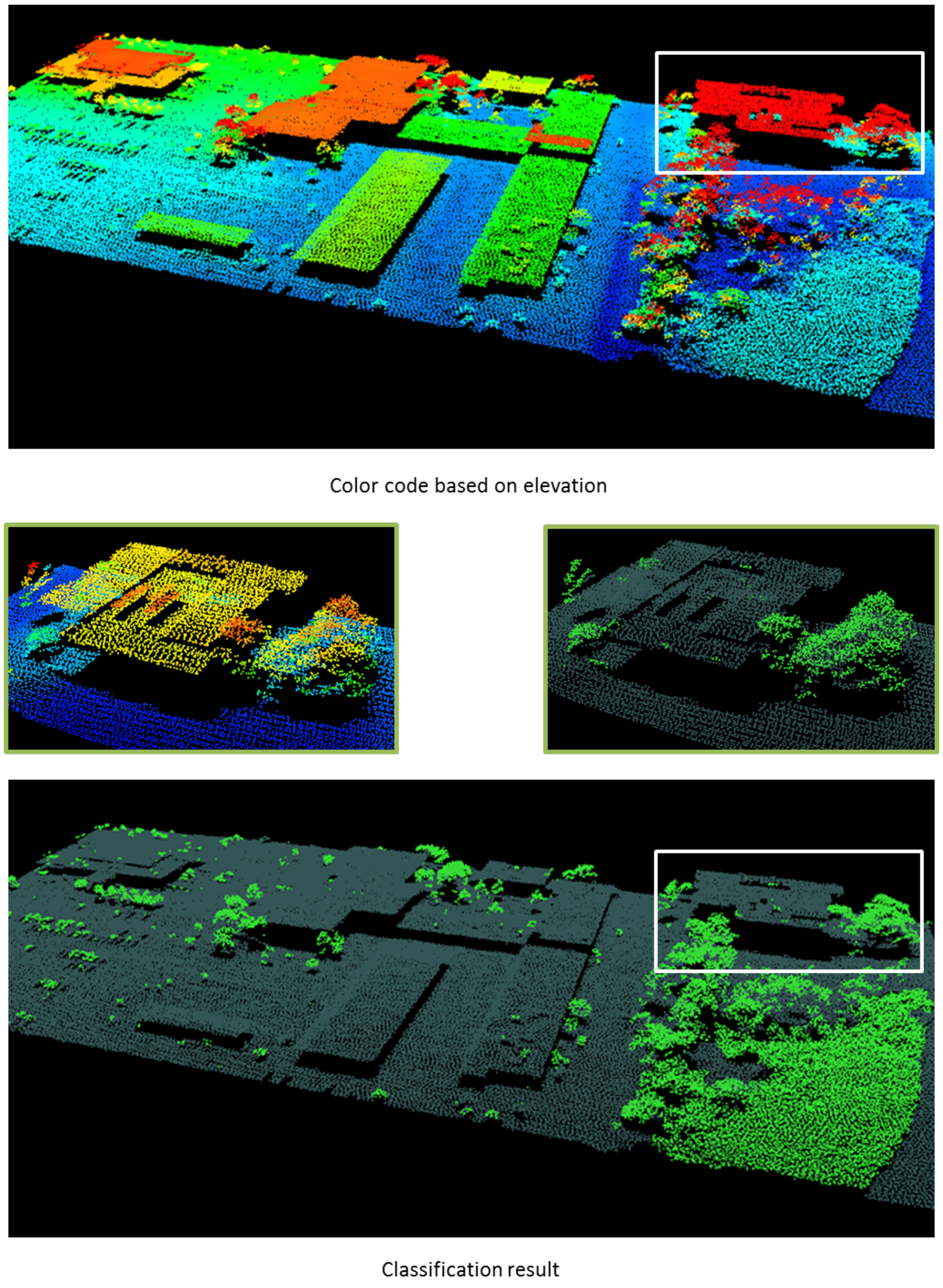
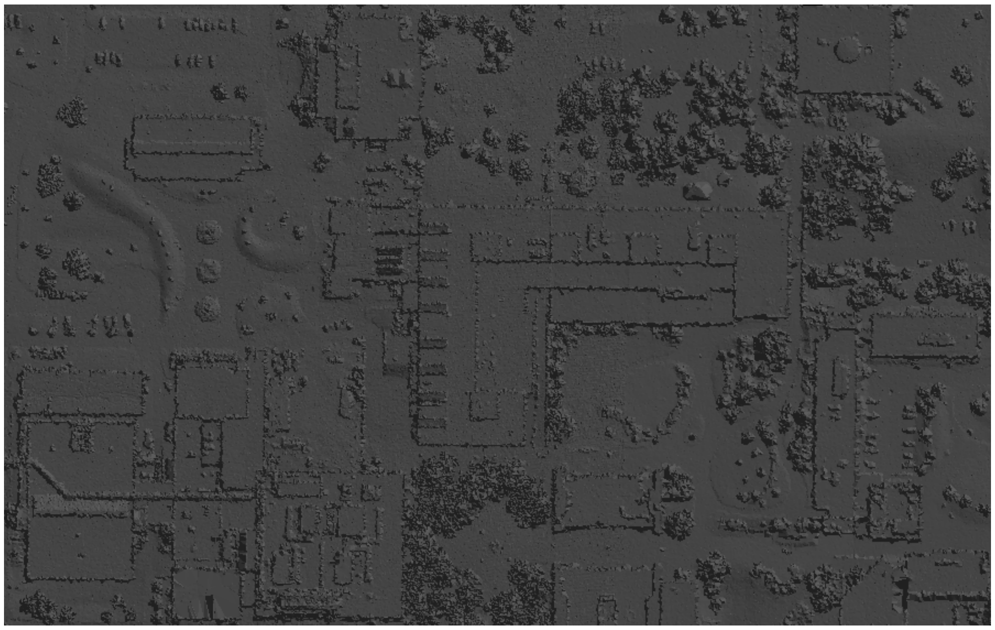
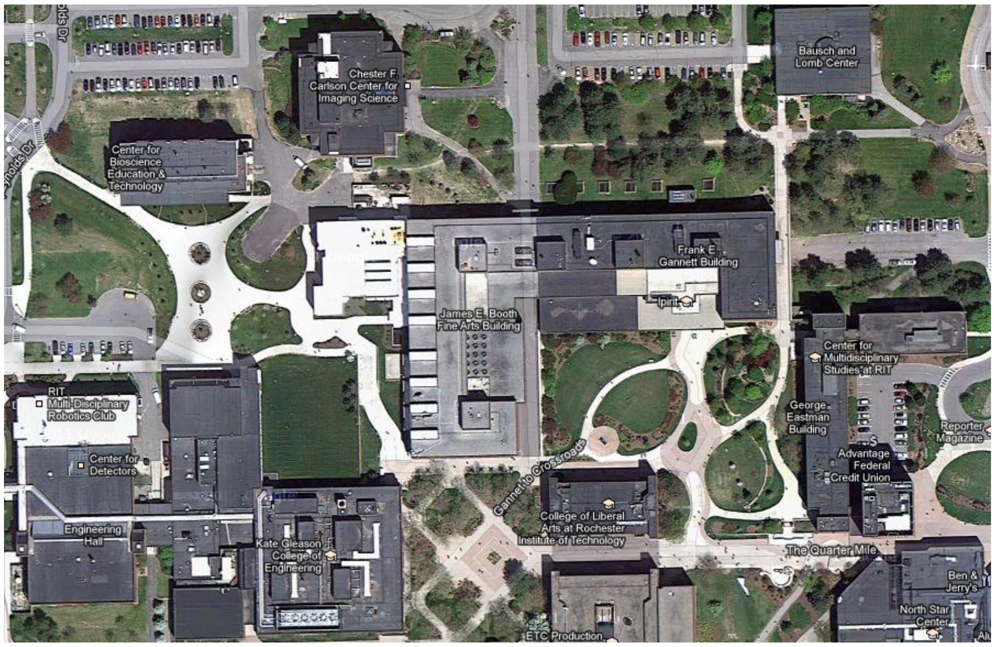


Figure 3.18: Vegetation detection result on testing area NO.2: the zoom-in region shows a good example that the method works effectively even sometimes the tree canopy blocks part of the rooftop.

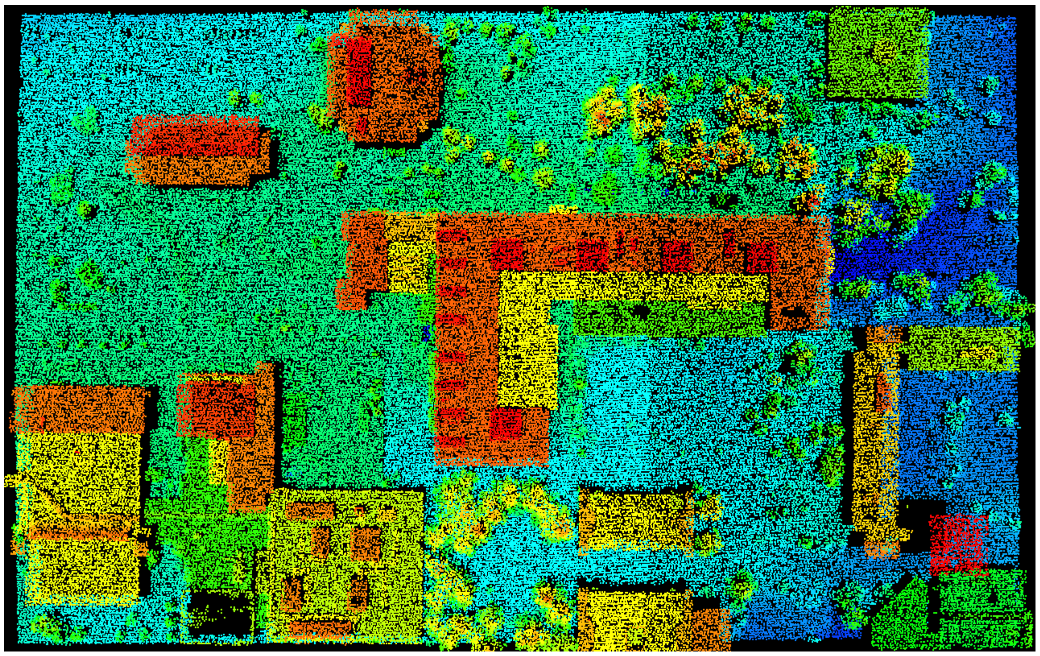


(a) 2D ortho-projected LiDAR Image (tessellated)

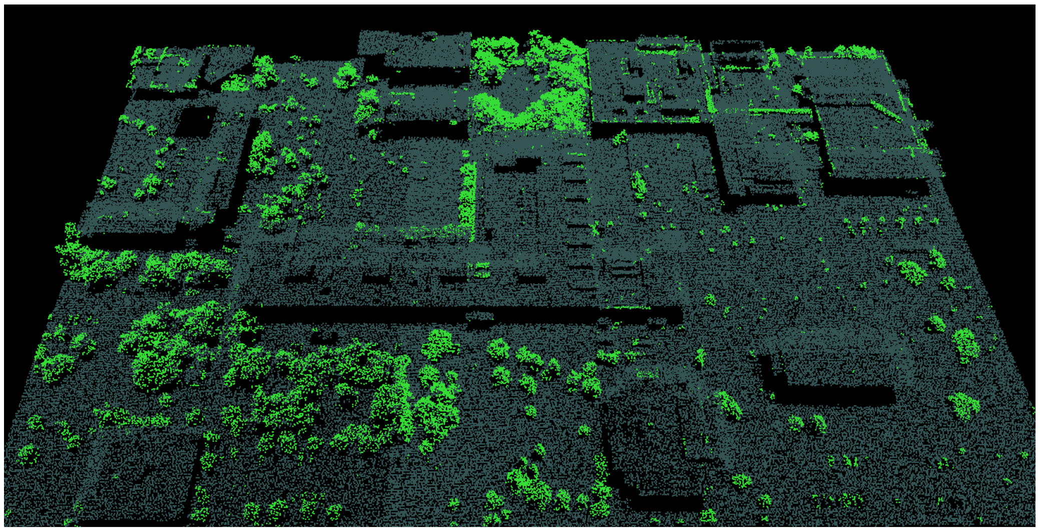

Chester F. Carlson Center for Imaging Science
Bausch and Lomb Center
Center for Bioscience Education & Technology
Frank E. Gannett Building
James E. Booth Fine Arts Building
Center for Multidisciplinary Studies at RIT
RIT Multi-Disciplinary Robotics Club
George Eastman Building
Reporter Magazine
Center for Detectors
Advantage Federal Credit Union
Gannett to Crossroads
Engineering Hall
College of Liberal Arts at Rochester Institute of Technology
Kate Gleason College of Engineering
The Quarter Mile
Ben & Jerry's
North Star Center
ETC Production

(b) Optical Image (courtesy to Google Map)

Figure 3.19: Testing area NO.3 on the whole northern part of R.I.T campus



(a) Color-coded point cloud based on the elevation.



(b) Separation of trees from the other content.

Figure 3.20: Vegetation detection result on testing area of the whole northern part of R.I.T campus. Bright green points are labeled as vegetations.

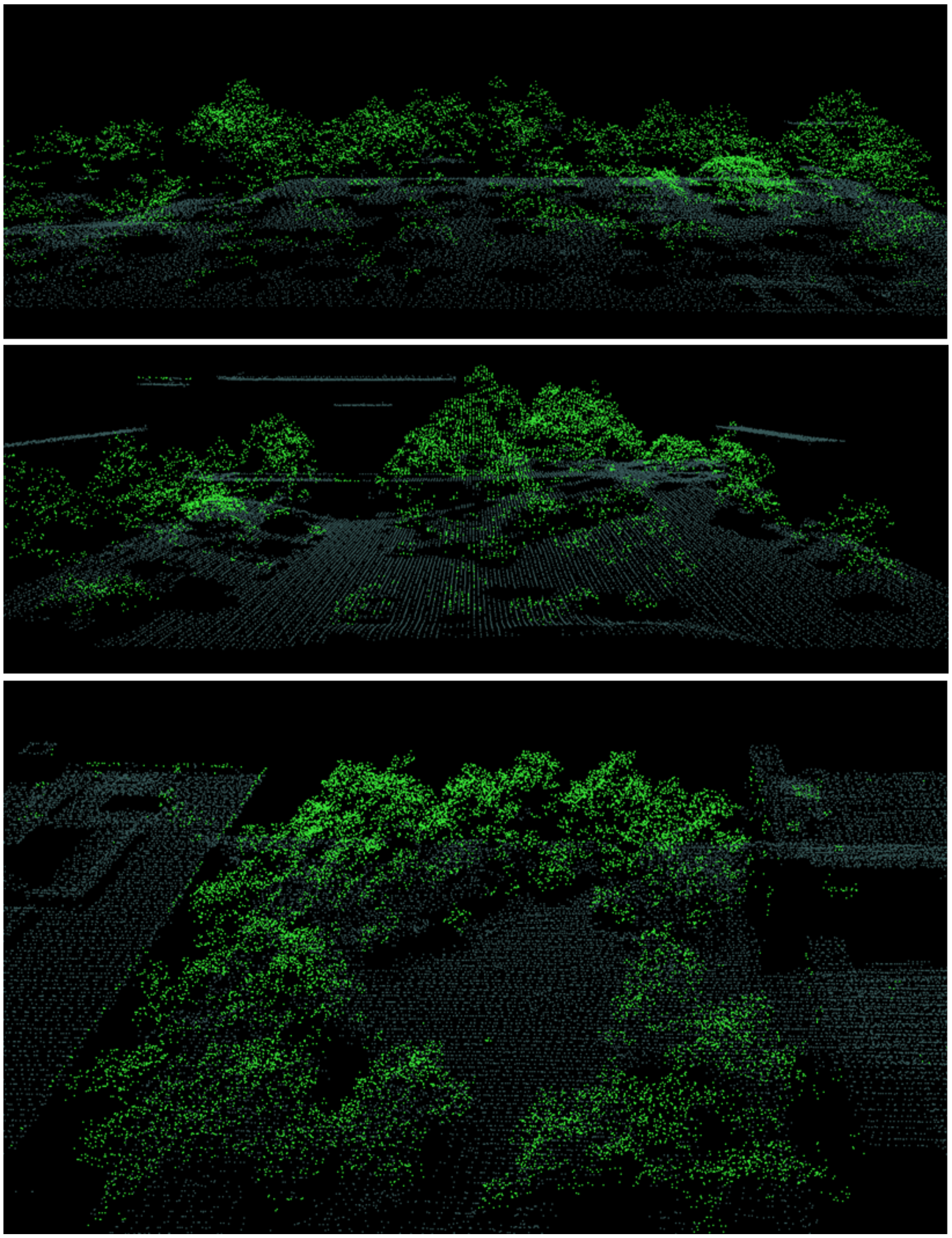


Figure 3.21: A few of zoom-in views of the regions in the whole northern part of RIT campus showing the details of vegetation extraction by the graph cuts algorithm.

Here are some more experimental results. Fig.3.17 shows the LiDAR image (about 3 pts/m^2) along with the corresponding optical image of testing area NO.2 on the southern part of RIT campus. Fig.3.18 shows the classification result of the testing area NO.2. It is exciting to see how the graph cuts based classification algorithm works effectively to separate trees from buildings even when the tree canopy blocks a portion of the rooftop.

Fig.3.19 shows the LiDAR image (about 3.5 pts/m^2) and the corresponding optical image of testing area NO.3 on a large area of the northern part of R.I.T campus. Fig.3.20 shows the classification result of the testing area NO.3. This area is quite a challenging testing area, because many trees are in contact with the buildings. However, the method here delivers a robust outcome. Fig.3.21 provides detailed views of the classification result. Points on trees (bright green points) are nicely identified and labeled for future removal.

An important issue with this algorithm needs to be pointed out. The problem is that relatively tiny point clusters, mainly in forms of roof fences, edge walls, air conditioning units, pipes and fans on the building rooftops are often grouped into the category of vegetations (Fig.3.22). The cause of this is also clear. The sampling process always makes these kinds of features lack regularized geometric properties. These features are much less important for the future process of rooftop description, therefore, it is a tolerable issue and the existence of this kind false alarm is tolerated.

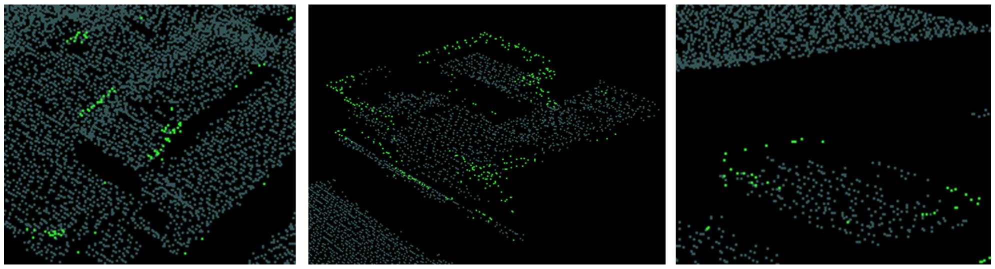


Figure 3.22: False alarms of the graph cuts based classification result.

Another relevant issue that also needs to be addressed here is the LiDAR image and the optical image were not captured at the same time. Therefore, scene content in both images is not identical. For example, in testing area NO.2, a large eastern part of the scene was covered by vegetations when the LiDAR image was taken, whereas, it was almost baren when the optical image was captured. The purpose of showing corresponding optical images is to better present what the testing area looked like from a more familiar point of view for most readers.

3.5 Terrain and Building Footprints Extraction

Most vegetative area, including unidentified tiny clusters can be successfully removed by the classification methods presented in the previous section. The remaining portions of the scene presumably consist of the terrain and all man-made building structures. At this moment, there is still no knowledge of how to recognize which part belongs to the terrain and which parts are individual buildings. However, this recognition is very critical to the upcoming processing. Many current approaches to urban modeling treat the terrain part as a giant flat surface, which does not account for one important characteristic of the terrain, the natural variation in elevation values. With regards to this fact, extracting the terrain from the scene is not a trivial step that can be neglected. Furthermore, building footprints detection must also be accomplished. Each building structure can be considered as an important cluster with a significant size on the ground. Different from point clouds produced by multi-view or stereo matching, airborne LiDAR point clouds mainly cover the building rooftops. Much fewer points actually fall on the building sides, and a lot of them might be eliminated through the step of noise removal. Because of this feature, the ground can be assumed to be disconnected from the building footprints. All these inherent facts can help to identify the terrain and the building footprints. The next step is to detect terrain and extract all possible rooftop patches from the scene which has already had vegetation excluded.

3.5.1 Euclidean Based Clustering

A clustering method is chosen as the approach to conduct the extraction of terrain and building footprint features. The purpose of this clustering is to divide an unorganized point cloud into smaller parts each of which is supposed be a meaningful representation of an object. It is also a spatial decomposition problem which uses Euclidean distance as the metric. This spatial metric measures the membership to different clusters. The details of this approach are depicted in Algorithm.3.5.1. Let $\mathcal{C}$ be a list of clusters. $\mathcal{Q}$ is a queue of points that waits to be checked. N^k is a set of points that are neighbors of the currently examined point p by radius searching. The algorithm terminates after every point p in the input point cloud, having been processed and assigned to any cluster in the list $\mathcal{C}$.

Algorithm 3.5.1: EUCLIDEANCLUSTERING($\mathcal{P}, radius$)

```
comment: The general steps of Euclidean distance based clustering.
$\mathcal{C} \leftarrow \emptyset; \mathcal{Q} \leftarrow \emptyset;$
for each $p \in \mathcal{P}$
  do if $!IsProcessed(p)$
    then
      $Insert(\mathcal{Q}, p);$
      for each $p \in \mathcal{Q}$
        do
          $N^k \leftarrow RadiusNeighborSearch(p, radius);$
          do
            for each $p \in N^k$
              do
                if $!IsProcessed(p)$
                  then $Insert(\mathcal{Q}, p);$
      if $IsProcessed(\mathcal{Q})$
        then
          $Insert(\mathcal{C}, \mathcal{Q});$
          $\mathcal{Q} \leftarrow \emptyset$
return $(\mathcal{C})$
```

As long as various targets in the scene are spatially separated from one another, it is reasonable to apply the Euclidean based clustering to group and identify points that make up these targets.

3.5.2 Two-step Hierarchical Euclidean Clustering

A strategy referred to as two-step hierarchical Euclidean clustering (Fig.3.23) was used. Two consecutive passes, utilizing two different searching radii in the neighborhood, are conducted.

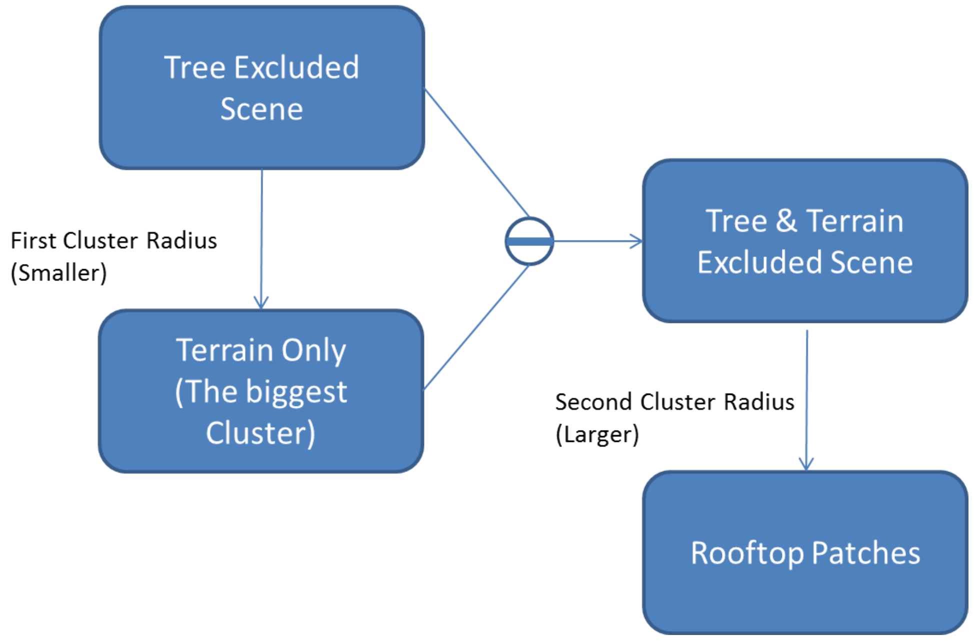

Tree Excluded Scene
First Cluster Radius (Smaller)
Terrain Only (The biggest Cluster)
Tree & Terrain Excluded Scene
Second Cluster Radius (Larger)
Rooftop Patches

Figure 3.23: Flow of the two-step hierarchical Euclidean clustering. The special operator refers to the operation of subtracting the terrain part from the input tree excluded scene.

The idea is to divide the points into a number of clusters based upon their spatial relationship - Euclidean distance. Points that stay closely in the space are considered as belonging to the same cluster. It is reasonable to say the terrain represents the largest areal coverage in most urban scenes. The first step attempts to separate this largest area from the rest of scene elements (the buildings). Due to the nature of airborne LiDAR data, the terrain portion of the scene will have very little contact with the buildings, or other man-made objects, and can be successfully isolated and removed based on Euclidean clustering with a reasonable radius value (depicted on the left in Fig.3.23). Once the terrain has been extracted, the remaining features, primarily buildings, can be also clustered by using a larger radius value than the one used in the first clustering pass (depicted on the right in Fig.3.23). The choice of these two values depends on the density of the urban area. If buildings distribute relatively sparsely, the two values are very different. Otherwise, the two radii can adopt very similar magnitudes.

3.5.3 Example Results

In the examples shown below, a scene consisting of the northern part of R.I.T campus is examined. Assuming all building objects are at least 1 meter above the ground and the distance between building footprints is larger than 5 meters, the first cluster radius is 1, and the second cluster radius is 5 (unit: meter). In Fig.3.24, the scene has been classified into two categories. The green points are identified as tree areas based on the graph cuts algorithm. Once the detected vegetation are removed from the scene, the result is shown in Fig.3.25. At this stage, it is ready to apply the two-step hierarchical Euclidean clustering method. After the first pass, the terrain is recognized as the largest cluster (Fig.3.26). The remaining clusters are put through the second pass. The resulting building footprints are shown in Fig.3.27, and Fig.3.28 provides a complete view of the result of this two-step hierarchical Euclidean clustering process.

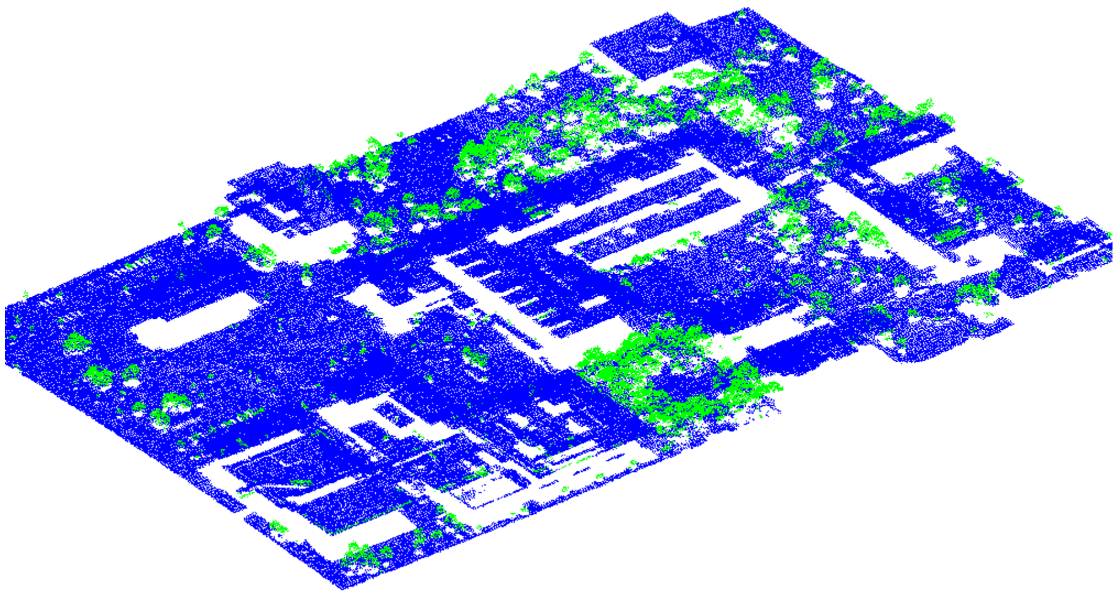


Figure 3.24: A scene of the northern part of R.I.T campus, and the vegetative areas are detected based on the graph cuts algorithm.

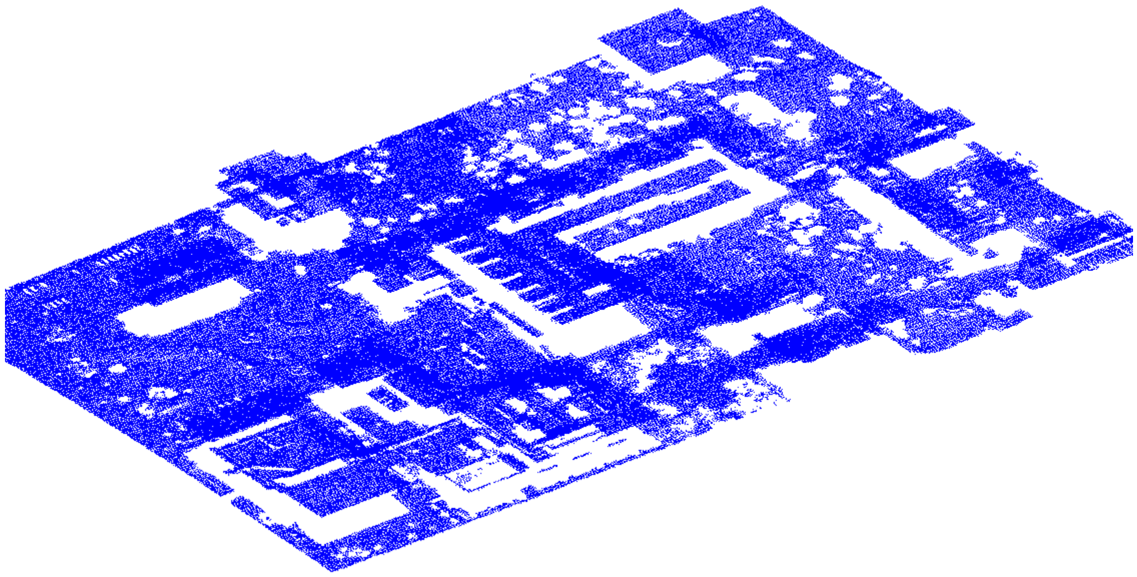


Figure 3.25: A scene of the northern part of R.I.T campus, and the vegetative areas are removed and only the ground and buildings are left.

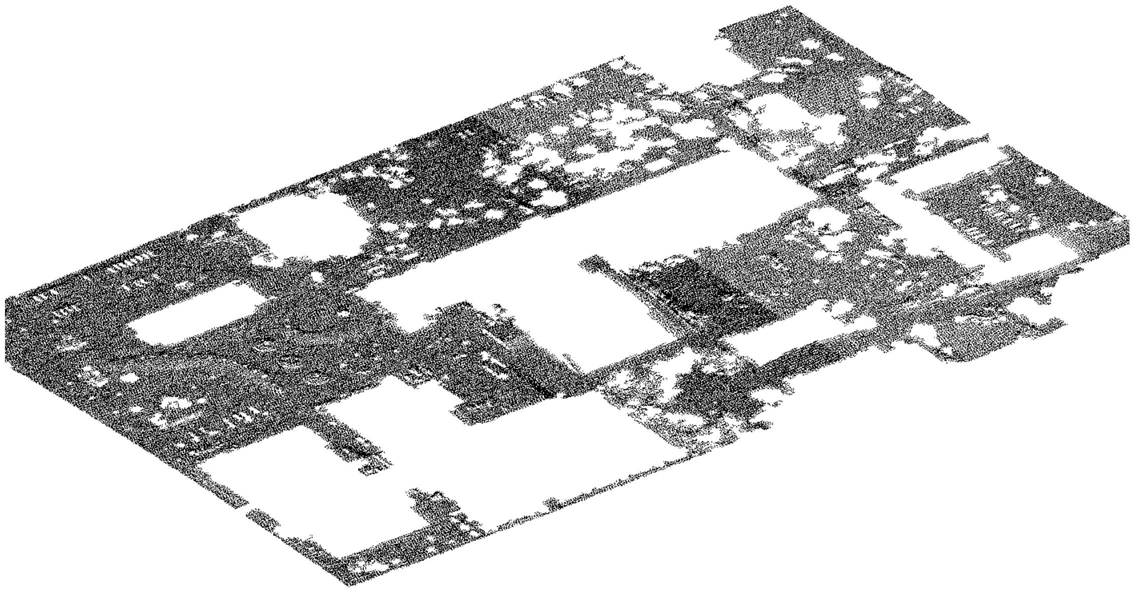


Figure 3.26: The terrain part of the northern part of R.I.T campus. It is obtained by the first step of the two-step hierarchical Euclidean clustering.

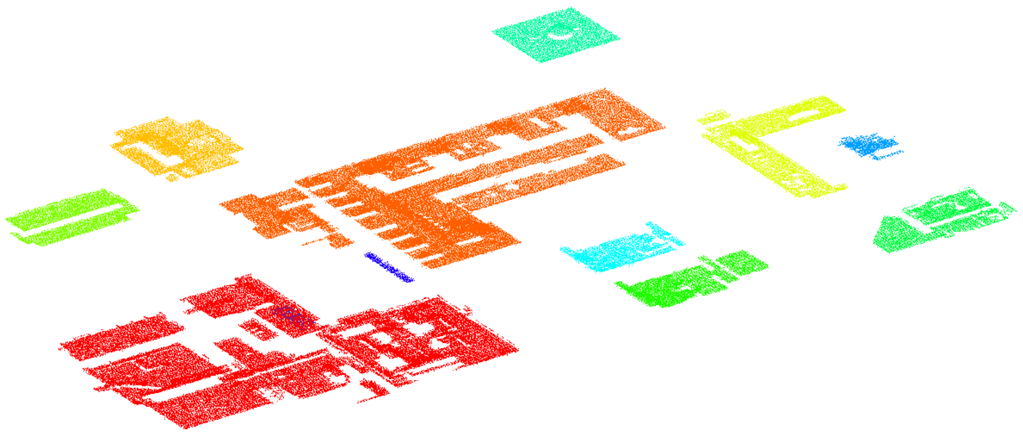


Figure 3.27: All significant building footprints in the scene of the northern part of R.I.T campus. They are obtained by the second step of the two-step hierarchical Euclidean clustering.

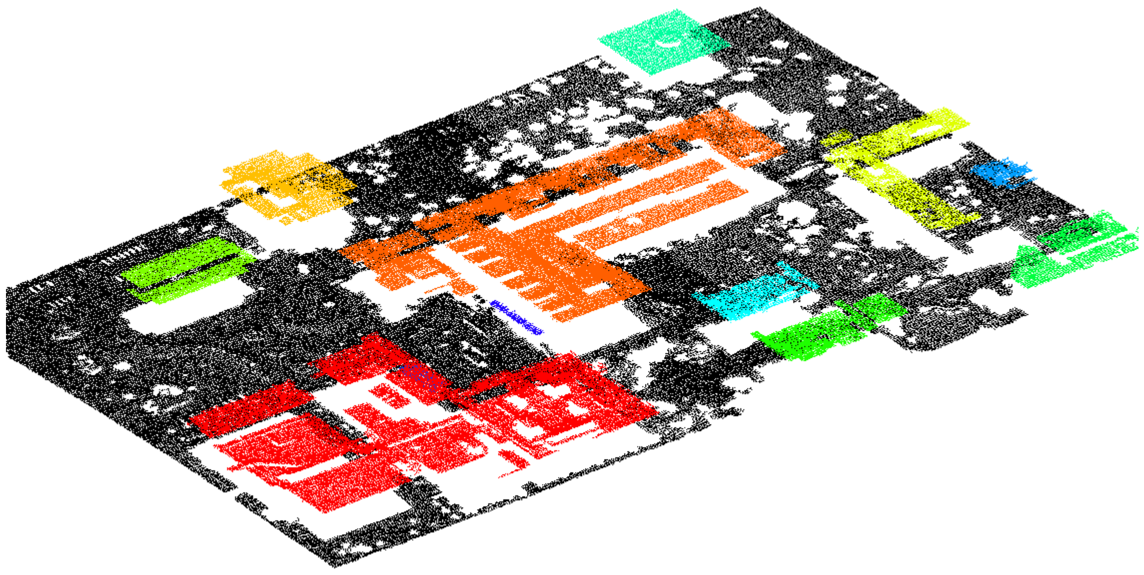


Figure 3.28: A scene of the northern part of R.I.T campus (include both terrain and building footprints).

3.6 Building Rooftop Description

Prior to this point, the whole point cloud was taken as input. The point cloud was put through the process of noise removal, vegetation detection, and terrain and building footprints extraction. The input point cloud has been divided into separate parts, such as the terrain portion, the vegetative areas, and individual rooftop footprints. Processing each individual cluster of building points is now the major task. After obtaining all of the desired building rooftop patches, the processing unit is reduced to each individual rooftop. All building footprints (clusters) can be processed in parallel utilizing appropriate resources. It is very important to have a reasonable and fine description of the rooftop from the raw points. Generally speaking, rooftop description is predictable since in most urban areas, the design of rooftops follows certain rules, such as common planar surfaces, sharp corners, straight linear edges, and so on. But at the same time, it is also unpredictable because there is a huge variety in rooftop shapes. A rooftop can have multiple layers or consist of many significant features. This contradiction always makes describing a rooftop a challenging task to conduct. Fig.3.29 shows the variety from one roof to another in terms of the complexity, even with the relatively few numbers of buildings that exist on a college campus compared to a urban downtown area.

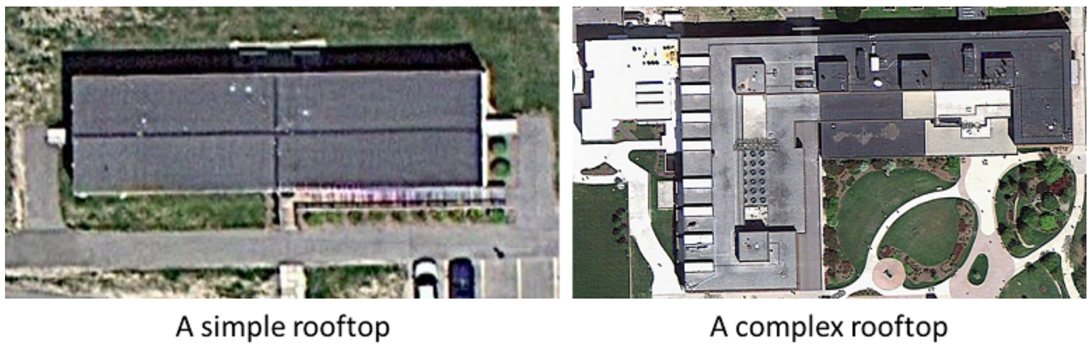


Figure 3.29: Two examples of rooftops with different complexities (courtesy to Google Map). A simple one could have only one flat surface. A complex one could have a complicated major outline and many other parts on the roof as significant features of itself.

3.6.1 Rooftop Feature Detection

At this stage, one is ready to produce mesh models from the individual point sets that represent building footprints. However, in order to achieve constructed models containing fine details, it is necessary to identify those significant features that are parts of the rooftop and describe them in a complete way. Since there is no prior knowledge on what kinds of geometric primitives that consist of the rooftop, template fitting is not an option. The only information available is the raw points and their estimated normal vectors and curvature values. It is reasonable to segment different features of a rooftop by comparing the similarities between normal vectors and curvatures within a local neighborhood. A group of neighboring points probably belong to the same feature of the rooftop, if they have normal vectors with very similar directions and their curvatures do not vary significantly within the neighborhood. So, this detection process could be interpreted as a region growing based segmentation problem. Rabbani *et al*[75] proposed a segmentation method using a smoothness constraint which avoids calculating properties like curvatures. Instead, they calculated the residual value obtained by plane fitting to a small surface area and utilized it as a substitution to the curvature property. Their method was only tested on surfaces of indoor objects exhibiting small size, not on large surfaces like rooftops. The algorithm presented here is similar. The smoothness constraint is maintained, however, the curvature property is also directly explored. The presented approach works specifically for building rooftops.

The region growing segmentation process takes a raw unorganized 3D point cloud of any building rooftop and uses their estimated point normals and estimated curvatures as well, in accordance with some predefined values as thresholds, to group points that belong to the same region. The process for estimating the normal vectors and surface curvatures has been already introduced in Section.3.3. This region growing process examines the local connectivity and the consistency of normals. Specifically, local connectivity is a constraint saying that the points belonging to the same surface or in the same segment should be locally close to each other even though they are not directly next to each other. It is naturally enforced since nearest neighbor searching is utilized here to form a local neighborhood. It guarantees these points are spatially related. Furthermore, the surface belonging to a roof feature should be locally smooth, which can be considered as another constraint. In many cases of urban rooftops, it is also reasonable to claim that the surface should be planar. There is no requirement to specify the number of expected roof features. It is completely data driven and depends on the structure of the roof itself. By changing a small number of parameters, including thresholds for surface curvature and normal changes, the features of the rooftop can be revealed and eventually reconstructed to various levels of details. In other words, the simplicity or complexity of a modeled rooftop can be determined by users of the algorithm based upon the needs of different applications.

Algorithm 3.6.1: ROOFTOPSEGMENTATION($\mathcal{P}$, *normals*, *curvatures*)

```
comment: Segment a given point cloud of a rooftop into a group of roof features.
$\mathcal{R} \leftarrow buildEmptyList;$
$i \leftarrow 1$
while $!isEmpty(\mathcal{P})$
do
    $\mathcal{R}[i] \leftarrow \emptyset$
    $\mathcal{S}[i] \leftarrow \emptyset$
    $p_{min} \leftarrow getPointwtSmallestCurvature;$
    $insertPoint(\mathcal{R}[i], p_{min});$
    $insertPoint(\mathcal{S}[i], p_{min});$
    $removePoint(\mathcal{P}, p_{min});$
    for $k \leftarrow 1$ to $size(\mathcal{S}[i])$
    do
        $\mathcal{B}[i][k] \leftarrow knnSearch(\mathcal{S}[i][k]);$
        for $j \leftarrow 1$ to $size(\mathcal{B}[i][k])$
        do
            $p \leftarrow \mathcal{B}[i][k][j]$
            if $p \in \mathcal{P}$ and $angle(normal_p, normal_{\mathcal{S}[i][k]}) < \delta_{ang}$
            do
                $insertPoint(\mathcal{R}[i], p);$
                $removePoint(\mathcal{P}, p);$
                if $curvature(p) < \delta_c$
                  do $insertPoint(\mathcal{S}[i], p);$
    $i \leftarrow i + 1$
return $(\mathcal{R})$
```

The detail of the steps is given in Algorithm.3.6.1. In this piece of pseudo code, $\mathcal{P}$ is the input point cloud or point list. $\mathcal{R}$ is a list of regions detected by the segmentation process. $\mathcal{S}[i]$ is a list of initial feature seeds when the algorithm is in the ith round. The seed points are selected based on their curvature values instead of in a random fashion. Every time, the curvature value of one point is examined and the point will be put into the seed list if its estimated curvature is the smallest in the point list or small enough. It guarantees the smoothness property acts as a constraint to make region be a smooth surface, or a planar surface under many situations.

This algorithm is also depicted in the flowchart shown in Fig.3.30. The algorithm first picks a point with the smallest curvature value as the starting seed point. Within a small spatial neighborhood of this seed point, it compares the direction of the normal vector of any other point with the normal direction of the current seed point. If the directional difference (angle between two vectors) is larger than a predetermined threshold (e.g. 5°), the point being examined does not belong to the group initiated by the seed point, otherwise, it does. Of those points which have been grouped together by the seed point, points with curvature values lower than another predetermined threshold (e.g. 0.2) are chosen as new seed points. The procedure continues in an iterative fashion and stops when all points are visited. During the growing process, the curvature property helps to grow the region and group points which are supposed to belong to the same region. This approach can successfully segment the major regions that consist of a complete rooftop. For each segmented region, RANSAC is applied to fit a particular type of surface (most are planar surfaces) from the candidate points.

Though the normal and curvature parameters are estimated as closely as possible, they are still represented only by their best approximation. It is inevitable that over-segmentation occurs due to the nature of the problem. A great number of tiny regions in which there are only a few points, or even only one point, are generated. Another refinement step needs to occur to merge or combine these tiny regions with the major regions to which they likely belong. For each undetermined point, the distance d_1 to each estimated surface by RANSAC and the distance d_2 to each major region are calculated. After normalization, d_1 and d_2 are combined (Eq.3.19) and used as a metric to determine which major region the target point in the tiny region belongs to.

$$d = \rho_1 d_1 + \rho_2 d_2 \tag{3.19}$$

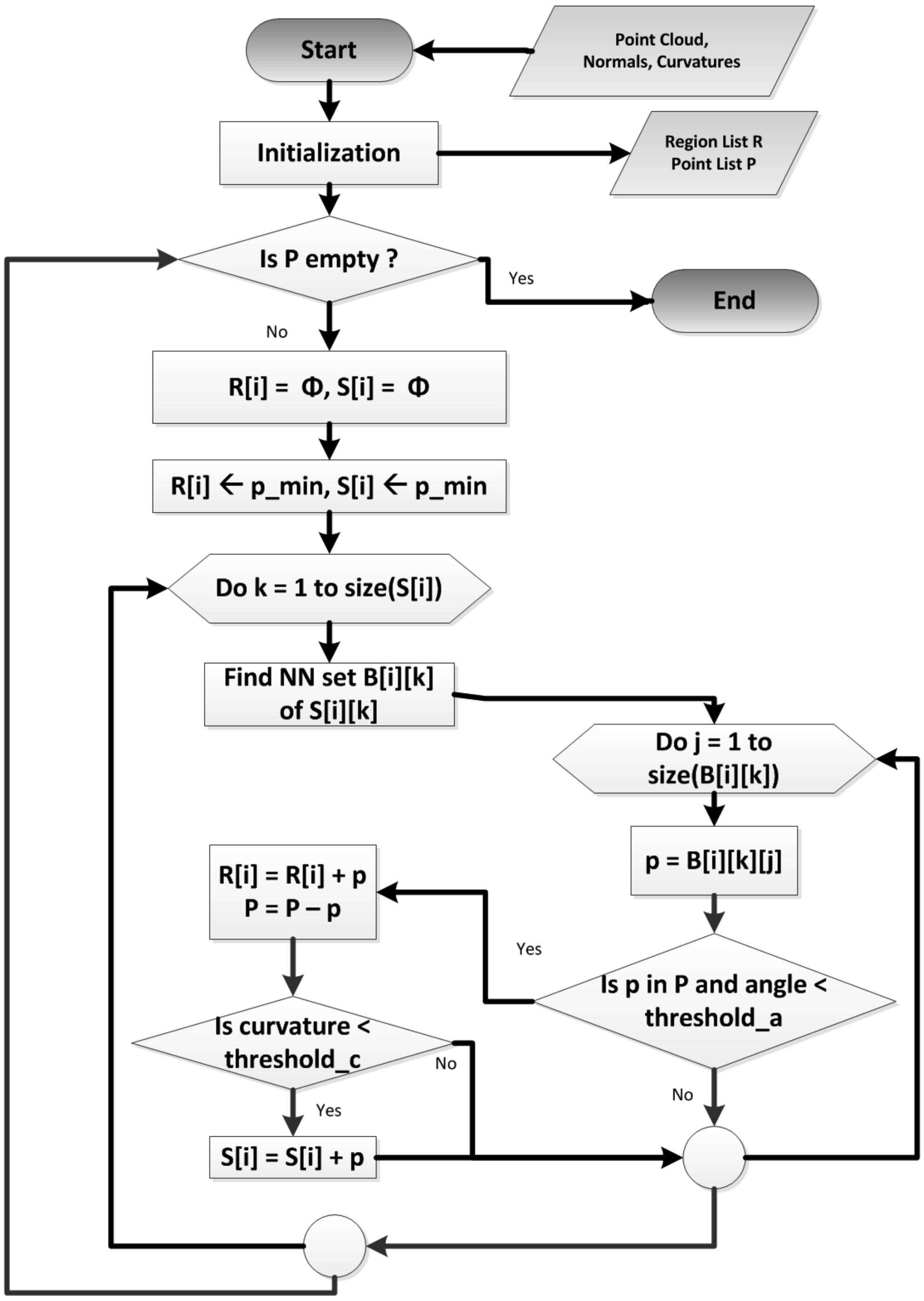
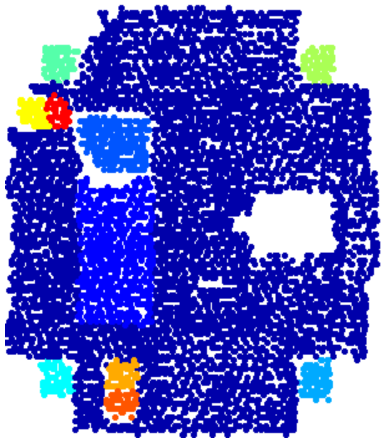

Start
Point Cloud, Normals, Curvatures
Initialization
Region List R
Point List P
Is P empty ?
Yes
End
No
R[i] = Φ, S[i] = Φ
R[i] ← p_min, S[i] ← p_min
Do k = 1 to size(S[i])
Find NN set B[i][k] of S[i][k]
Do j = 1 to size(B[i][k])
p = B[i][k][j]
R[i] = R[i] + p
P = P – p
Yes
Is p in P and angle < threshold_a
Is curvature < threshold_c
No
Yes
No
S[i] = S[i] + p

Figure 3.30: Algorithm flowchart of region growing using smoothness constraint and curvature consistency to detect features on the rooftop.

3.6.2 Example Results

Segmentation results of several different building rooftops are shown here. Major regions on the rooftop before and after absorbing minor features are presented. The profile views provide a perspective to illustrate how the plane fitting improves the flatness of the original planar features.


(a) top view

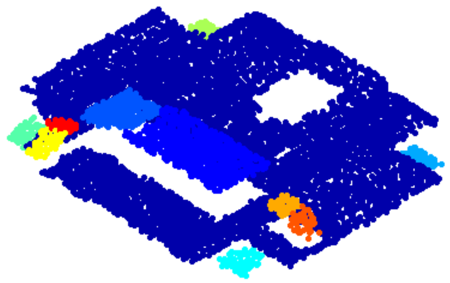
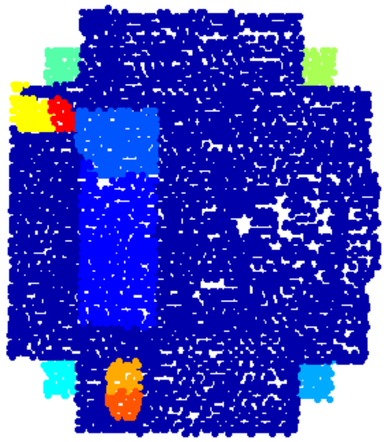

(b) slant view

Figure 3.31: Direct result from the region growing based segmentation (Roof NO.1).


(a) top view

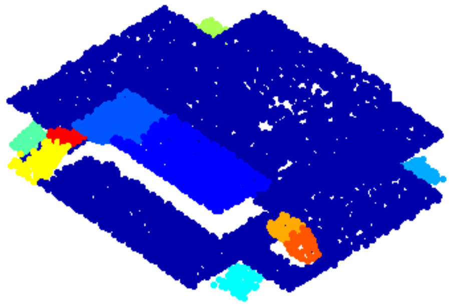
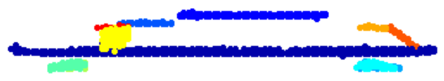

(b) slant view

Figure 3.32: After adding tiny regions back to major regions (Roof NO.1).


(a) Before fitting planes to points

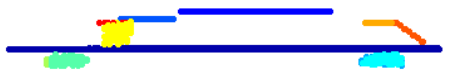
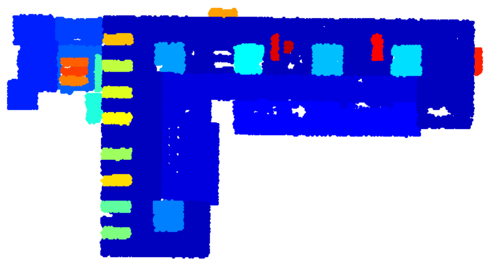

(b) After fitting planes to points

Figure 3.33: The profile view of the rooftop (Roof NO.1).


(a) top view

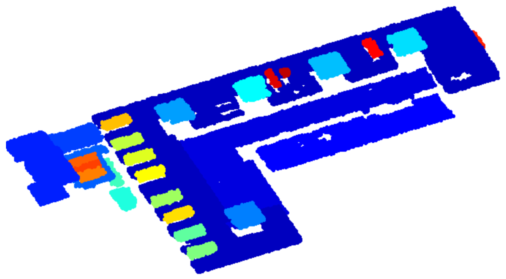
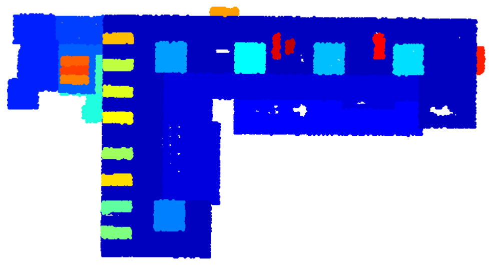

(b) slant view

Figure 3.34: Direct result from the region growing based segmentation (Roof NO.2).


(a) top view

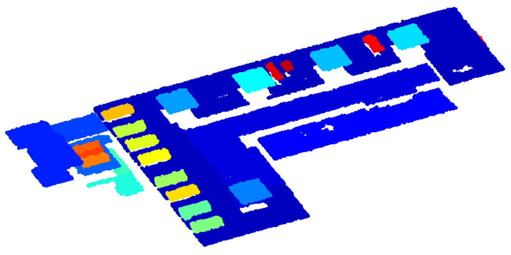
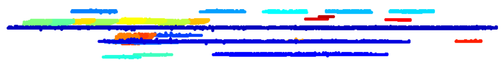

(b) slant view

Figure 3.35: After adding tiny regions back to major regions (Roof NO.2).


(a) Before fitting planes to points

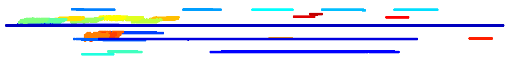
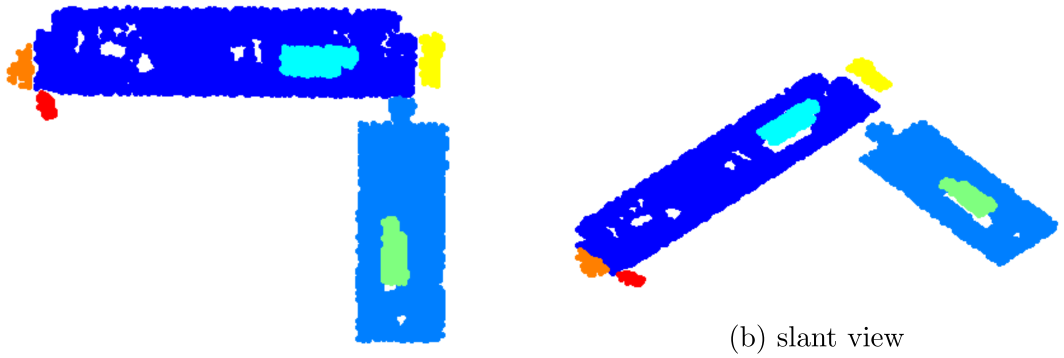
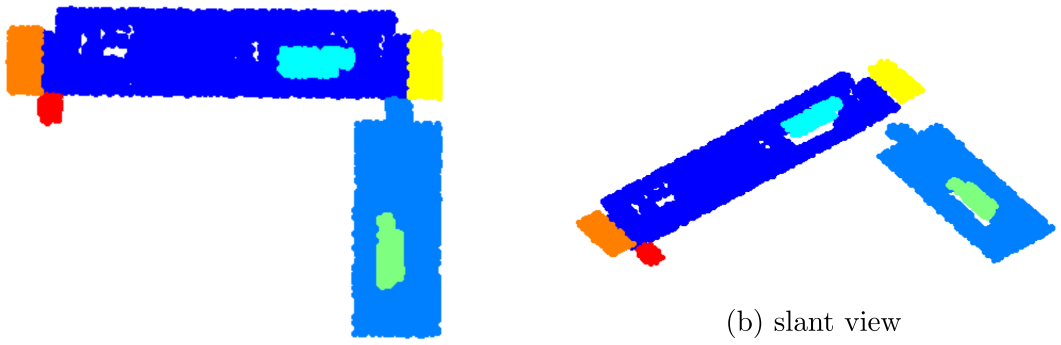

(b) After fitting planes to points

Figure 3.36: The profile view of the rooftop (Roof NO.2).



(a) top view (b) slant view

Figure 3.37: Direct result from the region growing based segmentation (Roof NO.3).



(a) top view (b) slant view

Figure 3.38: After adding tiny regions back to major regions (Roof NO.3).

(a) Before fitting planes to points (b) After fitting planes to points

Figure 3.39: The profile view of the rooftop (Roof NO.3).

3.6.3 Determination of Rooftop Orientation

In the previous section, a building rooftop was segmented into different parts. Each part represents an significant feature of the rooftop. For all planar surfaces, perfect plane models are estimated and all original surface points are realigned in accordance with the estimated planes. The next critical step is to find an appropriate and rigorous outline for each rooftop segment. It is not a very straightforward procedure since the sampling process of LiDAR often misses the points on the corners which are the most important hints for outline definition. Instead of directly moving on to the discussion of outline generation, determination of the orientation of each rooftop part is first discussed.

In this section, the problem domain is changed from $\mathcal{R}^3$ to $\mathcal{R}^2$. Each set of 3D points to be processed is ortho-projected onto a 2D plane, because it is easier to look at this problem from the two-dimensional perspective. The focus of the problem is directed at how to define and determine the principal orientation of a projected rooftop. The reason that the principal orientation needs to be examined is that it is the highly valuable information to assist in refining the outlines of the building rooftop features. One practical observation is the design of urban man-made structures often exhibit roof edges parallel to the rooftop principal orientation. The principal orientation of an object can be described by the principal axises of its minimum bounding box, particularly speaking, the major axis and the minor axis. Fig.3.40 illustrates a hypothetical roof part (top-view) with its principal axes and minimum bounding box depicted. Since the principal axes indicate the orientation of a building rooftop part, determining this principal orientation is equivalent to finding the principal axes. Furthermore, the principal axes can be easily derived if the minimum bounding box is known. Conversely, a corresponding bounding box is also straightforward to be defined if the two orthogonal axes are provided. The approach of using minimum bounding box to help determine the principal orientation is completely data-driven.

On the topic of bounding box fitting, Chaudhuri and Samal [76] introduce a simple approach for fitting a bounding rectangle to a closed regions. These closed regions can be convex or concave. A least square approach is used to determine the directions of major and minor axes of the object. These two axes indicate the principal orientation of the object. The four vertices of the bounding rectangle (box) are then computed by pairwise solving the intersections of four lines. They only take boundary points of the object to conduct the calculation. One major concern about this approach is whenever the shape of the test object is not symmetrical, such as L shape, the least square method lacks the capability to provide a robust result of principal axes. Besides, there is no formal mathematical proof to claim the resulting bounding box is the expected minimum bounding rectangle.

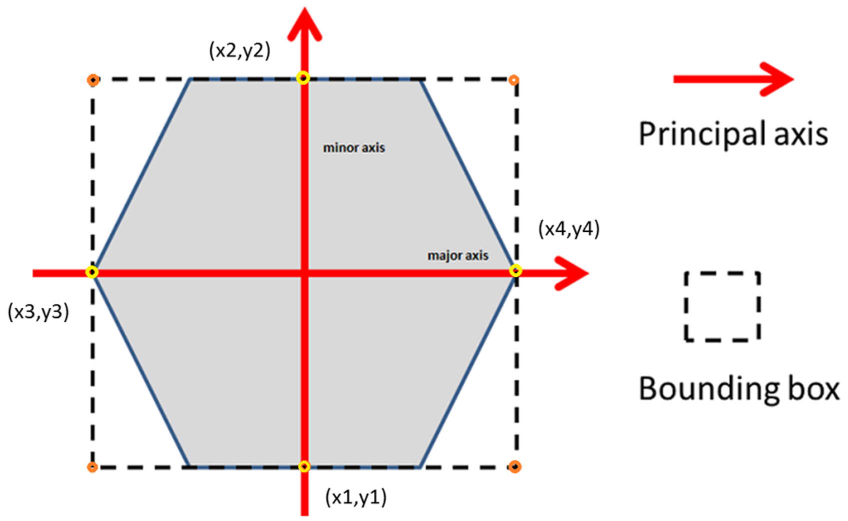


Figure 3.40: A hypothetical example with its principal axes and bounding box.

Determination of Bounding Box Given Arbitrary Orientation of Axes

It is difficult, in one step, to directly compute the accurate principal axes and the minimum bounding box of a target object. If some estimate of the orientation of the principal axes is given, the corresponding bounding box of the target object could be derived using the approach described in this section. Using this same calculation, and subsequent estimates, a group of corresponding bounding boxes can be obtained by testing a group of candidate orientations of the principal axes. Therefore, the minimum bounding box can be found among these candidate bounding boxes.

A bounding box can be defined if the angle, α, between the major axis and the x-axis is provided in an orthographically projected plane. Consider a 2D ortho-projection of 3D points of an isolated rooftop part, where $(x_i, y_i), i = 1, 2, ..., n$ are the 2D points of the projection. The centroid $(\bar{x}, \bar{y})$ is defined as:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.20}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.21}$$

The centroid is the center of the bounding box, and also the intersection of the major and minor axes (principal axes). An equation can be defined as below:

$$f = (y_i - \bar{y}) - \tan\alpha(x_i - \bar{x}) \tag{3.22}$$

or

$$f = (y_i - \bar{y}) - \cot\alpha(x_i - \bar{x}) \tag{3.23}$$

when $f = 0$, it represents the major or minor axis. If f is larger than zero, (x_i, y_i) is above or to the right of the line $f = 0$; if f is smaller than zero, (x_i, y_i) is below or to the left of the line $f = 0$. Four furthest points in four directions (lower and upper with respect to the major axis, and lower and upper with respect to the minor axis) can be derived here. Let $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ be these four vertices (Fig.3.40).

The four corners of the bounding box can be then computed from these four furthest vertices with respect to the principal axes. Lines which pass through (x_1, y_1) and (x_2, y_2) and are parallel to the major axis are given as:

$$(y - y_1) - \tan\alpha(x - x_1) = 0 \tag{3.24}$$

$$(y - y_2) - \tan\alpha(x - x_2) = 0 \tag{3.25}$$

Lines which pass through (x_3, y_3) and (x_4, y_4) and are parallel to the minor axis are given as:

$$(y - y_3) - \cot\alpha(x - x_3) = 0 \tag{3.26}$$

$$(y - y_4) - \cot\alpha(x - x_4) = 0 \tag{3.27}$$

These four lines describe the four edges of the bounding rectangle. Lines given by Eq.3.24 and Eq.3.25 are both parallel to the major axis, and lines given by Eq.3.26 and Eq.3.27 are both parallel to the minor axis. The pairwise intersections of these lines can give the locations of the four corners (Fig.3.40) of the bounding box.

Eq.3.24 and Eq.3.26 intersect, and then the top left vertex of the bounding box

(x_l^t, y_l^t) is given as:

$$x_l^t = \frac{x_1 \cdot \tan\alpha + x_3 \cdot \cot\alpha + y_3 - y_1}{\tan\alpha + \cot\alpha} \tag{3.28}$$

$$y_l^t = \frac{y_1 \cdot \cot\alpha + y_3 \cdot \tan\alpha + x_3 - x_1}{\tan\alpha + \cot\alpha} \tag{3.29}$$

Similarly, Eq.3.24 and Eq.3.27 intersect, and then the top right vertex of the bounding box (x_r^t, y_r^t) is given as:

$$x_r^t = \frac{x_1 \cdot \tan\alpha + x_4 \cdot \cot\alpha + y_4 - y_1}{\tan\alpha + \cot\alpha} \tag{3.30}$$

$$y_r^t = \frac{y_1 \cdot \cot\alpha + y_4 \cdot \tan\alpha + x_4 - x_1}{\tan\alpha + \cot\alpha} \tag{3.31}$$

Eq.3.25 and Eq.3.26 intersect, and then the bottom left vertex of the bounding box (x_l^b, y_l^b) is given as:

$$x_l^b = \frac{x_2 \cdot \tan\alpha + x_3 \cdot \cot\alpha + y_3 - y_2}{\tan\alpha + \cot\alpha} \tag{3.32}$$

$$y_l^b = \frac{y_2 \cdot \cot\alpha + y_3 \cdot \tan\alpha + x_3 - x_2}{\tan\alpha + \cot\alpha} \tag{3.33}$$

Finally, Eq.3.25 and Eq.3.27 intersect, and then the bottom right vertex of the bounding box (x_r^b, y_r^b) is given as:

$$x_r^b = \frac{x_2 \cdot \tan\alpha + x_4 \cdot \cot\alpha + y_4 - y_2}{\tan\alpha + \cot\alpha} \tag{3.34}$$

$$y_r^b = \frac{y_2 \cdot \cot\alpha + y_4 \cdot \tan\alpha + x_4 - x_2}{\tan\alpha + \cot\alpha} \tag{3.35}$$

Each set of $((x_l^t, y_l^t), (x_r^t, y_r^t), (x_l^b, y_l^b), (x_r^b, y_r^b))$ with the centroid $(\bar{x}, \bar{y})$ describes a fitted bounding box for the test object with a certain orientation. The next step is to figure out which one is the minimum bounding box.

Searching Minimum Bounding Box

The minimum bounding box is defined as the one whose area is the smallest among all candidates. The range of the orientation, α, is always within $[0, 2\pi)$. It is reasonable and practical to conduct an exhaustive searching in the range $[0, \pi)$ to find out the particular angle with which the specific bounding box has the minimum area. The searching space is linear and the size of the space is dependent on the interval between two consecutive steps. The cost function D could be defined as:

$$D = \arg\min(A_{box}(\alpha)) \tag{3.36}$$

where $A_{box}(\alpha)$ represents the area of the bounding box and is a function of α.

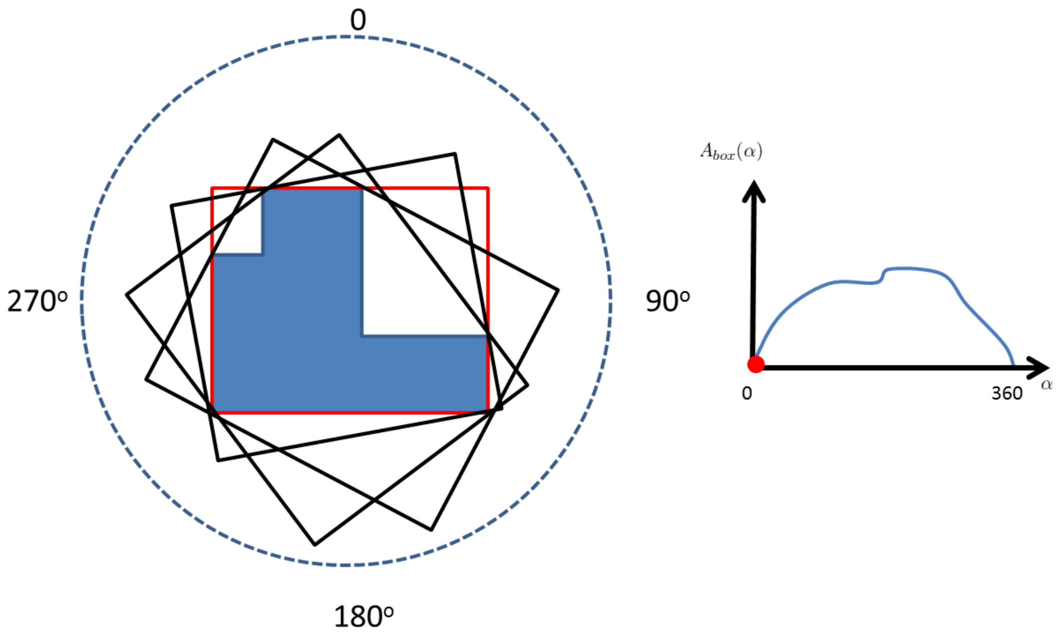


Figure 3.41: Search the minimum bounding box. The rectangles in the left image are different bounding boxes with different orientations. The right image shows the plot of the area with respect to the orientation. In this figure, the red rectangle is the final minimum bounding box.

Fig.3.41 illustrates how the searching process is conducted. Four candidate boxes are shown in the figure. The red box corresponding to the red dot in the plot is the minimum bounding box. The value of D is also shown minimum at where the red dot locates in the plot. In this case, the optimal value of the angle is 0.

Experimental Results on Synthetic and Real Data

For the purpose of testing the effectiveness of the search strategy, a set of objects, including both synthetic and real data, are analyzed. It is straightforward to produce synthetic data as binary images shown in Fig.3.42 and Fig.3.43. The white area in the image is the testing target simulating some random rooftop, and the black area is the background. Fig.3.44 shows real data derived from some LiDAR rooftop points. Some preprocessing is conducted here. The 3D points are first projected onto the 2D grid and transformed to a 2D binary image. All the steps of bounding box determination are applied to these binary images. Locations where minimum cost values happen can be easily identified from the plots from Fig.3.42 to Fig.3.44. The results show that this approach is accurate and fast.

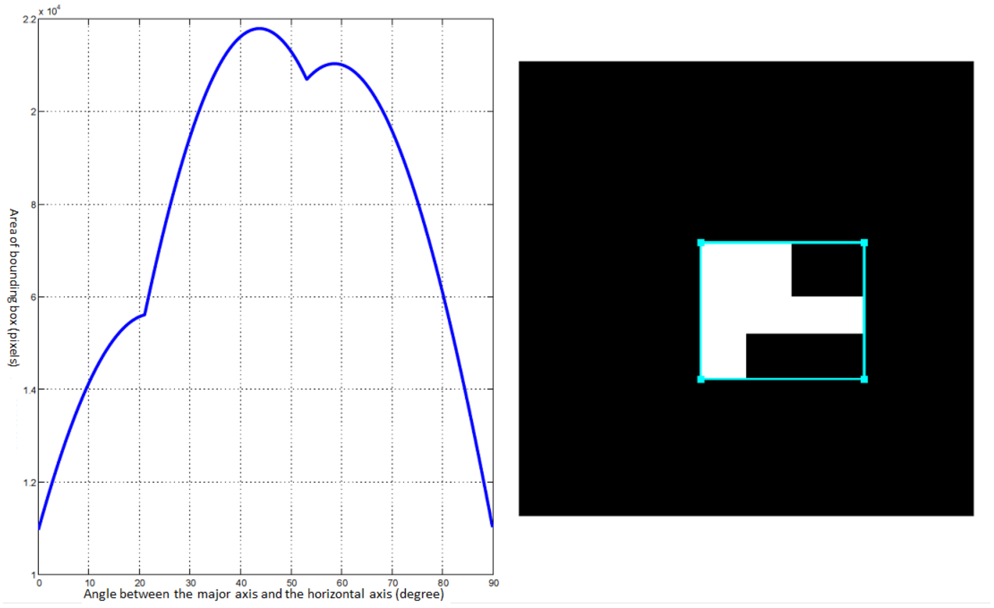


Figure 3.42: Synthetic data No. 1. Left plot shows where the minimum area is. The cyan rectangle in the second figure is the final minimum bounding box.

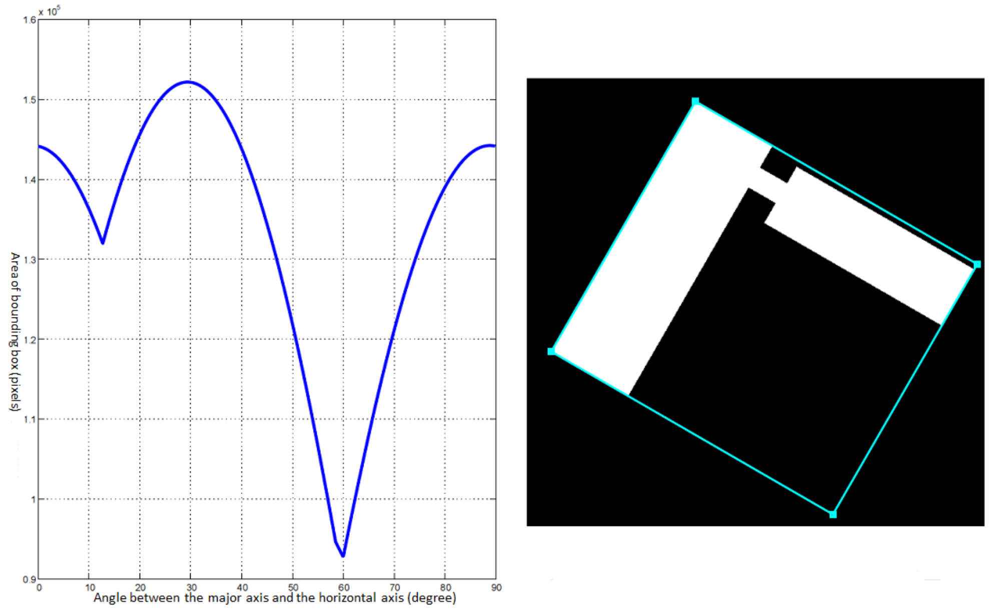


Figure 3.43: Synthetic data No. 2. Left plot shows where the minimum area is. The cyan rectangle in the second figure is the final minimum bounding box.

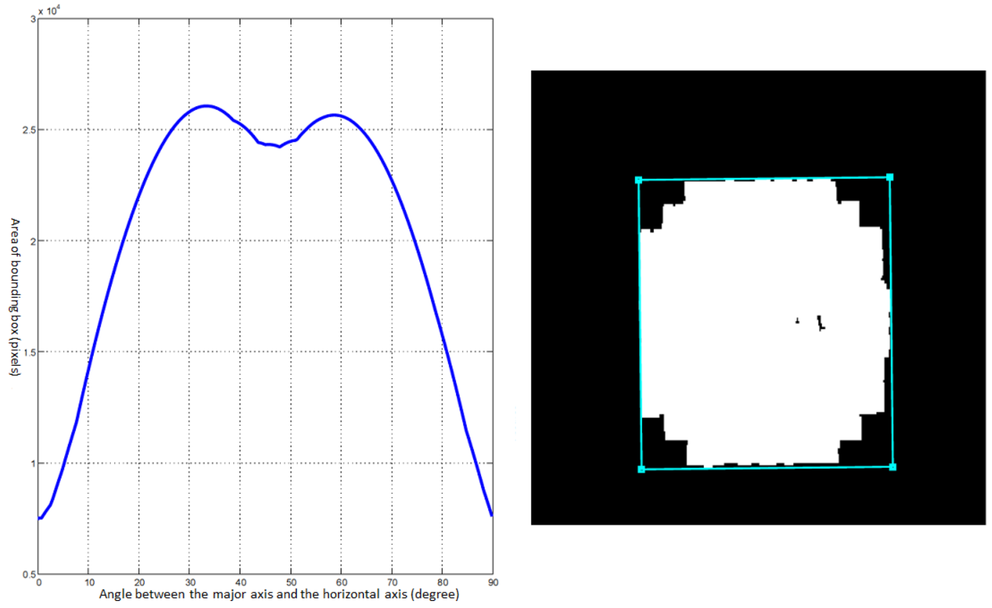


Figure 3.44: Real example data. Left plot shows where the minimum area is. The cyan rectangle in the second figure is the final minimum bounding box.

3.7 3D Modeling

Once the orientation of the structure is determined, one can move on to the topic of final description of building rooftops - building models. Successful extraction of major rooftop features and estimation of rooftop orientation is not sufficient to deliver a CAD-like model. This section introduces, as the final major step in the developed workflow, a description of how to generate simplified polygonal building models and terrain (if required). The structure of a building model will be first defined. Accurate boundary production, surface triangulation, and some advanced processing will be also discussed. For the terrain, the only issue that needs to be addressed is hole filling where vegetation may have been removed.

3.7.1 Structure of Building Model

After the rooftop feature detection, a building roof is divided into a number of regions. Each individual region can be described by an arbitrary shape. The boundary of the shape defines this rooftop feature. In the modeling process, each complete building model consists of a group of independent structures, each of which corresponds to one feature on the rooftop, and is produced by extruding its boundary vertically to the ground. In this fashion, each structure is created as a bottomless volumetric box, and the vertical sides of each box can form the walls naturally. Obviously, each box can only have one top but multiple sides (at least three). The combination of all the boxes is expected to deliver a watertight geometric model. Fig.3.45 gives as an visual example. The building model used in this research can be represented as:

$$Building = \bigcup_k Box_k (k = 1...n) \tag{3.37}$$

$$Box = Top \cup Sides \tag{3.38}$$

Lin *et al* [77] design their building models in a general, hierarchical-tree based representation, which also interprets models as a group of boxes (blocks). But, the form of representation is designed particularly for residential houses. They also proceed to employ strict rules of structure which contradicts the task here of imposition of no predefined models.

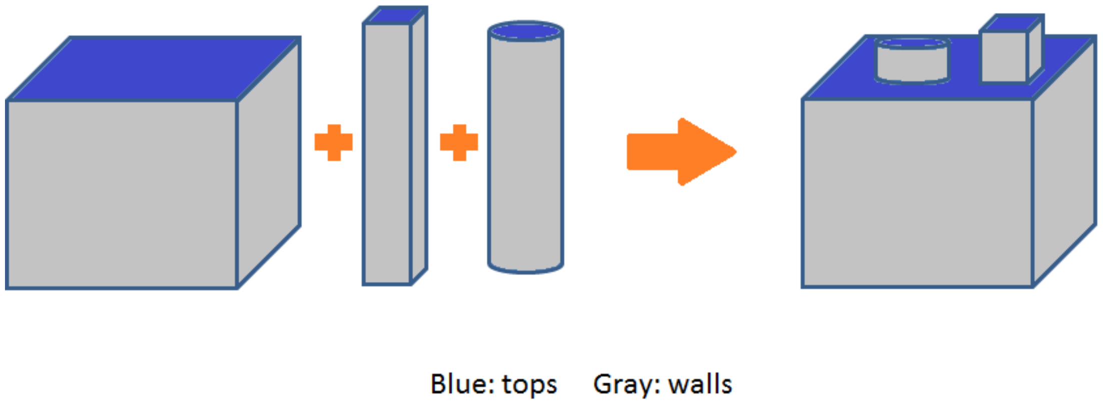


Figure 3.45: A sketch example showing one building model obtained by combining three boxes together.

3.7.2 Roof Boundary Extraction, Simplification and Refinement

For every feature that is part of the rooftop, the boundary is extracted from the 2D binary image generated by the same 2D projection method introduced in last section. This outline is a 2D closed contour, namely $\mathcal{B}_d$, which is actually a dense set of sequential points. The Douglas-Peucker line simplification algorithm [16] is then applied to this dense set of points. The result becomes a simplified contour, namely $\mathcal{B}_s$, consisting of a set of vertices whose number is largely reduced but finely representative of the shape of the original contour. For each line segment $e \in \mathcal{B}_s$, its direction is refined by examining how much it diverges from the principal direction. If the divergence is small enough, then e will be snapped to the principal direction. For those relatively small features on the rooftop, the original shapes of theirs are extremely difficult to estimate due to the limitation of sampling. Thus, for these small features, their minimum bounding boxes are used to represent their geometry.

3.7.3 Generation of Polygonal Mesh

Given the defined extruded models from the previous section, the next step is the generation of a polygonal model. The outlines produced in the previous step are arbitrary polygonal loops. Self-intersecting (one edge goes across another edge in the same loop) is not allowed to exist. The generated polygons are then connected to the ground level by adding vertical walls, which can be called ”orthographic-extruding”. The boundary loops have already been topologically-corrected, which means though the x and y locations of each vertex of the loop are determined in 2D fashion, the elevation value of each vertex is calculated according to the parametric model estimated by RANSAC (non-level features). Fig.3.46 shows a couple of polygonal meshes with RANSAC rectified points on them, the non-level features are also highlighted.

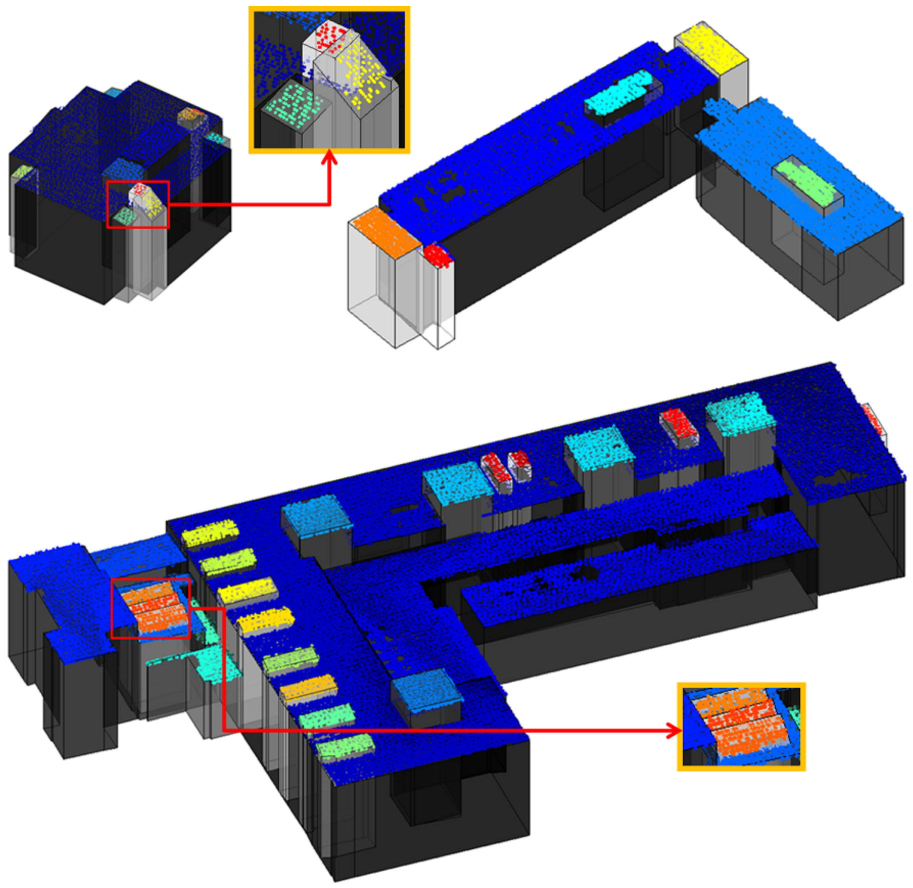


Figure 3.46: Some polygonal mesh building models with points also plotted on them. For each building model, every color represents an individual rooftop feature. For those relatively small features, their shapes are described by their minimum bounding boxes directly.

3.7.4 Surface Triangulation

As this point, each rooftop feature has been described by a fine polygonal contour that could be convex or concave, but not self-intersecting. When the derived model is written to files based on industrial standard 3D file formats (ply, obj, json, etc.), each polygon is recognized as a single face regardless of its possible complicated shape. The reality is that all 3D rendering engines require the surface element of the geometry to be triangular facets. In order to meet this demands, each polygonal loop must be triangulated.

A constrained 2D Delaunay Triangulation [78] is adopted here. For each individual rooftop feature, once the boundary production is done, all the vertices of the boundary are used to create a triangulated mesh which is close-to-Delaunay. General Delaunay Triangulation does not take into account the order of vertices from the input polygon and makes unnecessary connections between points. Thus, it introduces extra edges outside the original polygon loop given a non-convex shape. Constraints are usually added to triangulate a non-convex polygon. It is not necessary to impose constraints on convex shapes. Fig.3.47 shows the difference between the unconstrained Delaunay Triangulation and the constrained Delaunay Triangulation. The red contour in the figure indicates the input non-convex polygon that is used as the constraint. Fig.3.48 shows how the constrained Delaunay Triangulation applies to polygons in the real model.

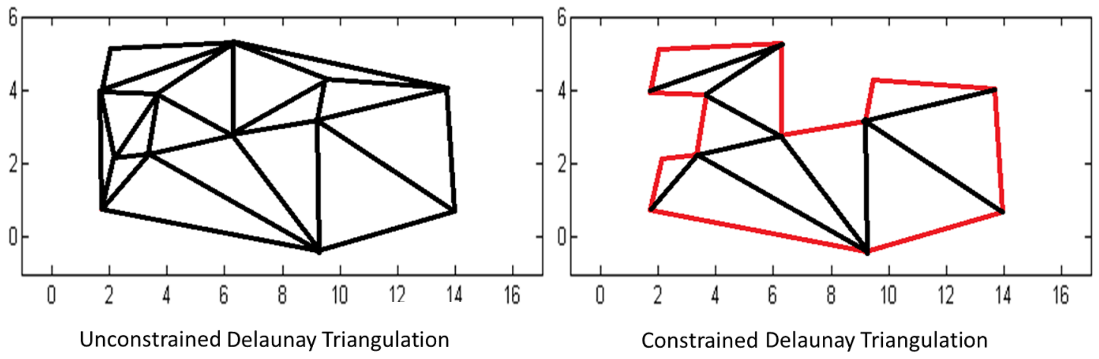

6
4
2
0
0
2
4
6
8
10
12
14
16
Unconstrained Delaunay Triangulation
Constrained Delaunay Triangulation

Figure 3.47: The difference between unconstrained Delaunay Triangulation and constrained Delaunay Triangulation, the red line in the bottom figure is the constraint which is the outline of this non-convex polygon.

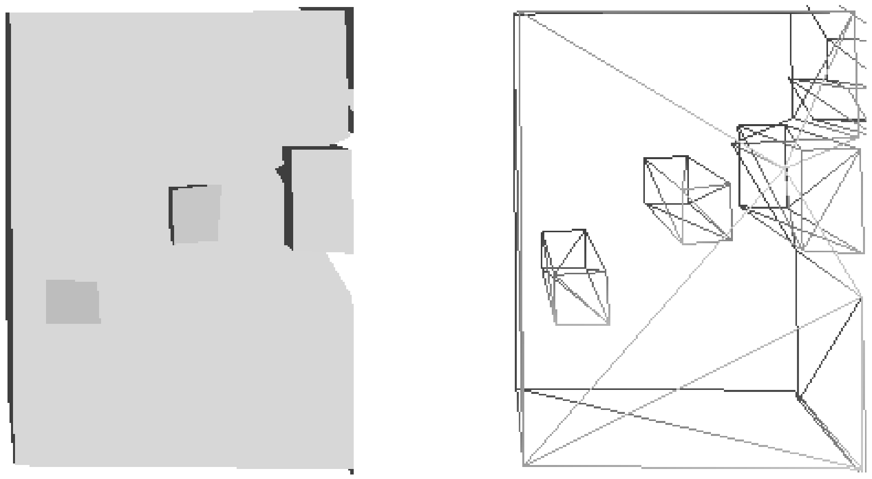


Figure 3.48: Part of an actual 3D model and its wireframe after constrained Delaunay Triangulation.

3.7.5 Terrain Hole Filling

The terrain portion extracted from the whole point cloud is incomplete after all the other objects (vegetations and building footprints) are removed from the scene. Multiple holes in random shapes are left where buildings or trees were located (Fig.3.26). When the building models and the terrain model are put back together, the holes on the ground caused by the buildings can be ignored. However, the trees are removed permanently and the holes caused by them are exposed if no further processing is done. Filling the holes and producing a smooth, hole-less ground surface (a.k.a. DEM) becomes necessary here. There are some free DEM data available but with very low resolution (e.g. 10 meters) which is not good enough. Precise DEM estimation is not one of the major tasks in this research, instead, the purpose of filling the holes is mainly for visualization and completeness of the scene.

All the points left on the ground with their normals compose a set of oriented points. A smooth surface is desired. This can be regarded as a spatial Poisson problem, therefore, a Poisson surface reconstruction [79] technique can be applied to re-sample the original set of points and produce a smooth ground surface, roughly equivalent to a precise DEM.

Poisson problems have been arising in a variety of applications. The particular problem here is taking a set of samples. Each sample is an oriented point. All of these oriented points are assumed to lie on or near the implicit surface of an unknown model. The goal is to reconstruct a watertight, triangulated surface which is the best approximation of the real surface. Unlike many other surface reconstruction approaches that usually segment the point set into a group of regions and do local fitting, the Poisson formulation handles all the oriented points at once and is very resilient to noise. The most popular implementation was developed by Kazhdan [79] and is used in this research.

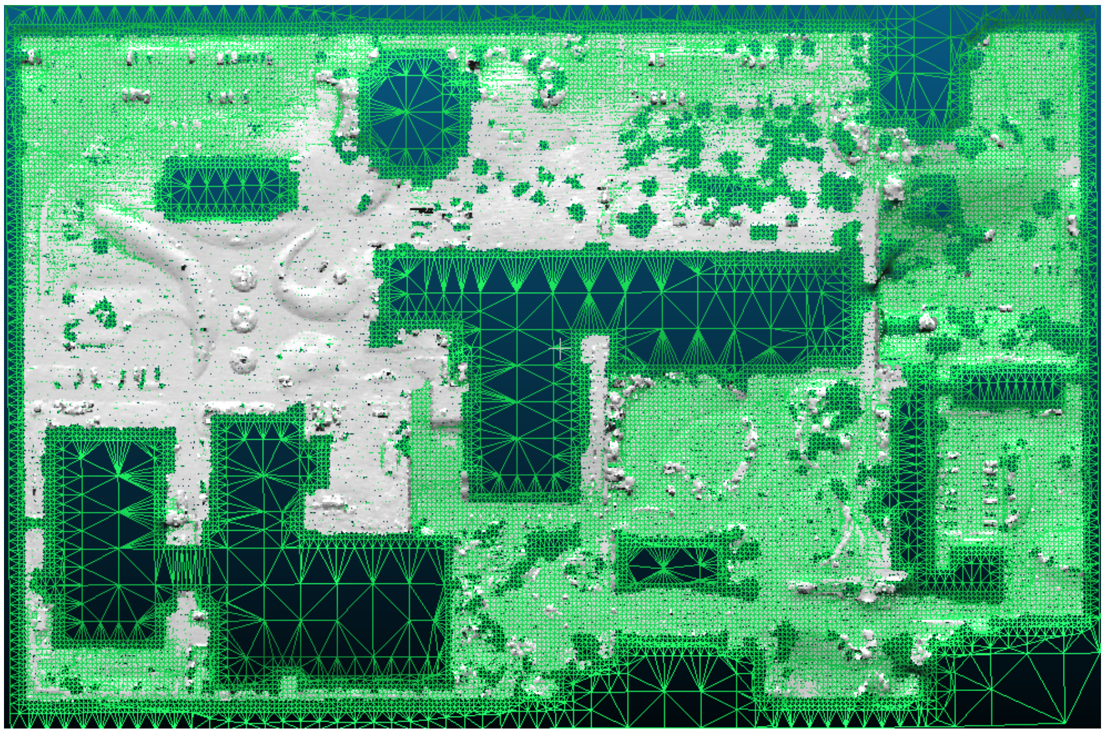


Figure 3.49: The original points of the terrain (white) and the reconstructed wire-frame of the Poisson surface.

The white dense point set depicted in Fig.3.49 are the original, oriented points. The green wire-frame shows the triangulated reconstructed surface. It can clearly been seen that this is a nonuniform resampling process.

3.7.6 Roof Feature Polygon Clipping and Closing

During the previous discussion, each rooftop feature is considered as an independent part to be processed, including the shape (plane) fitting and boundary production. The final combination of all the boxes creates a complete building model. Each polygon remains intact after the boundary production regardless of potential impacts from neighboring roof features. So far, this type of impact appears to not be a necessary concern since most of features we have seen are horizontal planar surfaces, which is quite common in urban scenes. For this kind of roof polygons, their boundaries have no contact with each other. Therefore, no impact would be made to the existing outlines. However, it is also common to find more complicated structures rather than one simple planar face on the rooftop, such as a shed structure containing one feature only, a gabled structure containing two connected slant features, or a hipped structure containing more than three connect slant features (Fig.3.50). For these kinds of structure consisting of more than one slant feature, the features are supposed to naturally meet together and produce ridges or spines. Due to the imperfection of input data plus the errors being generated along the pipeline, this behavior can not be obtained for free. Further processing is necessary. A way to find these ridges or spines is to find the common intersections between the planes defined by these slanted polygons. Recovering this level of detail is also critical to create realistic and tight building models.

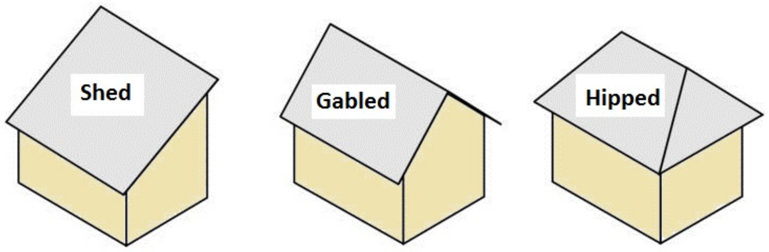


Figure 3.50: Some basic non-level roof types.

During the modeling process, it is common to observe two types of interaction between adjacent non-coplanar faces; over-intersection or under-intersection with each other (Fig.3.51). In Fig.3.51, polygon $\mathcal{P}_1$ (green) and polygon $\mathcal{P}_2$ (yellow) are supposed to meet and produce a natural, sharp spine edge. In the left figure, they are over-intersected; that is the common intersection of these two polygons is approximately used as the spine of the structure. The redundant parts can then be cut off to leave a clean structure. This step is called ”clipping”. In another case, shown in the right figure, $\mathcal{P}_1$ and $\mathcal{P}_2$ are under-intersected. They have no contact with each other. The solution to this example is to extend the plane (gray outline) defined by $\mathcal{P}_1$ to intersect with $\mathcal{P}_2$. The right edge on $\mathcal{P}_2$ can be found by conducting ”clipping”. Once the appropriate edge is found, $\mathcal{P}_1$ is then extended to the new found intersecting line in $\mathcal{P}_2$. This step is called ”closing”.

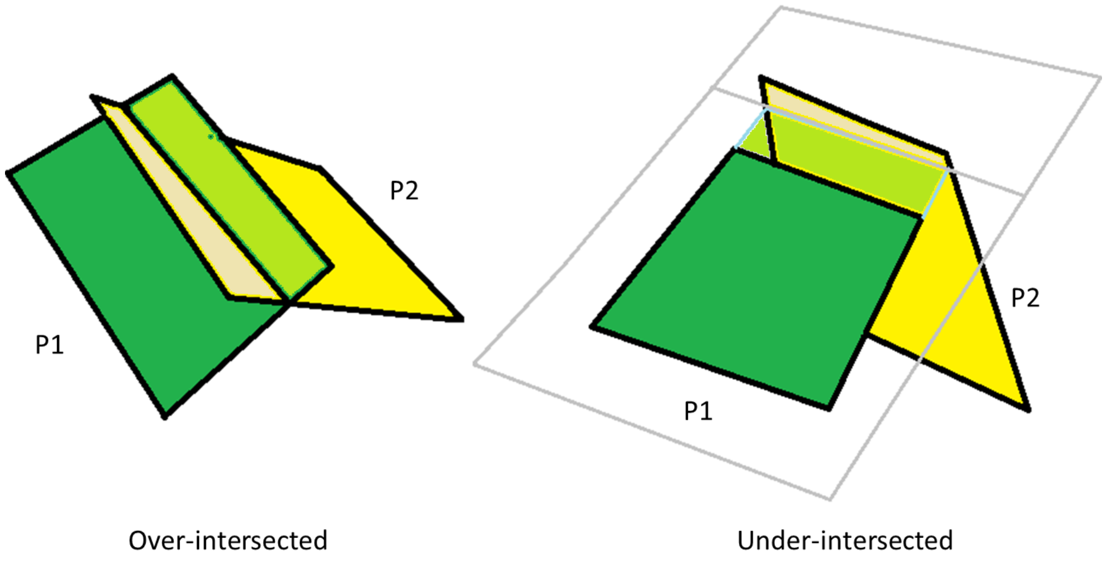


Figure 3.51: Two types of interaction between two adjacent faces: over-intersect and under-intersect. $\mathcal{P}_1$ and $\mathcal{P}_2$ are supposed to naturally meet and provide a spine edge.

In order to maintain the automated workflow in place so far, the system must be able to decide when to apply clipping and closing to rooftop faces. A grouping rule has to be implemented so that when complicated roof structure is encountered, extra steps are taken to achieve the desired rooftop geometry. Fig.3.52 shows the logic of categorizing all the rooftop features into two different kinds, level ones and slant ones, based on their face normal directions. For the set of slanted faces, it can be furthered divided into smaller groups. Each group corresponds a single complex structure on the rooftop, such as gabled shape, hipped shape, or more complicated collections. For each individual group, the specifically designed clipping and closing strategy can be applied in a pairwise fashion. It is important to point out that the groupings from the original list of rooftop features can be achieved here due to the nature of region growing approach. The neighboring faces in the list are also neighboring faces on the rooftop. It is a critical piece of information for the system to decide if two rooftop features are potentially connected.

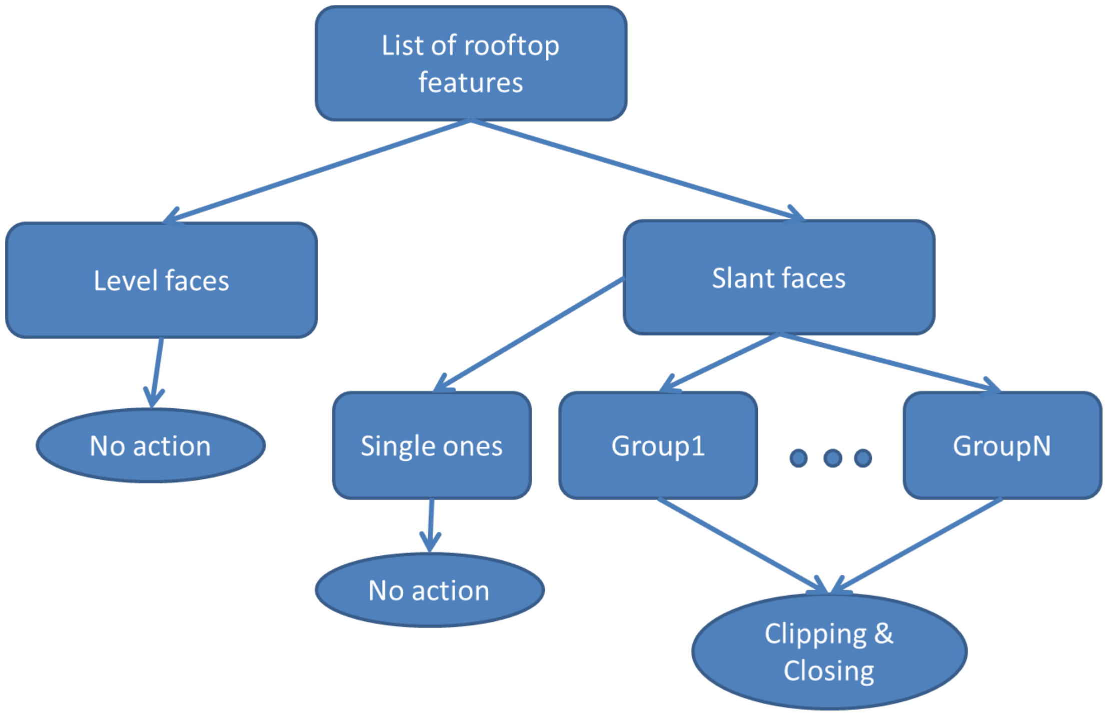


Figure 3.52: Pipeline to process the rooftop with multiple structures consisting of slant surfaces other than level surfaces.

Fig.3.53 shows an example of a particular roof that has a number of level and slanted features. This is the top of the Bausch & Lomb building in downtown Rochester, NY. All the slanted faces can be grouped into two clusters. Each cluster contains four connected faces. Without any processing, it is easy to observe that they are over-intersected. After the clipping and closing, the appearance of the model looks more appealing, and accurate than before.

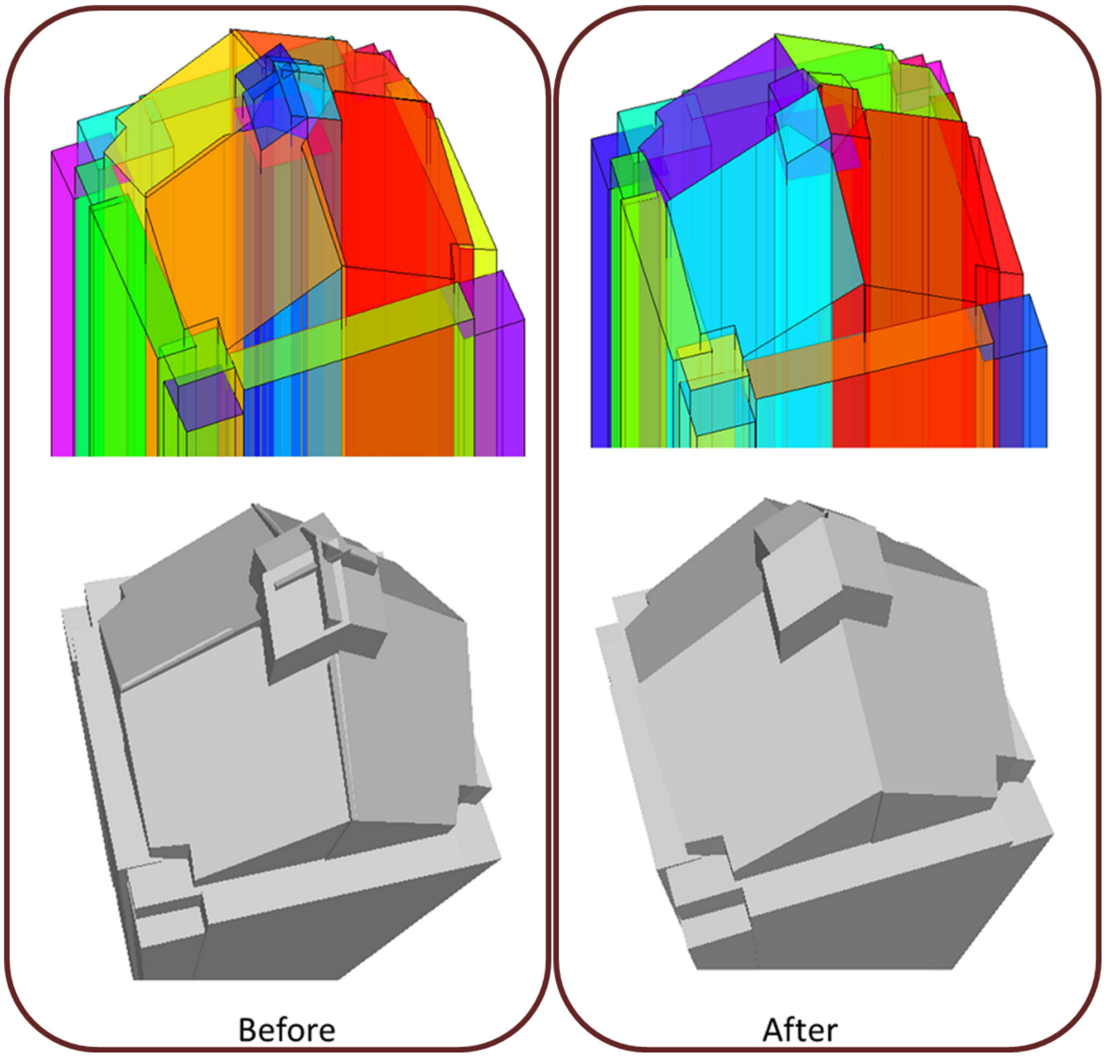


Figure 3.53: An example of rooftop with both level and slant features. All the slant features are refined by the extra effort of clipping and closing.

Chapter 4

Experiments and Visual Results Using Real Data

In this chapter, visual results generated by applying the developed workflow on the real LiDAR data will be presented. The presentation includes both individual building structures and complete scene models.

4.1 Implementation and Parameter Setting

The whole pipeline has been implemented in MATLAB with geom3d (a MATLAB library to handle 3D geometric primitives: create, intersect, display, and basic computations) and C++ with PCL (Point Cloud Library). The main entrance of the workflow is a standalone MATLAB script (the sample is shown in Appendix B.1). All the experiments were performed on both Windows 7 and CentOS Linux operating systems.

There are a few parameters that are introduced in the pipeline. One of the major advantages of the developed algorithm is that many parameters, such as the number of neighbors that needs to be used at multiple places, the search radius while growing regions, are highly related to the point density, and the point density can be derived from the input itself (Eq.3.1). The parameters that are not straightforward to be derived, such as the coefficients used in graph cuts based classification, curvature and angle thresholds used in region growing algorithm, do not interfere with the result quality. Thus, these parameters can be very stable on a big range of data. In summary, all the parameters through the workflow can be either fixed or derived. Very little effort in tuning parameters is required. However, users are still allowed to specify these necessary parameters if they receive proper training and understand this workflow well. Please refer to Appendix B.1 for setting up these major parameters.

4.2 Data and Considerations

The workflow has been tested on various types of LiDAR data collects, including a university campus, a midsized city and an industrial plant in a rural setting. The point density varies from about 4 pts/m^2 to 30 pts/m^2. Fig.4.1 shows the major data sets used; the R.I.T campus with wide open space between buildings, the dense business district in downtown Rochester, NY with large and tall buildings and trees. The input data from airborne LiDAR scanner contain millions of points. The results are produced using zero knowledge of the type of landscape and object layout in the scenes.

The level of detail of the reconstruction results is constrained by two factors. One is the input point density, and the other one is the fashion of data collection (aerial or ground). For instance, the rooftop details such as air-conditioning, chimneys or other types of small structure that might only be represented by very small number of points are ignored by the workflow. In addition, since only aerial data is involved in this research, since ground data is not available, facades of buildings are nearly impossible to reconstruct with detail and are ignored. All the sides in the reconstructed building models are chosen to be simple, extruded planar faces.

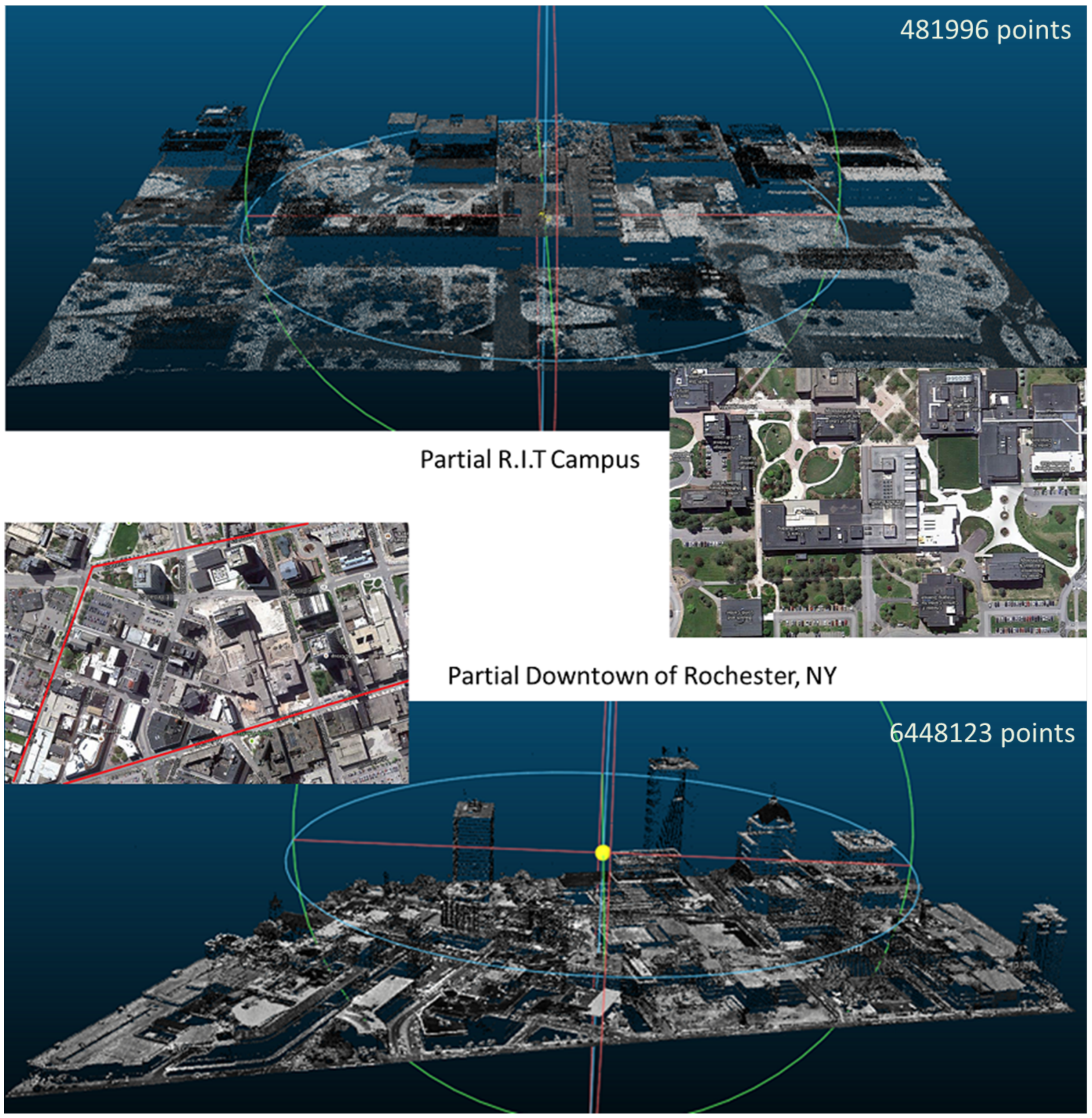


Figure 4.1: The major data sets tested in this thesis, including partial R.I.T campus and partial downtown of Rochester, NY. The point densities are about 4 pts/m^2, 25 pts/m^2, respectively. The intensity property is also displayed but not used in this research. Ortho-photographs are grabbed from Google Map.

Unlike regular residential areas where rooftops can be always decomposed into some standard shapes, such as gabled, hipped, shed, or mansard, eclectic university architectural design or variational urban construction discourages the involvement of geometric primitives. No rooftop templates will be considered in these examples in order to provide the flexibility of modeling any type of structure.

4.3 Performance and Visualization

The evaluation of the performance of building modeling extraction approaches is agreed to be a commonly known difficult job. Benchmarks in this field have not been established at this point in time. Visual examination is still the most widely used form of evaluation. Some researchers have developed a few objective ways to come up some measurements with respect to the input points for evaluation. Unfortunately, it still can not be accomplished without ground truth. In the next chapter, some rigorous evaluation will be performed on a simulated data set whose true measurements are known.

In this section, the visual evaluation and comparison with the widely accepted approach of 2.5D dual contouring [80] is provided. 2.5D dual contouring can also create crack-free models composed of rooftops connecting to the ground by vertical walls through energy minimization. It can recover planar or arbitrary, non-planar surfaces well, but lacks the ability to produce smooth and rigorous boundary. Shown in Fig.4.2 are the results from both the presented method and 2.5D dual contouring on the key data provided by the authors in their paper [80]. It is fair to say the method presented here produces better geometry with fewer vertices and faces. In other word, the file size is smaller, which is critical in many applications. Loading large-sized 3D data in most web browsers using WebGL, is certainly not a preferred choice. Most GIS applications also favor the type of model produced here rather than those from previous approaches [80]. The approach presented here is not perfect. There are many small features, such as chimneys, that are not reproduced by this method.

In regard to vegetation detection, the results are sufficient and satisfactory as discussed in Section.3.4.2. Tiny rough areas on the rooftop, or rooftop feature contours, are often recognized as tree points. In order to overcome this, more information, such as multi-spectral knowledge has to be involved. This is beyond the scope of this research.

Meshlab [81] and CloudCompare [82] are used here to visualize all the 3D models. These tools are free, open-source and widely adopted in this field. The two software tools have different ways of triangulating surfaces. As a result, if a non-convex and nontriangulated polygon is loaded, the outcome due to triangulation is often unpredictable. It is highly recommended that triangulation is preformed before hand as described in this research.

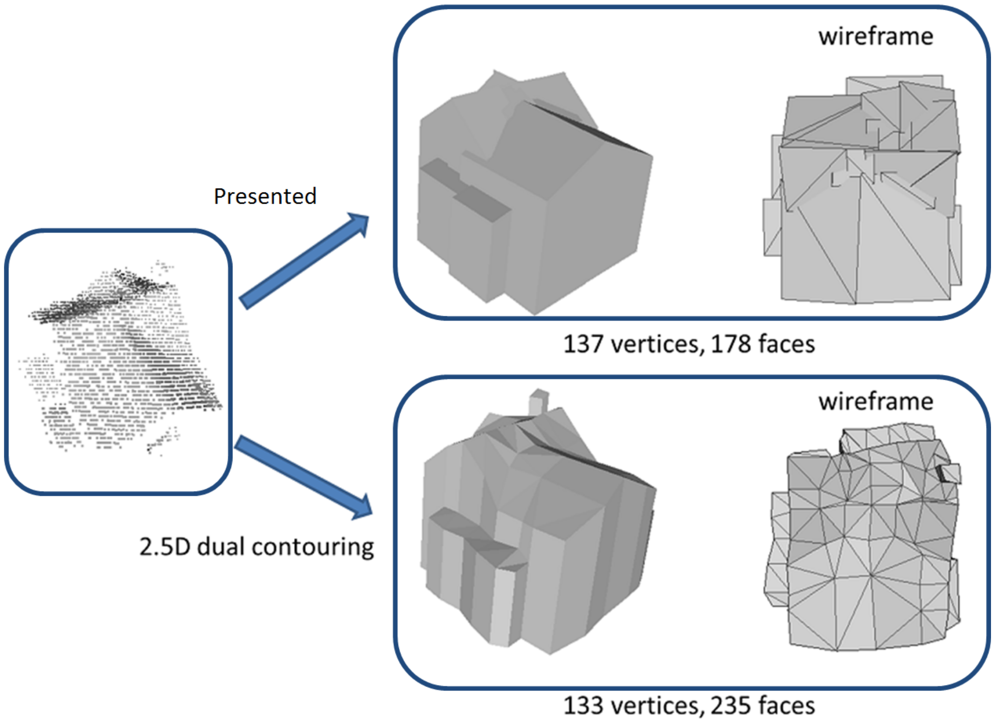


Figure 4.2: The data provided by [80]. Two results from both the developed single building modeling approach and 2.D dual contouring are presented here.

4.4 R.I.T Data

Fig.4.3, Fig.4.4, and Fig.4.5 present three building models from R.I.T campus. The 2.5D dual contouring version is also provided. Fig.4.6 and Fig.4.7 show the final modeling product from different view points.

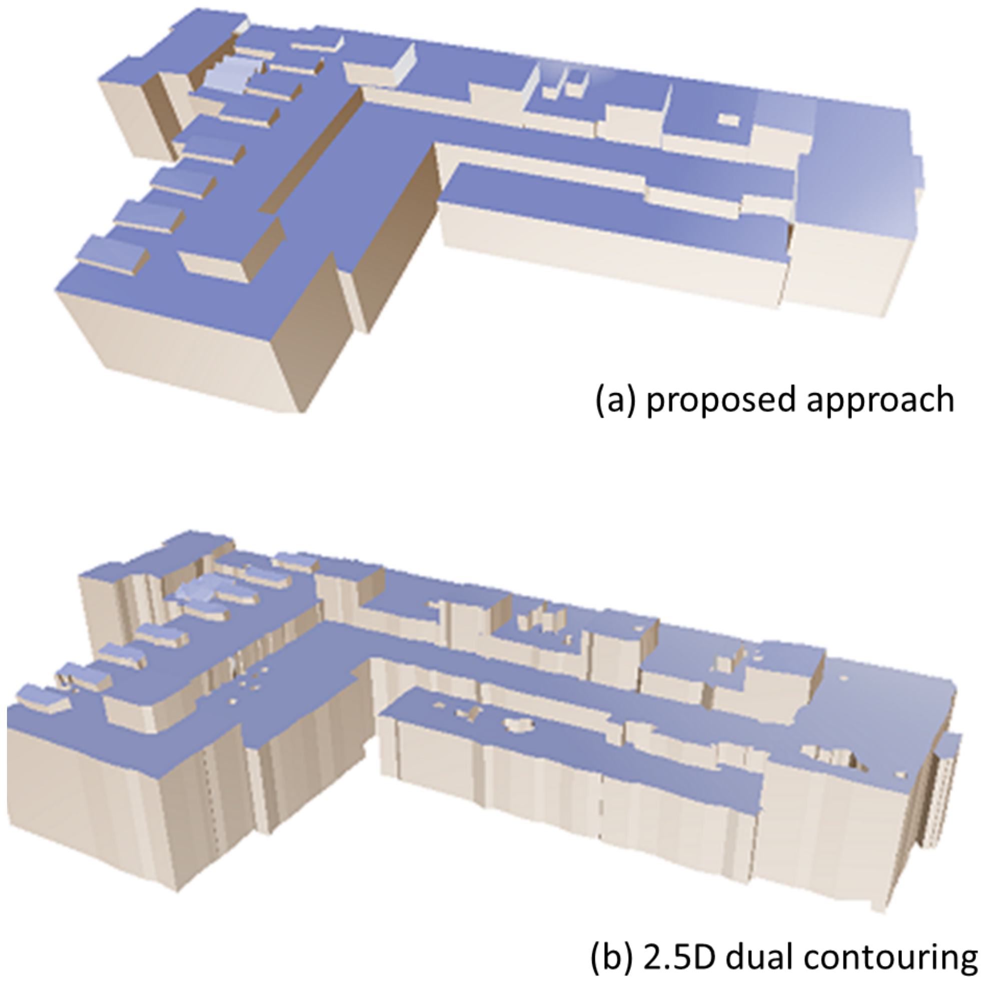


Figure 4.3: Final result of sample building models in comparison with 2.5D dual contouring result - Example 1 (from R.I.T campus).

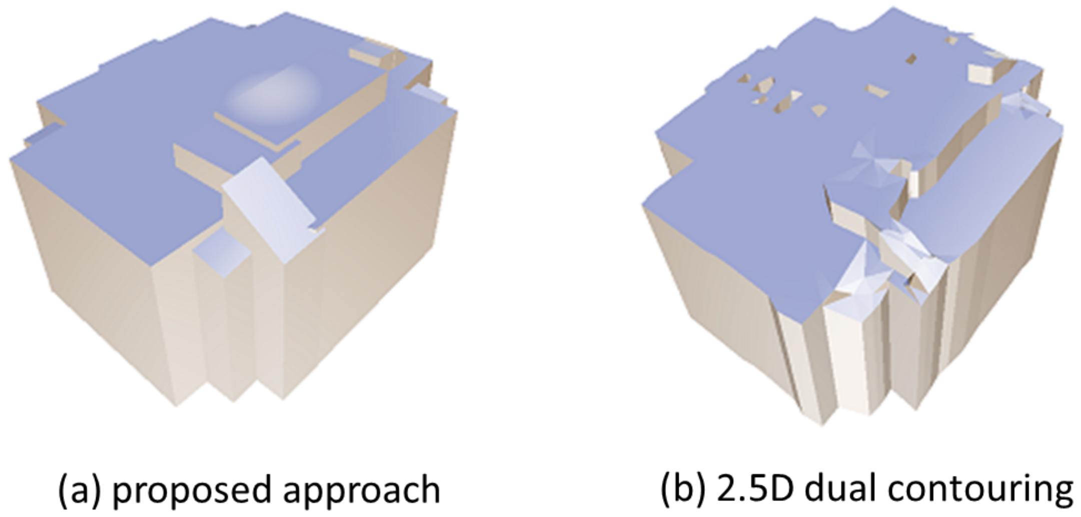


Figure 4.4: Final result of sample building models in comparison with 2.5D dual contouring result - Example 2 (from R.I.T campus).

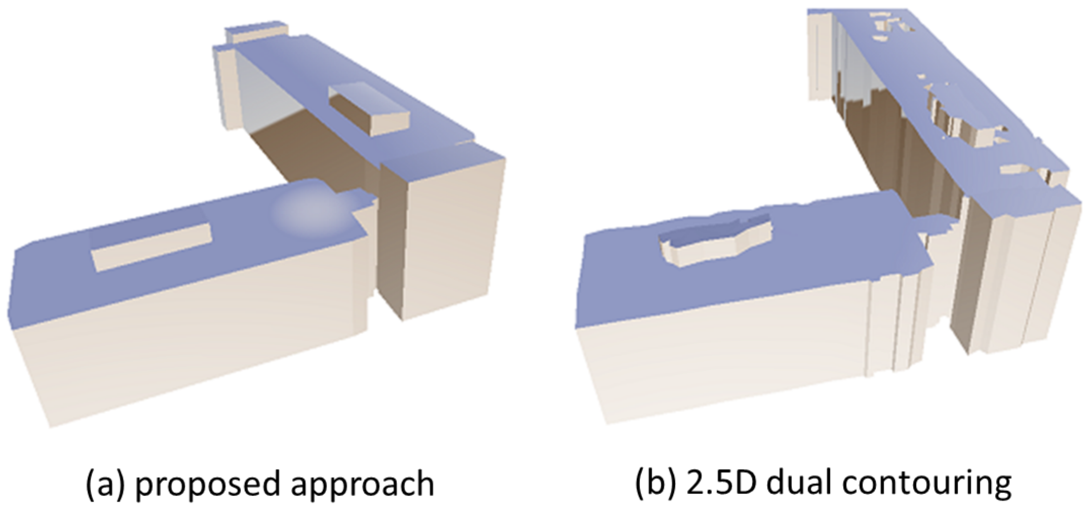


Figure 4.5: Final result of sample building models in comparison with 2.5D dual contouring result - Example 3 (from R.I.T campus).

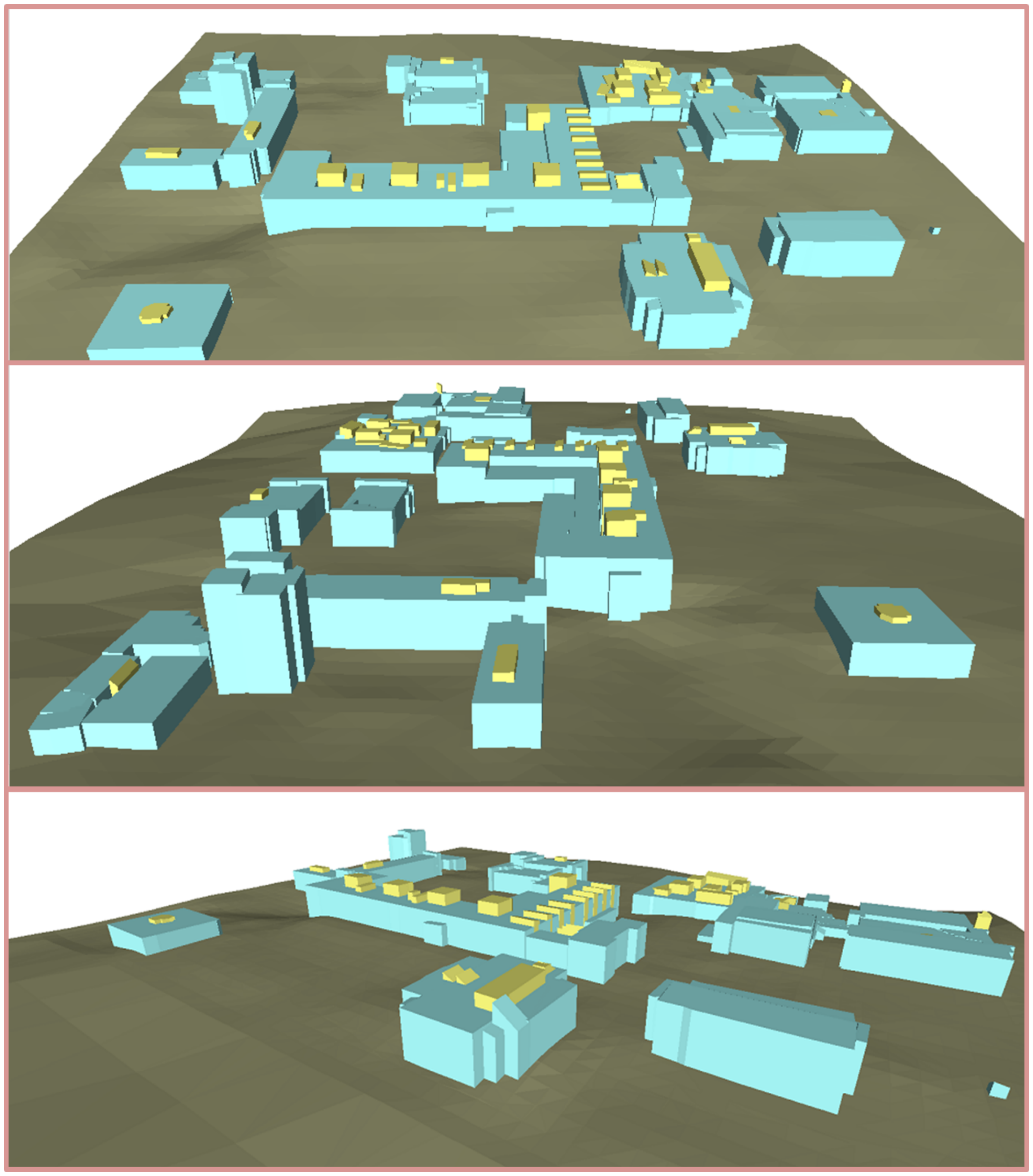


Figure 4.6: Final scene model of the northern part of R.I.T campus.

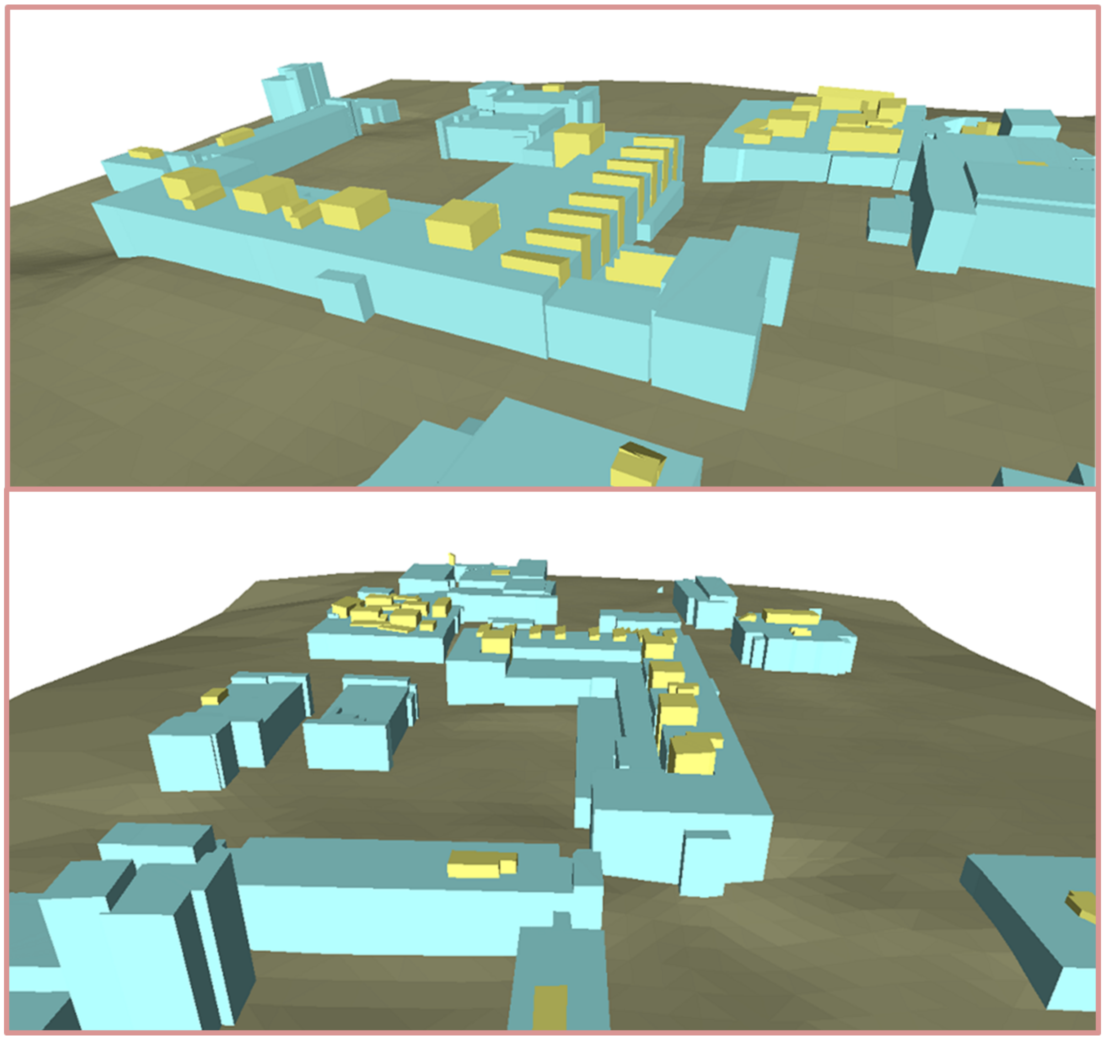
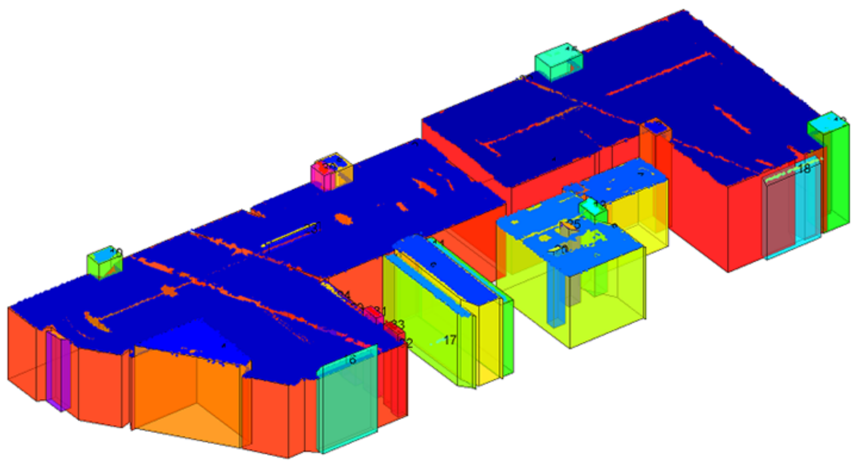


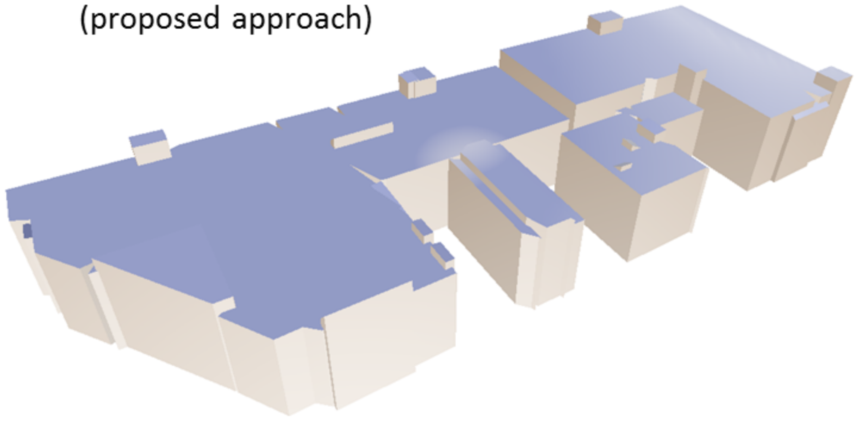
Figure 4.7: Final scene model of the northern part of R.I.T campus (continue).

4.5 Downtown Rochester and Alcoa Data

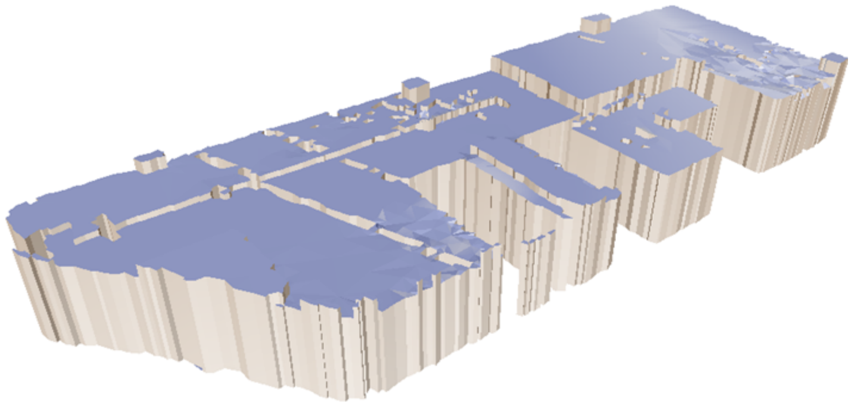
Fig.4.8, Fig.4.9, Fig.4.10, and Fig.4.11 present four building models from downtown Rochester, NY. The 2.5D dual contouring version is also provided. Fig.4.12 and Fig.4.13 show the final modeling product from different view points. Fig.4.14 is the data captured from another site, the Alcoa production facility in Massena, NY. Fig.4.15 shows the final scene model derived from this data.



(a) Polygonal model plus points with color coding (proposed approach)



(b) Polygonal model (proposed approach).



(c) Mesh model (2.5D Dual Contouring).

Figure 4.8: Final result of sample building models in comparison with 2.5D dual contouring result - Example 4 (from downtown of Rochester).

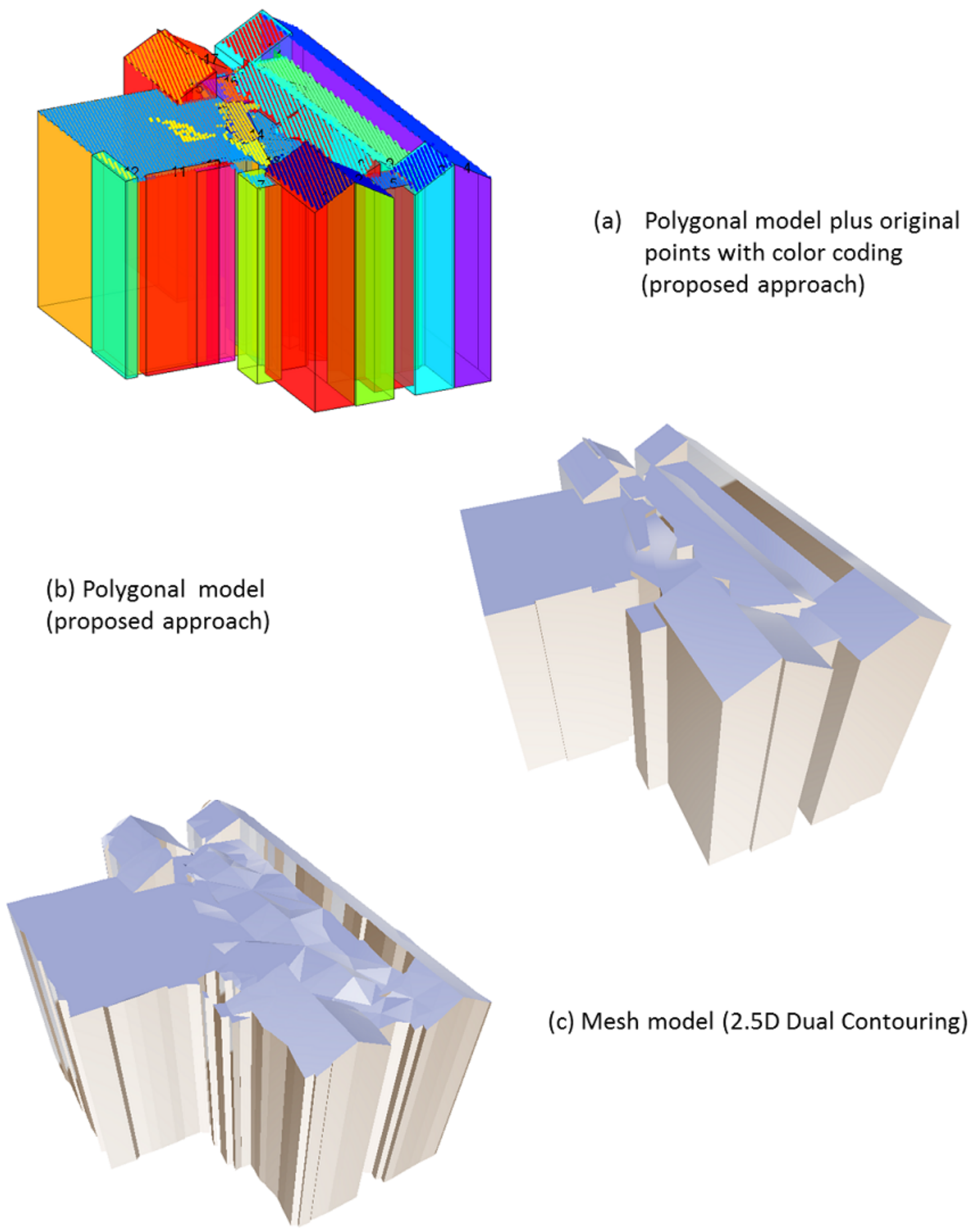

(a) Polygonal model plus original points with color coding (proposed approach)
(b) Polygonal model (proposed approach)
(c) Mesh model (2.5D Dual Contouring)

Figure 4.9: Final result of sample building models in comparison with 2.5D dual contouring result - Example 5 (from downtown of Rochester).

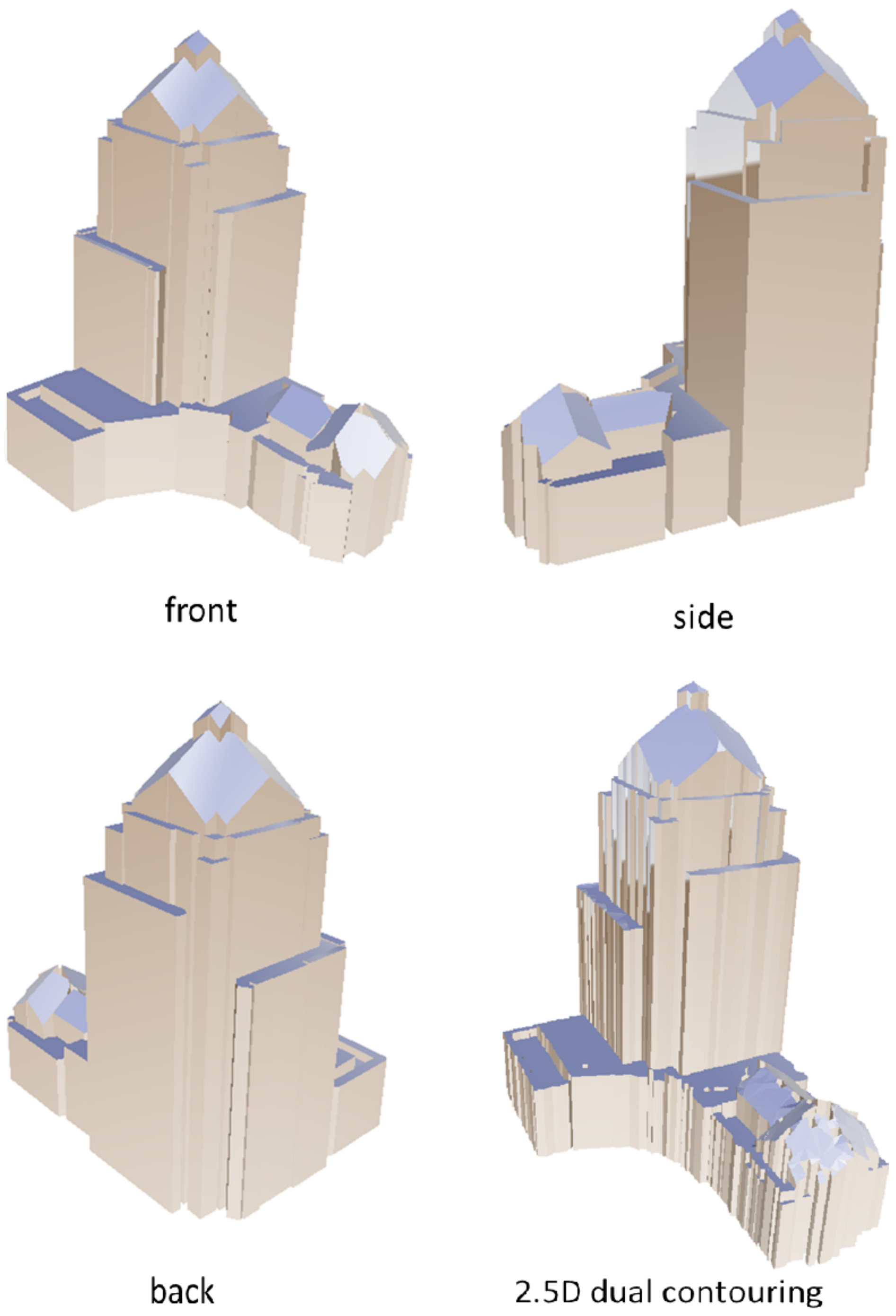


Figure 4.10: Final result of sample building models in comparison with 2.5D dual contouring result - Example 6 (from downtown of Rochester).

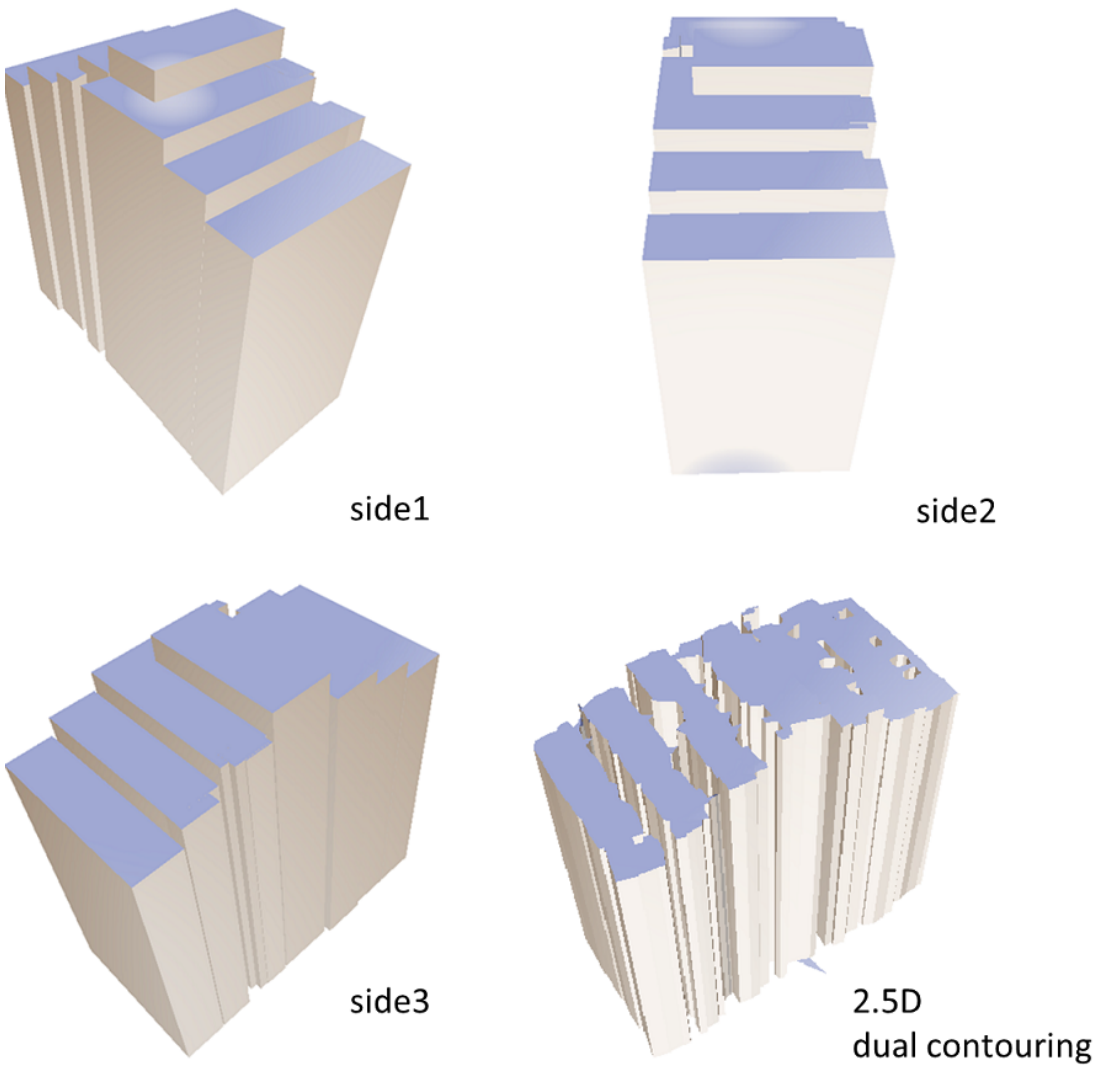


Figure 4.11: Final result of sample building models in comparison with 2.5D dual contouring result - Example 7 (from downtown of Rochester).

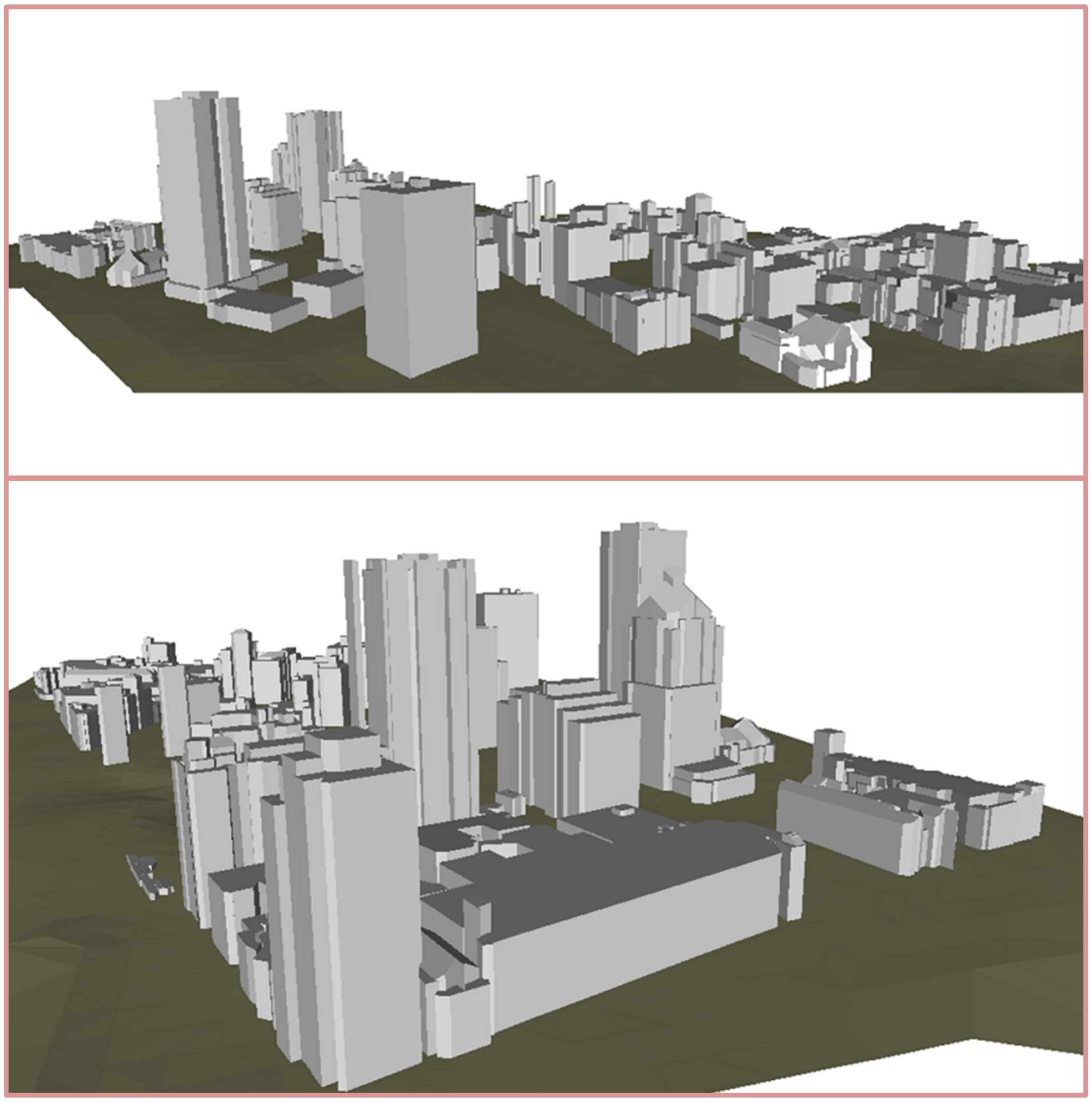


Figure 4.12: Final scene model of the partial City of Rochester.

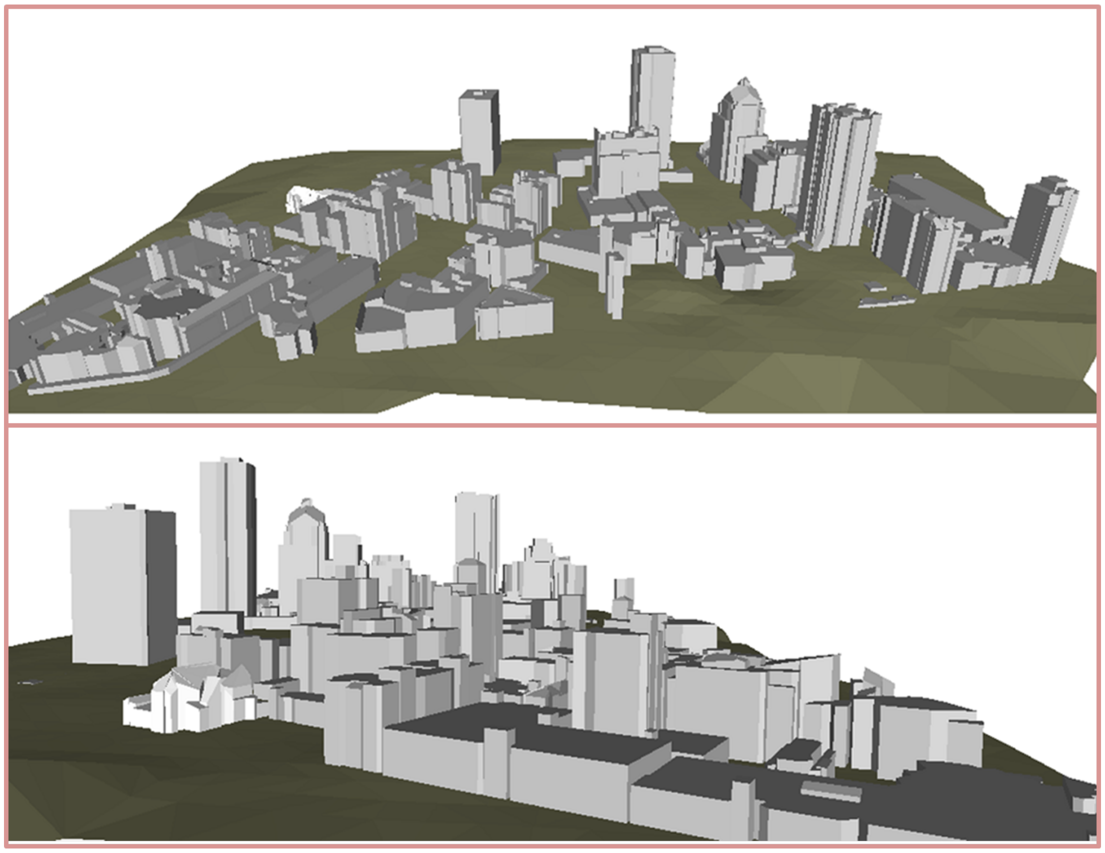


Figure 4.13: Final scene model of the partial City of Rochester (continued).

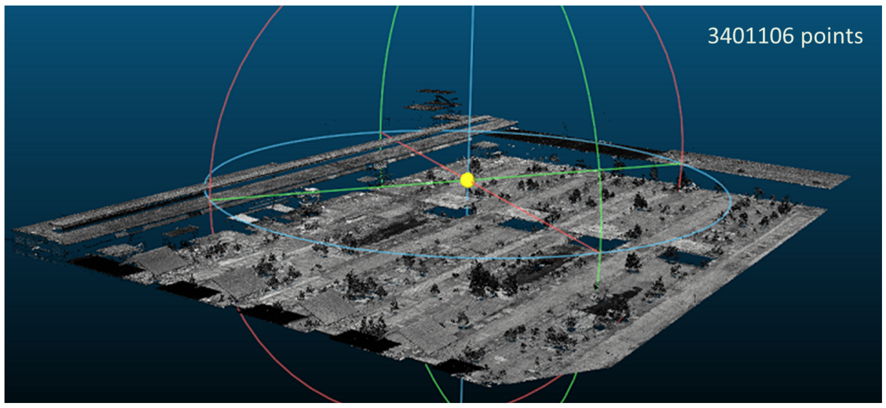


Figure 4.14: The data collected from a plant in Massena, NY.

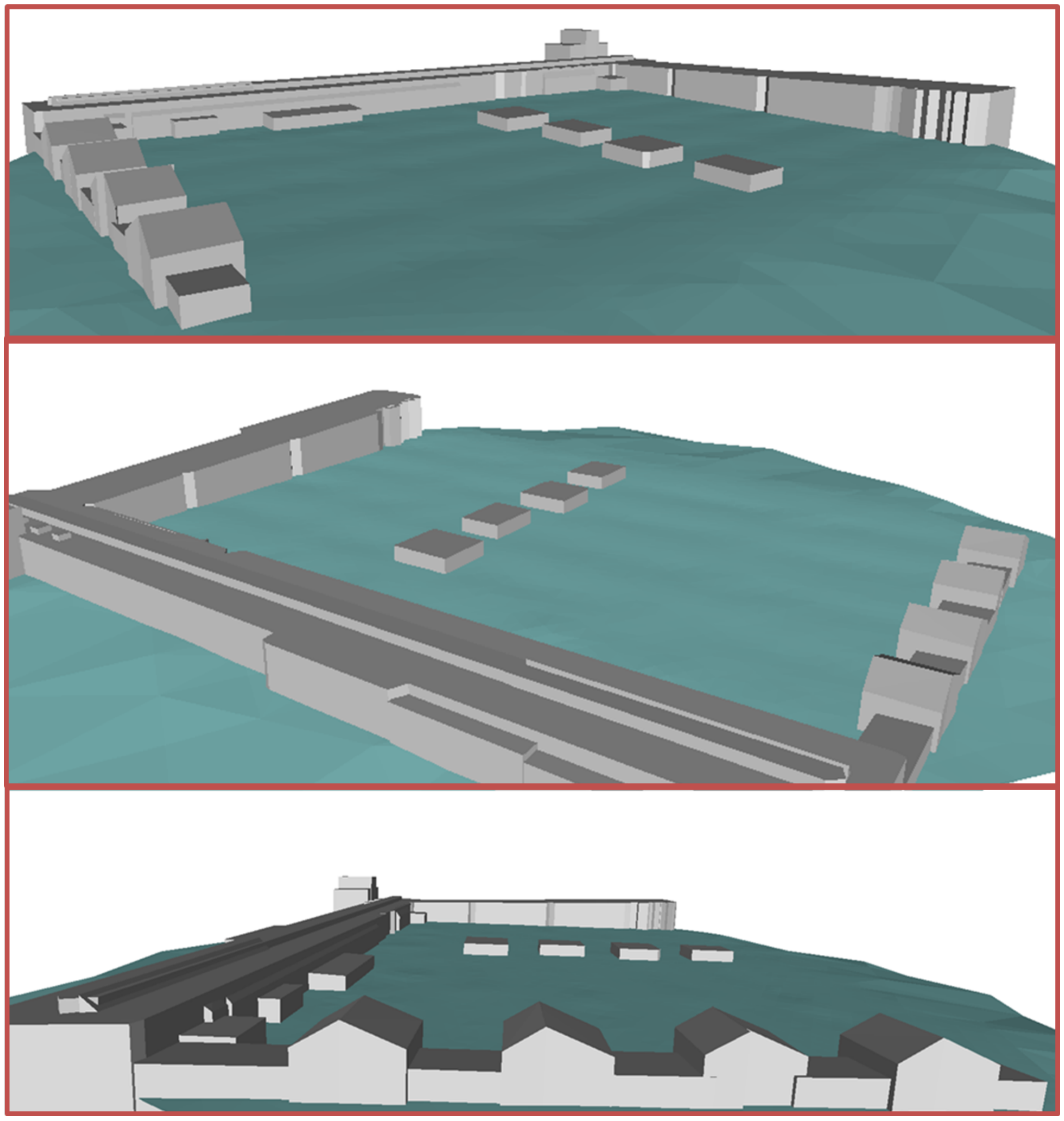


Figure 4.15: Final scene model of the plant in Massena, NY.

Chapter 5

Validation With DIRSIG Simulation

LiDAR point clouds are obtained through a sampling process of the target scene, thus the point clouds are not able to represent the true, continuous measurement of the scene features. The result of reconstruction from LiDAR data should be evaluated against the true scene measurements in order to present a convincing evaluation procedure. However this is impossible to do in practice for more than a handful of elements. Though it is difficult or impossible to get ground truth data of most urban or residential areas, the validation of how the workflow performs can not be simply ignored. Therefore, simulated scene data becomes a viable input for for the purpose of validation. The R.I.T tool called Digital Imaging and Remote Sensing Image Generation (DIRSIG) model is introduced in this section for the purpose of scene simulation and reconstruction evaluation. Given a set of scene parameters, including the scene content and sensor configuration, a simulated LiDAR point cloud can be produced with all the known true measurements for every feature in the simulated scene.

5.1 DIRSIG

The Digital Imaging and Remote Sensing Image Generation (DIRSIG) model is a first principles based synthetic image generation model developed by the Digital Imaging and Remote Sensing (DIRS) Laboratory at Rochester Institute of Technology. The model is able to generate passive single-band, multi-spectral or hyper-spectral imagery from the visible through the thermal infrared region of the electromagnetic spectrum. The model also has an very mature active LiDAR capability and an evolving active RF (RADAR) capability. The model can be used to test image system designs, to create test imagery for evaluating image exploitation algorithms, and for creating data for training image analysts [83]. DIRSIG is used to simulate a LiDAR scene in this work.

5.2 Scene Simulation

In order to throughly validate the capability and accuracy of the workflow, including scene classification and scene object reconstruction, a primitive scene has been simulated by DIRSIG. The true measurements of all the objects in the scene are known. A number of objects that have various geometric shapes are chosen to be in the simulated scene. The placed objects include two identical house with hipped rooftops, a stand-up cylinder, a partial sphere, a lie-down cylinder, and two cuboids. The details of the DIRSIG input parameters for these objects is presented in Table.5.2. Appendix.C contains the configuration file used to define the simulated scene. From the position of the sensor, one third of one of the two houses is occluded by one tree canopy, and half of one cuboid is occluded by another tree canopy. The terrain in this simulation is a totally flat surface. The resulting point cloud can be seen in Fig.5.2 and Fig.5.3. Fig.5.1 shows the original geometric primitives used in the simulation. The half-occluded object (one of the two cuboids) will not be recovered in this experiment due to the occlusion. The occluded house will be reconstructed though it is expected to be incomplete. There are 202,602 points in the original point cloud. After the noise removal, 185,121 points remain.

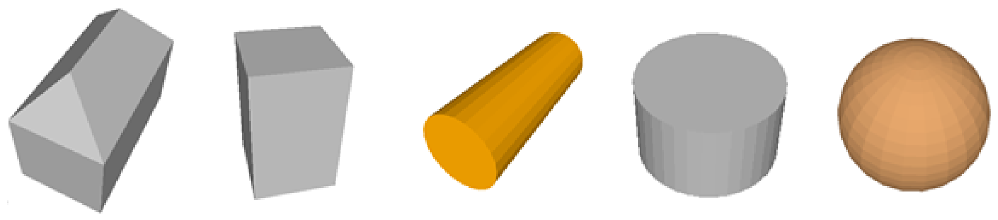


Figure 5.1: The original primitives used in the simulation. From left to right: gabbled house, cuboid, cylinder1, cylinder2, sphere.

Table 5.1: Parameters of scene objects (unit: meter)

Object	Translation (X,Y,Z)	Scale	Rotation (X,Y,Z)	
Cuboid1	4.04, -10.95, 0.34	1.06	0.0, -0.0, 0.0	
Cuboid2	-7.57, -22.29, 0.34	1.08	0.0, -0.0, 0.0	
House1	10.04, 10.69, -2.08	1.57	0.0, -0.0, 0.0	
House2	-38.05, -8.64, -2.04	1.52	0.0, -0.0, -179.73	
Object	Radius	Center	PointA	PointB
Sphere	4.0	-25, 13, 0.0	n/a	n/a
Cylinder1	4.0	13,15,2.5	13, 15, 0	13, 15, 5
Cylinder2	2.0	15, 0.0, 2	10, -3, 2	20, 3, 2

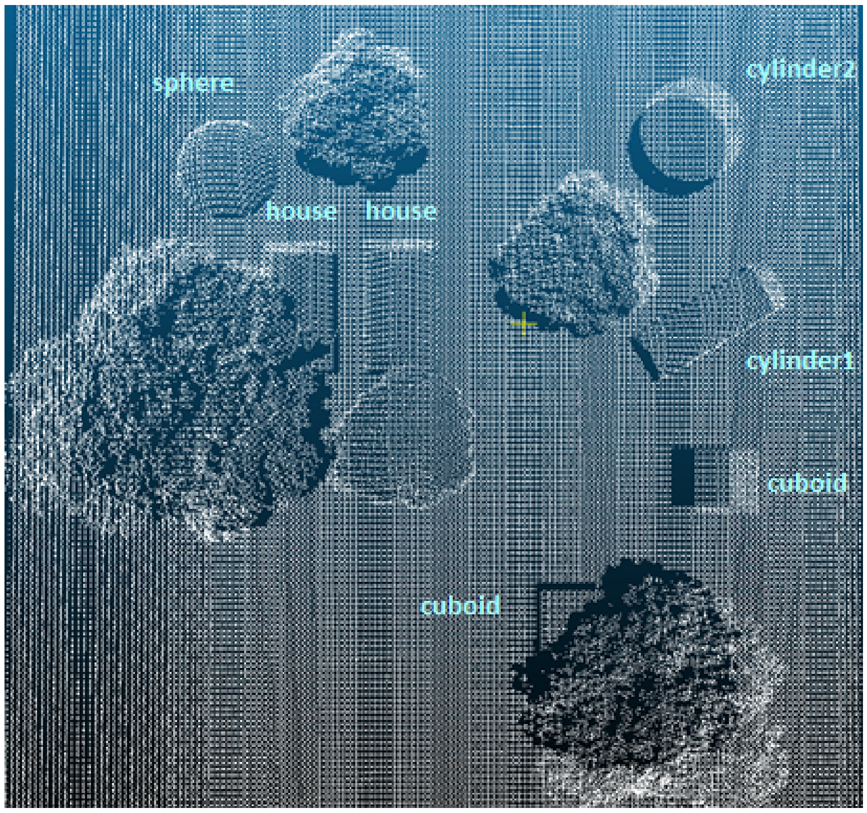


Figure 5.2: Simulated point cloud using DIRSIG, and all the objects are labeled to indicate their locations.

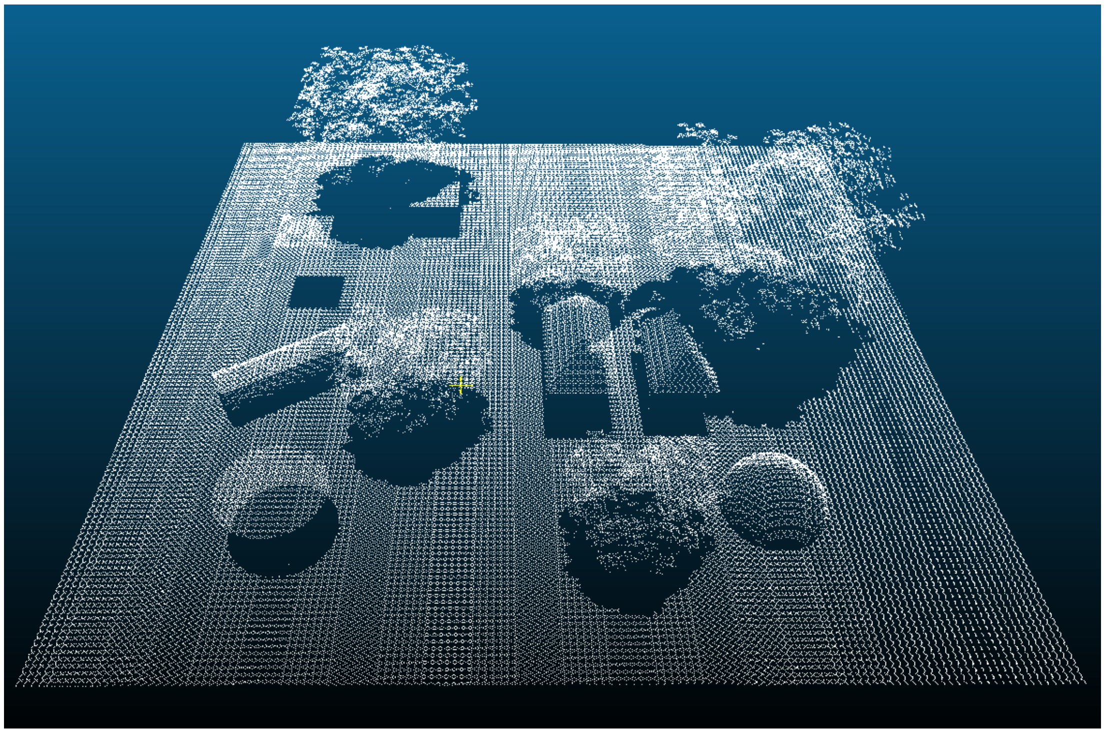


Figure 5.3: Simulated point cloud using DIRSIG (continued).

5.3 Result of Scene Classification

There are five vegetative areas in the simulated point cloud. The graph cuts based approach has been applied to detect these areas. The result is shown in Fig.5.5. In order to prove that the graph cuts based approach is necessary for the acceptable result, a further experiment has been conducted with the approach based on simple thresholding. Unlike the more sophisticated graph cuts based approach, the thresholding based one examines the histogram of the variation of point normals distribution and the makes a one-time judgment. According to some prior knowledge of the coverage of the vegetations in the scene or some experimental experience, a threshold value is picked to putatively separate tree points from non-tree points. Fig.5.4 shows the histogram of the variation of point normals distribution in which the majority of the points have small values. It can actually be anticipated by observing the scene content (a large number of points are on the flat ground). This histogram is used to derive a threshold (the red line in Fig.5.4) which helps to separate tree and non-tree points. Fig.5.6 shows the corresponding test result achieved by thresholding. This type of detector becomes more and more aggressive with the decreasing threshold value. As we can see, it is not good enough to precisely separate tree and non-tree points. The algorithm starts to pick up non-tree points before it finishes detecting the tree points. It explains why nearly half of the tree points are not detected, and all the points on the roof ridges are already mis-recognized as tree points using the specific threshold value in the figure.

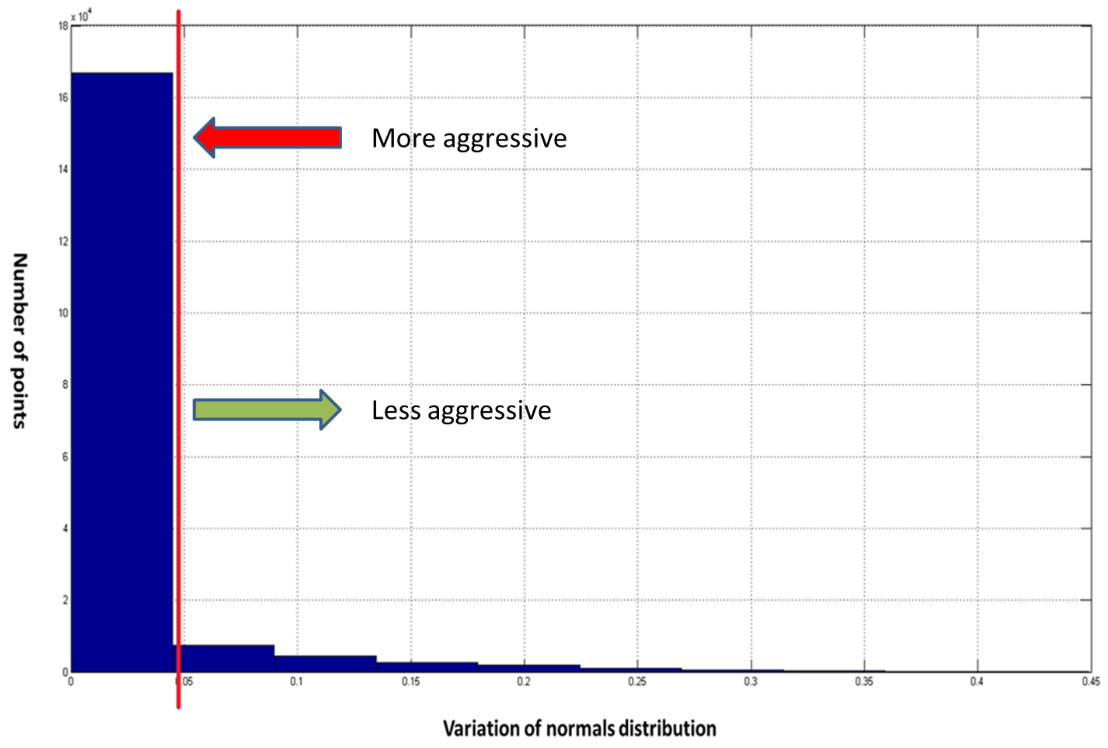


Figure 5.4: Histogram of the variation of point normals distribution in the simulated scene. Red line indicates where the threshold is. Increasing the threshold value leads to less aggressive result (green arrow), while decreasing the threshold value leads to more aggressive result (red arrow).

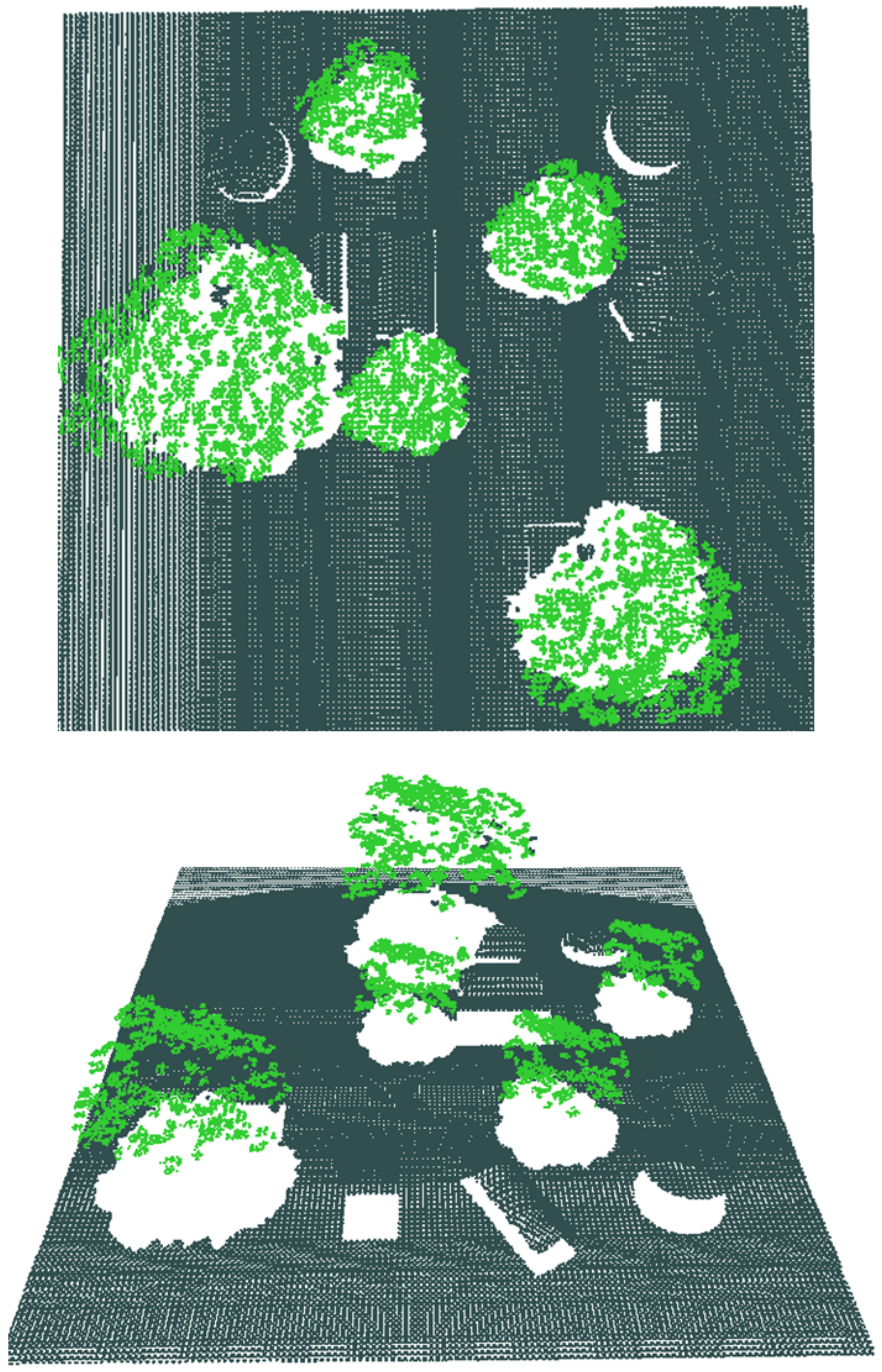
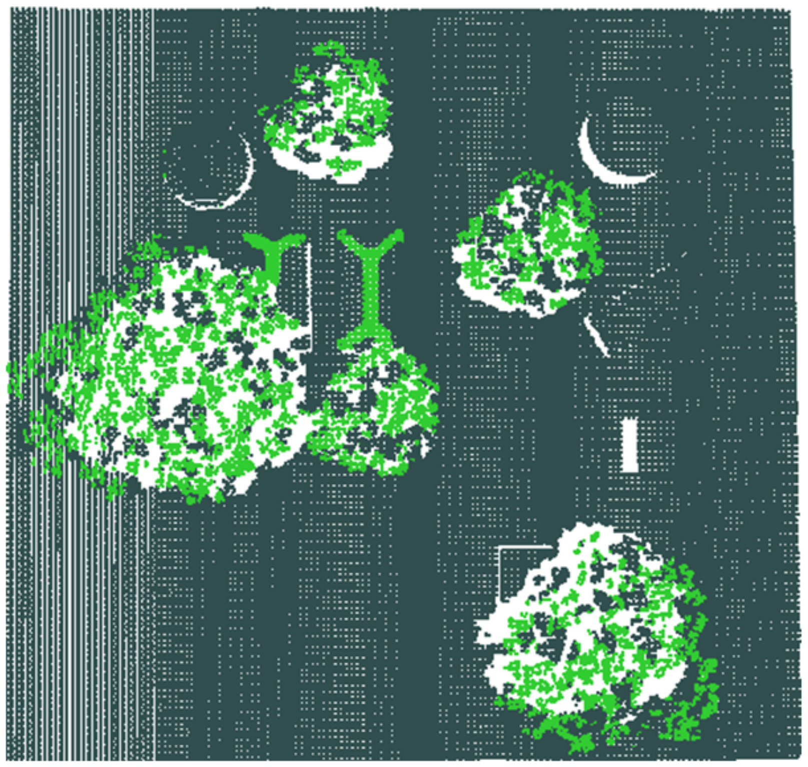


Figure 5.5: Result of scene classification using graph cuts based approach.



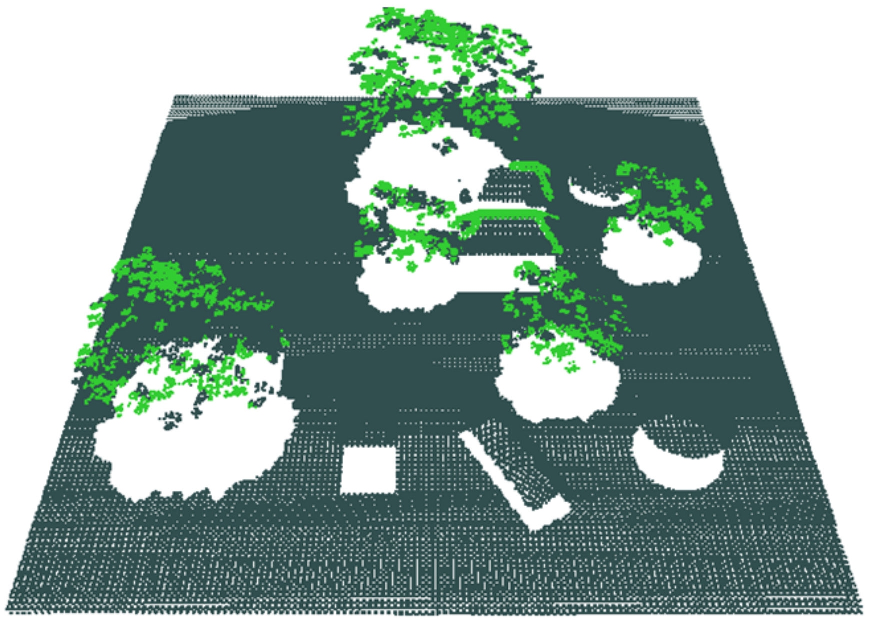


Figure 5.6: Result of scene classification using simple threshold based approach.

5.4 Result of Terrain and Footprints Detection

After the trees are removed from the scene, only the man-made objects and the terrain are left. Two-pass Euclidean clustering (see Chapter.3) is applied to separate terrain and each individual building footprint. The result is shown in Fig.5.7 and Fig.5.8.

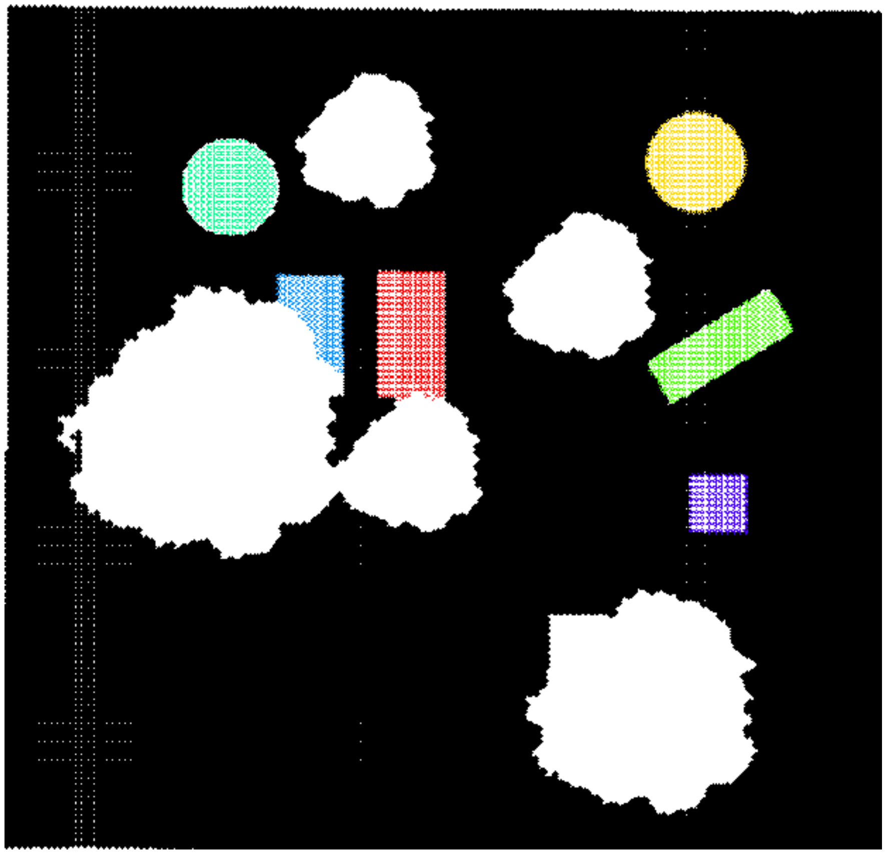


Figure 5.7: Result from two-pass Euclidean clustering: terrain (black), man-made objects (other colors).

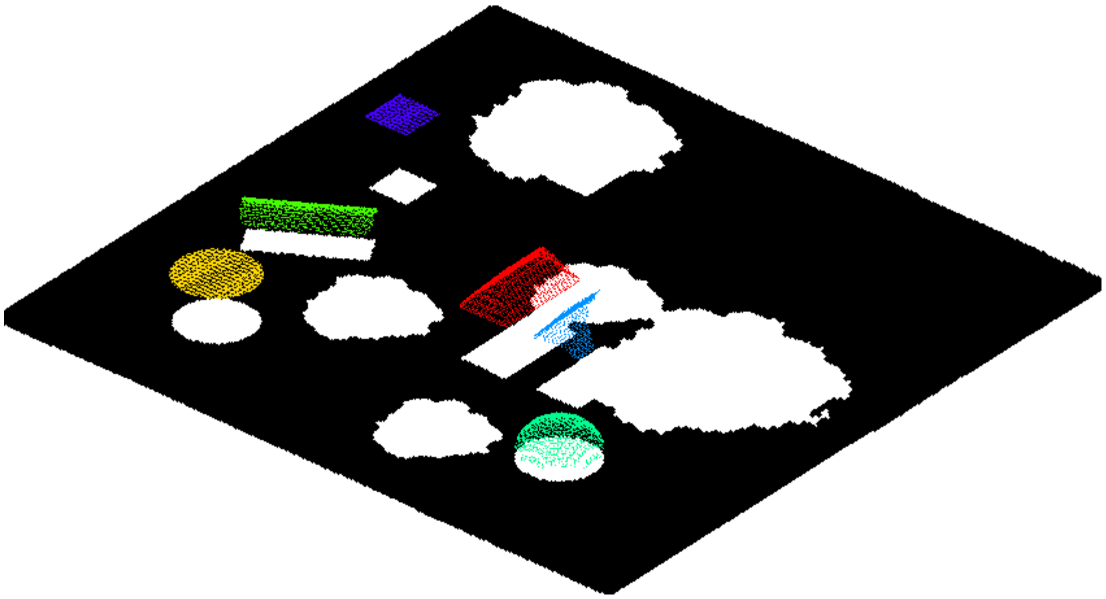


Figure 5.8: Result from two-pass Euclidean clustering: terrain (black), man-made objects (other colors) (continued).

5.5 Roof Feature Detection and Various Shape Fitting - An Extended Discussion

This section is an extended discussion about the implemented region growing based roof feature detection algorithm. There are a variety of geometric shapes existing in the simulated scene, including sphere and cylinders which have non-flat surfaces. Therefore, extracting these non-planar surfaces successfully becomes more significant issue than previously attempted. In Section.3.6 that introduces rooftop feature detection, the discussion was mainly focusing on extracting and fitting planar surfaces on the rooftop. There is a strong reason for that since most features on rooftops are flat surfaces. However, it does not imply that the region growing based roof feature detection algorithm is only limited to planar surfaces. As matter of fact, due to the nature of the algorithm, it can cluster points on non-flat or curved surfaces as well. RANSAC can then be applied to fit a parameterized geometric shape to those non-flat points. Schnabel *et al* [84] presented an automatic algorithm to detect some primitive shapes from point clouds. Their method aims at detecting planes, spheres, cylinders, cones and tori. It tries to convert the original point cloud into a set of primitive shapes with corresponding disjoint sets of points and some remaining points. The authors also provide the software implementation that the readers can test on their own data. However, it is still extremely difficult to find the right parameter set. The purpose in this research also tries to detect various geometric shapes from the point cloud, but in a different way. Once a rooftop feature is segmented from the roof point set, instead of only fitting a plane to the point set every time, other types of shape fitting, such as spherical and cylindrical, are also attempted in order to choose the best shape description for this target rooftop feature. Before moving forward, it is necessary to address the models of least square fitting for spheres and cylinders [85] which will be used in the RANSAC framework.

Sphere: a sphere can be defined by its center point (x_c, y_c, z_c) and radius r_0. All the points that are on the sphere satisfy the equation:

$$(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2 = 0, i = 1, 2, 3..., n \tag{5.1}$$

The minimization function for the estimation of a sphere is chosen to be:

$$f = r_i^2 - r_0^2 \tag{5.2}$$

where $r_i^2 = (x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2$. The linearization of the equation is done by expanding f:

$$f = -2(x_i x_0 + y_i y_0 + z_i z_0) + (x_i^2 + y_i^2 + z_i^2) + k \tag{5.3}$$

where $k = (x_0^2 + y_0^2 + z_0)^2 - r_0^2$. Let's rewrite it as:

$$\begin{bmatrix} -2x_1 & -2y_1 & -2z_1 & 1 \\ -2x_2 & -2y_2 & -2z_2 & 1 \\ : & : & : & : \\ -2x_n & -2y_n & -2z_n & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ k \end{bmatrix} - \begin{bmatrix} x_1^2 & y_1^2 & z_1^2 \\ x_2^2 & y_2^2 & z_2^2 \\ : & : & : \\ x_n^2 & y_n^2 & z_n^2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ : \\ f_n \end{bmatrix} \tag{5.4}$$

For a least square solution, $f = 0$. The whole equation can be notated as:

$$Ap - B = 0 \tag{5.5}$$

A sphere is uniquely defined by at least four points, so the cardinality of the minimal sample data is 4.

Cylinder: a cylinder can be defined by a point (x_0, y_0, z_0) on its axis and the direction of the axis (d_1, d_2, d_3) and the radius r. Let (x_i, y_i, z_i) be any point on the cylinder, then

$$[d_3(y_i - y_0) - d_2(z_i - z_0)]^2 + [d_3(z_i - z_0) - d_3(x_i - x_0)]^2 + [d_2(x_i - x_0) - d_1(y_i - y_0)]^2 = r_0^2 \tag{5.6}$$

This equation can be simplified as:

$$p_1x^2 + p_2y^2 + p_3z^2 + p_4xy + p_5xz + p_6yz + p_7x + p_8y + p_9z + p_{10} = 0 \tag{5.7}$$

where $p_1 = (d_2^2 + d_3^2), p_2 = (d_1^2 + d_3^2), p_3 = (d_1^2 + d_2^2), p_4 = -2d_1d_2, p_5 = -2d_1d_3, p_6 = -2d_2d_3, p_7 = -2(d_2^2 + d_3^2)x_0 + 2d_1d_2y_0 + 2d_1d_3z_0, p_8 = -2(d_1^2 + d_3^2)y_0 + 2d_1d_2x_0 + 2d_2d_3z_0, p_9 = -2(d_1^2 + d_2^2)z_0 + 2d_1d_3x_0 + 2d_2d_3y_0, p_{10} = (d_2^2 + d_3^2)x_0^2 + (d_1^2 + d_3^2)y_0^2 + (d_1^2 + d_2^2)z_0^2 - 2d_2d_3y_0z_0 - 2d_1d_3z_0x_0 - 2d - 1d_2x_0y_0 - r_0^2$.

This can be written as a linear system:

$$\begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & x_1y_1 & x_1z_1 & y_1z_1 & x_1 & y_1 & z_1 & 1 \\ x_2^2 & y_2^2 & z_2^2 & x_2y_2 & x_2z_2 & y_2z_2 & x_2 & y_2 & z_2 & 1 \\ : & : & : & : & : & : & : & : & : & : \\ x_n^2 & y_n^2 & z_1^2 & x_ny_n & x_nz_n & y_nz_n & x_n & y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \end{bmatrix} = A\bar{p} = 0 \tag{5.8}$$

A cylinder is uniquely defined by at least five points, so the cardinality of the minimal sample data is 5.

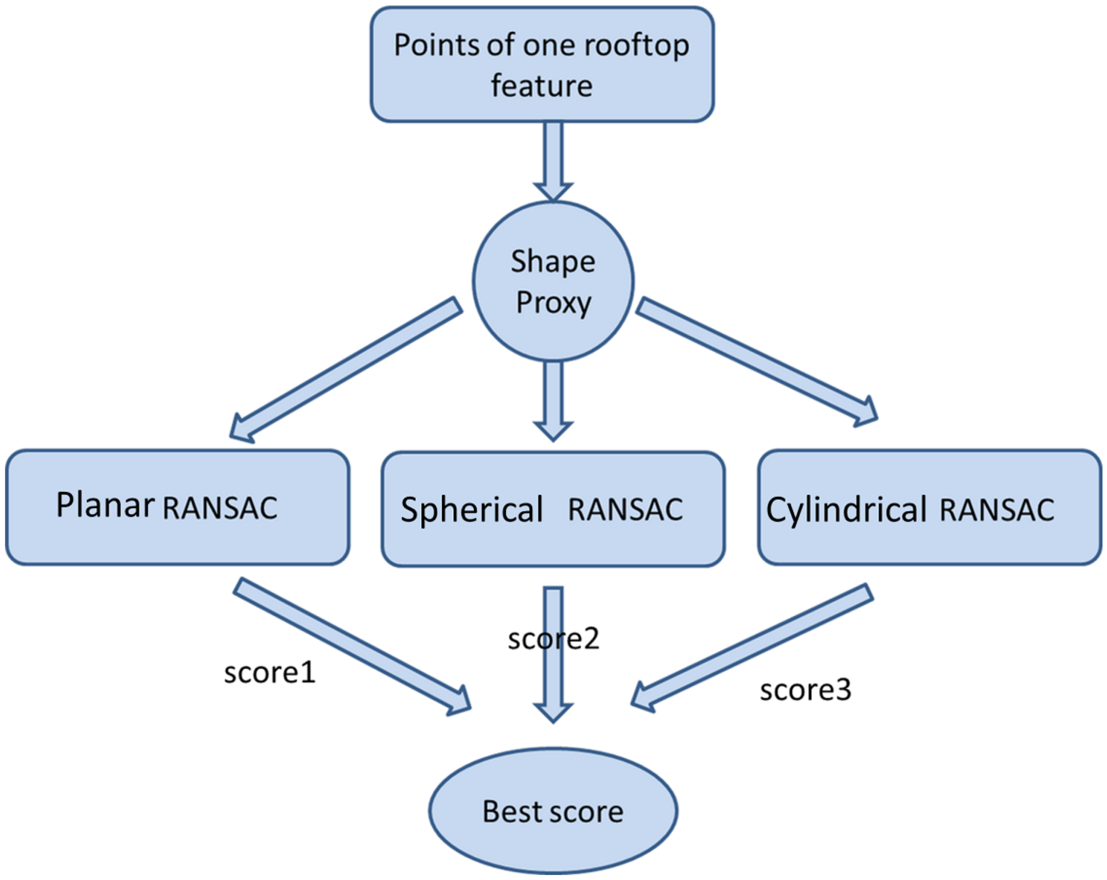


Figure 5.9: A shape proxy is introduced to test different types of RANSAC fitting on the point set of the rooftop feature. The one with best score (minimum error) is finally chosen.

In Fig.5.9, unlike what has been introduced in the Chapter 3, a shape proxy is brought into the shape fitting process. The proxy initiates three different types of RANSAC shape fitting, including planar fitting, spherical fitting, and cylindrical fitting. Each fitting gives a score back which can be directly or indirectly related to the error generated during the fitting process. The fit that provides the best score, or the minimum fitting error, is considered the best shape for the points being considered.

5.6 Results of Rooftop Feature Detection and Modeling

Fig.5.10 and Fig.5.11 show the result of detecting roof features from each footprint. The hipped rooftop in Fig.5.10 has four faces that were successfully extracted. The other footprints including non-planar shapes and round shape are classified as single feature since the smoothness of these surfaces are still high.

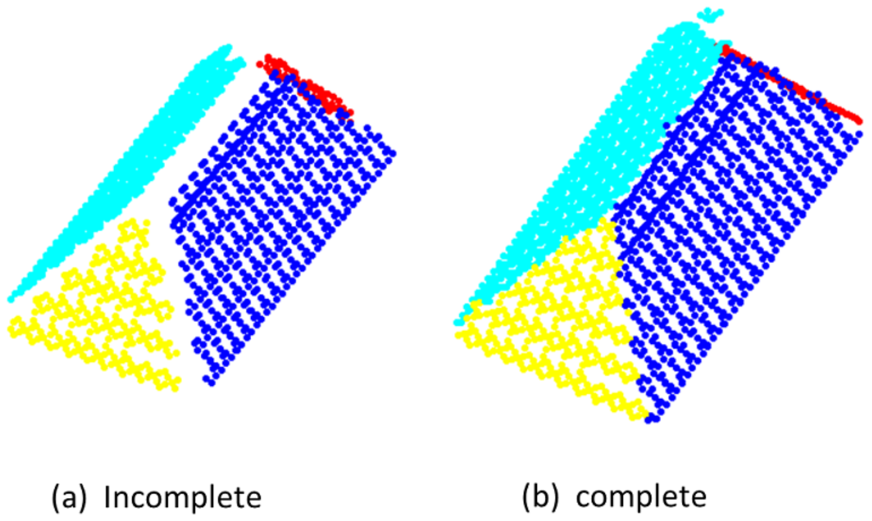


Figure 5.10: The simple hipped roof type, four faces are detected in (a) and completed in (b).

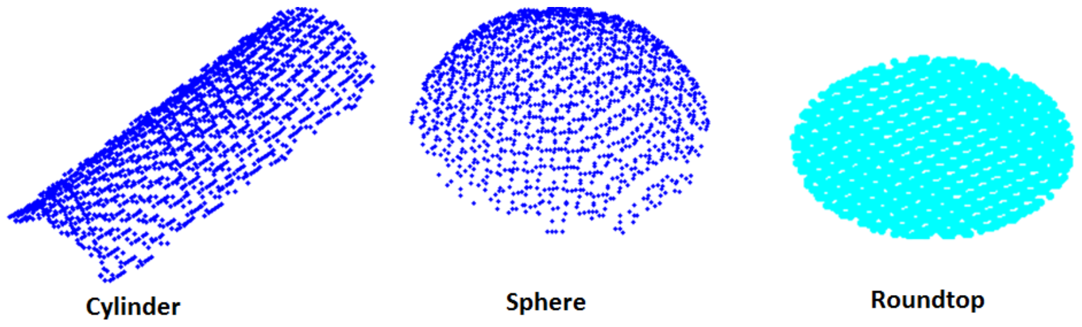


Figure 5.11: Cylinder, sphere and round top detected by the region growing based method. The unplanar smooth surfaces are detected as a single feature.

Fig.5.12 and Fig.5.13 present the final product of whole simulated scene reconstruction. One of the two houses is not complete since it was occluded by trees. Detailed accuracy and error analysis will be provided in next section.

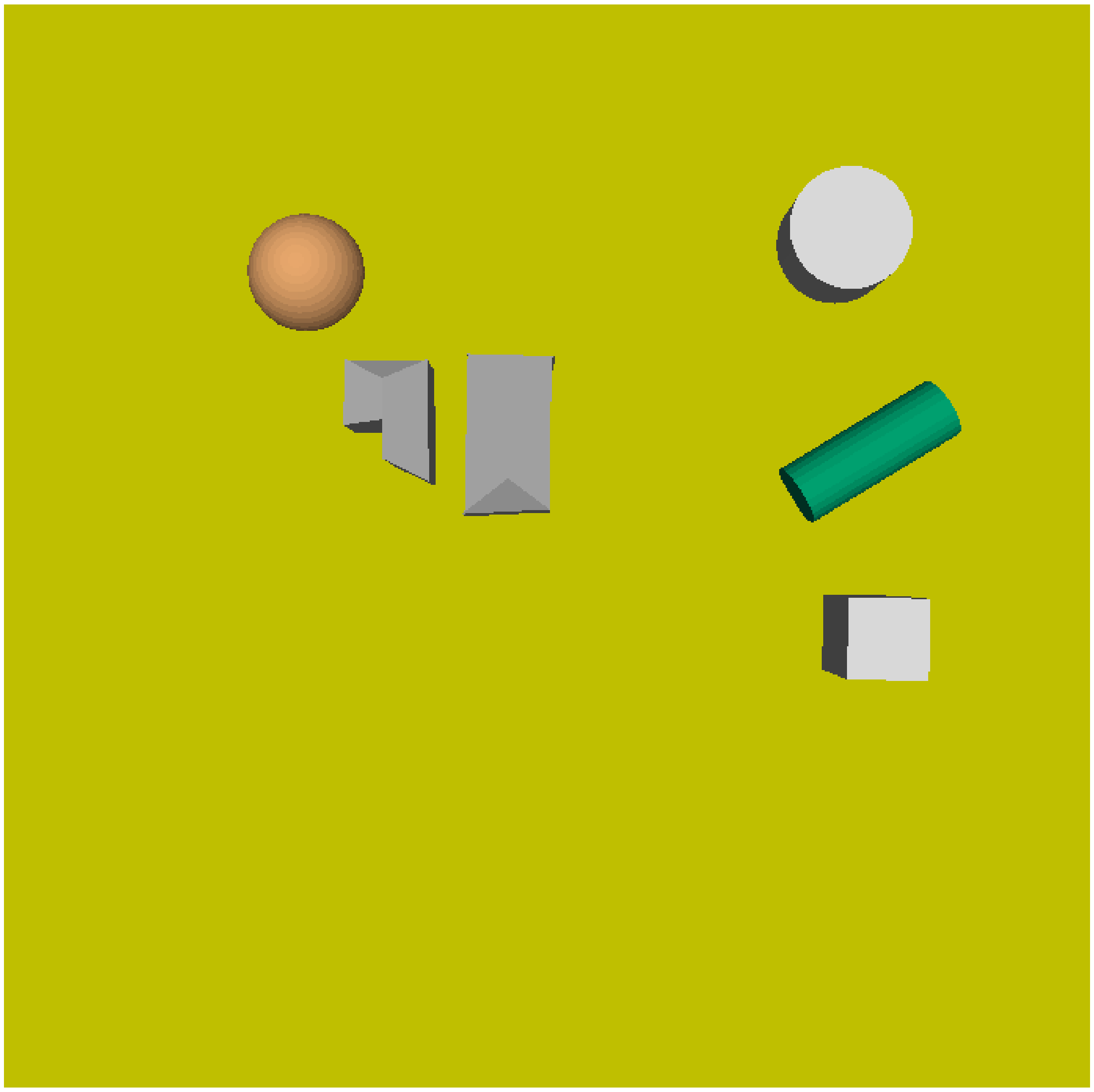


Figure 5.12: The final modeling product of the DIRSIG simulated scene, including the ground and objects (view1).

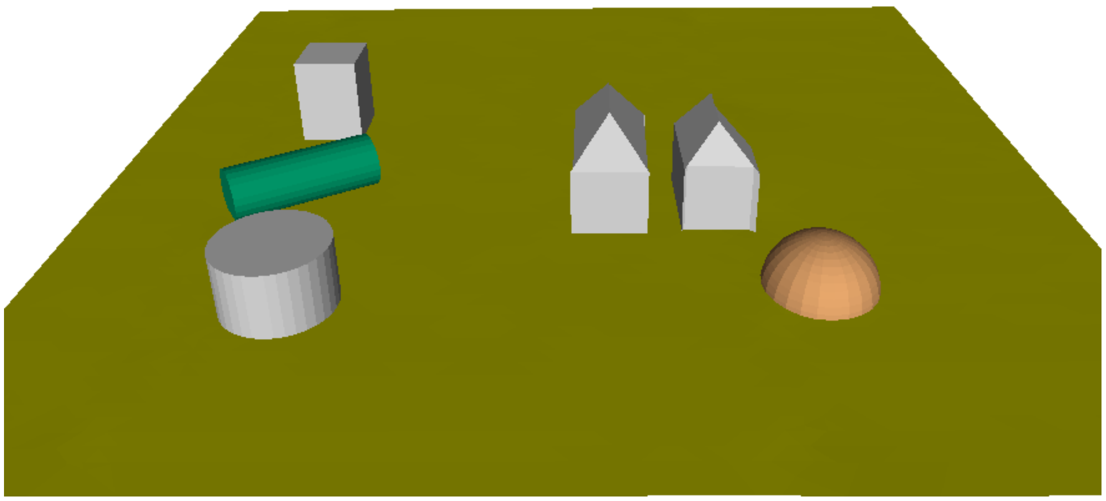


Figure 5.13: The final modeling product of the DIRSIG simulated scene, including the ground and objects (view2).

5.7 Analysis of Modeling Errors

Measuring the geometric accuracy of the final model is the key to evaluate the performance of the workflow. 3D urban models can be used for a variety of business purposes. Providing visual impact is highly likely to be the most important one in many current applications. Thus, visual inspection is the most popular evaluation method and the appearance of the model primarily conveys the quality. However, future applications will utilize these data to make mensuration. Measurements and absolute accuracy will become a critical criteria to understand for these approaches.

For the house and the cuboid, the optimal way to analyze the error is to measure the distance error between two mesh models. One is the ground truth model, and the other is the reconstructed model from the pipeline. Calculating the geometric difference between two models is actually a common task. Hausdorff Distance metric is used often for this purpose. Practically, a group of sampled points over a mesh A (reconstructed) is used to compute the distance, and for each sampling point in A, the closet point on a mesh B (ground truth) is searched. The result is strongly sensitive to the number of points that could be picked over mesh A. In this section, the original vertices of the reconstructed model are used as sample points. For each of them, it searches the closet vertex in the ground truth model. To better visualize the error, a 3D colored map can be generated based on the distance values.

For the 3D error map, the Red-Green-Blue color-map is chosen. Red indicates the minimum error, and blue indicates the maximum error. The color transition from red to blue indicates the error becomes larger. Fig.5.14 shows the ground truth and reconstructed models for the house object in the simulated scene. Fig.5.15 illustrates the error map of the reconstructed house model. In this result, the minimum error is 0.008967 *m*, the max error is 0.453570 *m* and the mean error is 0.151710 *m*. Fig.5.16 shows the ground truth and reconstructed models for the cuboid object in the simulated scene. Fig.5.17 illustrates the error map of the reconstructed cuboid model . In this result, the minimum error is 0.208124 *m*, the max error is 0.347489 *m* and the mean error is 0.282055 *m*. The publicly available CloudCompare tool is used here to assist in the visualization of colored error map.

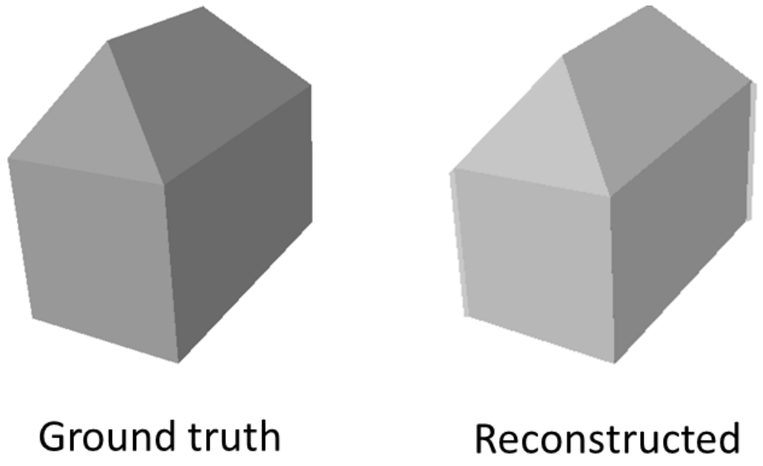


Figure 5.14: Ground truth model and the reconstructed model of the house.

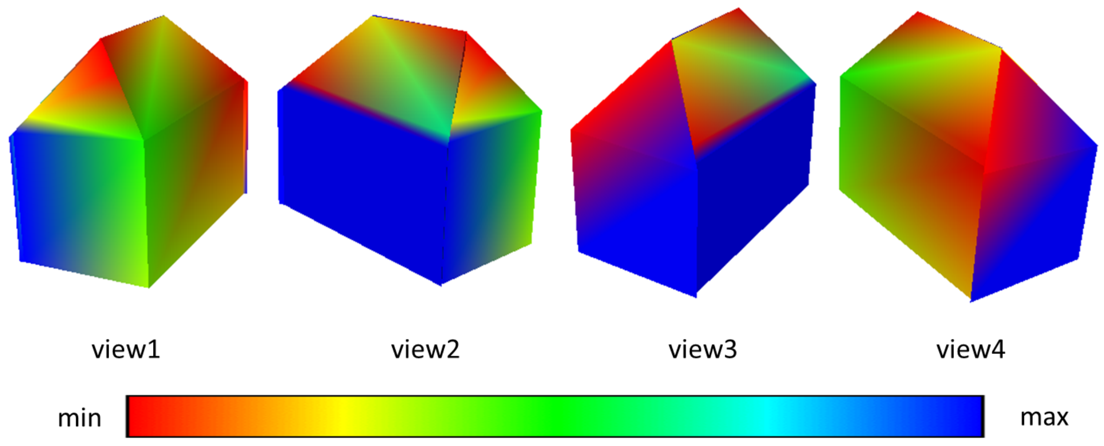


Figure 5.15: The error color-map of the house object. The error value goes up from the red to the blue (min: 0.008967 m, max: 0.453570 m, mean: 0.151710 m).

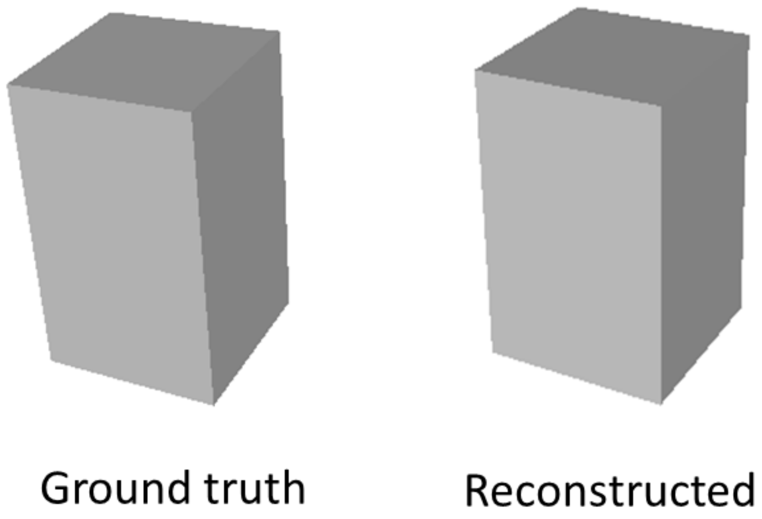


Figure 5.16: Ground truth model and the reconstructed model of the cuboid.

Table 5.2: Accuracy Evaluation (unit: meter)

	Ground truth		Modeled		
Object	Radius	Center	Radius	Center	Accuracy (%)
Sphere	4.00	-25.00, 13.00, 0.0	4.21	-25.00, 12.99, -0.31	94.75
Cylinder1	2.00	15.00, 0.0, 2.0	2.22	14.99, 0.01, 1.65	93.1
Cylinder2	4.00	13.00, 15.00, 2.5	4.19	14.98, 15.00, 2.47	95.25

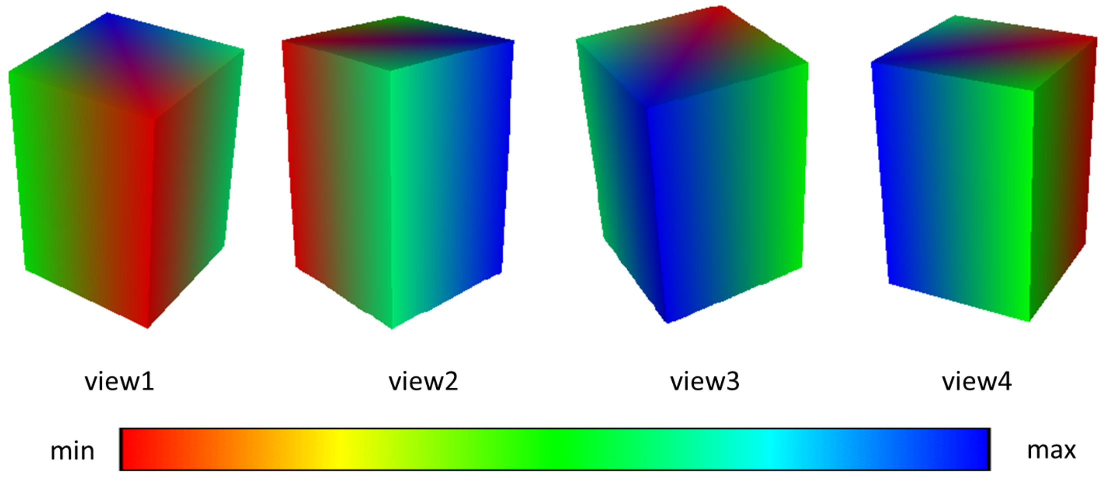


Figure 5.17: The error color-map of the cuboid object. The error value goes up from the red to the blue (min: 0.208124 m, max: 0.347489 m, mean: 0.282055 m).

For the rest of the parameterized models, such as sphere and cylinder, the optimal way to analyze the error is to compare the ground truth parameters and estimated parameters. Please refer to Table.5.7 for the details. The estimated values are products from RANSAC process. The average accuracy is 94.37%.

This chapter is designed to test the performance of the presented workflow. From scene classification to object modeling error analysis, it is fair to state the developed workflow generates fairly accurate modeling product.

Chapter 6

Summary

6.1 Conclusion

This research addresses the complicated problem of reconstructing urban models with a large variety of roof geometry and rich vegetation. The difference between buildings and trees is observed in terms of the implicit local surface characteristic. An original, automatic, complete pipeline has been developed for urban reconstruction from aerial LiDAR point clouds. The whole pipeline is composed of scene classification, building footprint and terrain detection, rooftop feature segmentation and final building modeling. There are a number of major contributions from this thesis work.

- The whole pipeline is a fast and fully automated system. No complicated setup is needed. Parameter settings are straightforward and easy to understand.
- Unsupervised graph cuts optimization on vegetation detection is developed using implicit surface properties only. No assistance from spectral information is required.
- A novel two-pass Euclidean clustering technique is developed to separate the terrain and building footprints.
- Dividing the rooftop into a group of major features using region growing-based segmentation technique provides the flexibility to explore detailed rooftop geometry in a thorough fashion. It can handle a wide variety of roof types from multi-layer urban rooftops to typical residential rooftops as well.

- A robust minimum bounding box searching (box rotating) technique is used to determine the major orientation of the rooftop. The rotating strategy guarantees that the discovered bounding rectangle is the one with minimum area.
- There is no library of rooftop shape templates required. It is common to define such a template library if the algorithm is designed to apply to a certain type of neighborhood, such as residential areas, where most of the rooftops are designed according to limited types of pattens. However, the intention in this work is to not restrict the applicability to some particular type of structures only. The approach can be applied to any type of urban or residential area.
- The developed pipeline provides reliable scalability with respect to the size and the variety of the input scene.

6.2 Limitations

There are other components in an urban scene other than buildings and terrain. Trees, bridges, and wire poles are just a few of these components. Trees are removed and not modeled in this work. Bridges that contain local planar or smooth surfaces are simply detected as buildings. Wire poles and wires might be or might not be detected, since the statistical noise removal may recognize these points as outliers and then remove them before any further processing is done.

Without the support from the ground level input, knowledge of building facades is not accessible. Important structures on facades, such as overhangs, recessed windows, bay windows, can not be recovered by this approach. It is a quite common issue when dealing with aerial laser scanned data. The only solution to this problem is to integrate both aerial and ground collections.

If a portion of the rooftop is occluded by another structure, such as an overhanging tree while collecting the data, then the information of the rooftop is incomplete and the scanned point set results in an incomplete reconstruction.

LiDAR point clouds convey weaker information on object corners compared to photogrammetric data. During the modeling process, the accuracy of corner approximation is still limited, especially when the point density is fairly low.

One last system-related issue, which is not actually the limitation of the approach is that a laser pulse typically is not able to deliver returns from certain transparent materials, such as glass. Thus, regions made of these type of transparent or non-reflective materials do not appear in LiDAR point clouds. It brings difficulty to the modeling process to derive geometry there.

6.3 Recommended Future Work

In the future, calibrated optical imagery should be introduced into this workflow in order to improve the geometry and provide texture on the models. Registration between these two different modalities is supposed to be conducted before any further processing. The advantages of using optical imagery include getting more accurate edge and corner estimations by applying edge detection and corner detection which are easier to determine in the image domain. For example, spectral information contained within the image can also help classify vegetative areas.

Ground level data, including both LiDAR and imagery, can be involved to provide the capability of modeling facade details. Particular objects, such as chimneys, bridges, wire poles, can be reconstructed by defining additional man-made structures besides buildings and designing special methods for the modeling of these particular shapes.

Currently, one global point density value is used for each LiDAR dataset. Local point density value can be exploited in order to provide adaptivity for the nearest neighbor searching. This can improve the accuracy of point normal estimation and also the Euclidean clustering for building footprint and terrain extraction.

The accuracy of merging minor regions into major regions during the region growing based rooftop feature detection can be improved by operating multiple minor regions at once instead of treating them independently to prevent minor regions from two distinct object components from being merged into a single component.

The efficiency of the workflow could be improved by implementing parallelization with the help of multi-core processors. Building footprint files can be processed in parallel. Within one building rooftop, boundaries of all the major features on the top can be calculated in a parallel fashion as well.

Appendix A

Proof

A.1 Relationship Between Sum of Squared Difference and Cross Correlation

In this section, the relationship between the sum of squared difference (SSD) and the cross correlation are derived.

Assuming there is an image $\mathbf{I}$, $\mathbf{T}$ is the template and the summation over positions (x, y) under the template that is located at (s, t), the SSD is defined as:

$$d(s,t) = \sum_{(x,y)} (\mathbf{I}(x,y) - \mathbf{T}(x-s, y-t))^2 \tag{A.1}$$

The cross correlation is defined as:

$$c(s,t) = \sum_{(x,y)} \mathbf{I}(x,y) \cdot \mathbf{T}(x-s, y-t) \tag{A.2}$$

The definition of SSD can be expanded as:

$$d(s,t) = \sum_{(x,y)} (\mathbf{I}(x,y)^2 - 2\mathbf{I}(x,y) \cdot \mathbf{T}(x-s, y-t) + \mathbf{T}(x-s, y-t)^2) \tag{A.3}$$

Obviously, the term $\sum_{(x,y)} \mathbf{T}(x-s, y-t)^2$ is a constant value, and the term $\sum_{(x,y)} \mathbf{I}(x,y)^2$ can be approximately assumed constant as well. So the remaining term

$-2\sum_{(x,y)} \mathbf{I}(x,y) \cdot \mathbf{T}(x-s, y-t)$ is $-2c(s,t)$.

Both SSD and the cross correlation are the measure of the similarity between the image and the template. The larger the value of c is or the smaller the value of SSD is, the more similar the image and the template are.

Appendix B

Source Code Examples

B.1 MATLAB Script of the Automatic Workflow

```
function runProcess
% parameters for noise removal
conf.nrmean = 10;
conf.nrstd  = 1.5;
conf.num_neighbor = 15;

% parameters for scene classfication
conf.lambda1  = 1e2;
conf.lambda2  = 1e4;
conf.C        = 1e2;

conf.min_cluster_size = 500;
conf.gheight = 130;

% input file
pointfile = 'subcamp3';

% region grow based binary file
if isunix
    binaryfile = '!/cis/phd/sxs4643/Research/rooftop_seg/build/roofseg';
elseif ispc
    binaryfile = '!D:\3DResearch\merge\rooftop_seg\rooftop_seg\Debug\
        rooftop_seg.exe';
else
```

```
    display('Unidentified machine type.')
end
conf.binaryfile = binaryfile;

% parameters for rooftop feature detection
conf.pts_in_rooffeature = 50;
conf.theta = '10';
conf.residual = '0.015';
conf.radius = '2';

conf.outfolder = ['regions_' pointfile];
if ~exist(conf.outfolder, 'dir')
    mkdir(conf.outfolder);
end

% run the whole work flow
start(pointfile, conf);
end
```

B.2 C++ Code for Region Growing Based Rooftop Feature Detection

```
template <typename PointT>
void sun::growRegion<PointT >::computeRegionGrowingSegment()
{
  std::cout<<"Region growing method to extract features"<<std::endl;
  int num_of_pts = static_cast<int> (cloud_in->points.size());

  std::pair<int, float> residual;

  points_flag.resize(num_of_pts, false);

  for (int i = 0; i < num_of_pts; i++)
  {
    residual.first = i;
    residual.second = normals->points[i].curvature;
    points_curvatures.push_back(residual);
    curvatures_tbl.push_back(normals->points[i].curvature);
  }

  points_curvatures.sort(comp_func);

  int num_seg_pts = 0;
  int current_pt_ind = -1;
  int current_seed = - 1;
  int pt_count = 0;
  std::queue<int> seed_list;
  std::vector<int> current_neighbors;
  std::vector<int> current_region;

  while (num_seg_pts < num_of_pts)
  {
    current_pt_ind = points_curvatures.front().first;
    points_curvatures.pop_front();

    while (points_flag[current_pt_ind] == true)
    {
      current_pt_ind = points_curvatures.front().first;
      points_curvatures.pop_front();
    }

    if (current_pt_ind == -1)
    {
      std::cout<<"No point is obtained from the point list"<<std::endl;
      return;
    }

    current_region.clear();
    current_region.push_back(current_pt_ind);
    // clean the current seed list, there is no clear() function of a
        queue
    seed_list = std::queue<int>();
    seed_list.push(current_pt_ind);
    points_flag[current_pt_ind] = true;

    while (!seed_list.empty())
    {
      current_seed = seed_list.front();
      if (current_seed == -1)
      {
        std::cout<<"No seed is obtained from the seed list"<<std::endl;
        return;
      }
      seed_list.pop();
      current_neighbors.clear();
      current_neighbors = point_neighbors[current_seed];

      // the first neighbor in the neighbor list is the target point
          itself
      for (int i = 1; i < static_cast<int>(current_neighbors.size()); i++)
      {
        float theta = ang_btw_normals(normals, current_seed,
            current_neighbors[i]);

        if (points_flag[current_neighbors[i]] == false && theta <=
            theta_th)
        {
          current_region.push_back(current_neighbors[i]);
          points_flag[current_neighbors[i]] = true;

          if (curvatures_tbl[current_neighbors[i]] <= r_th)
          {
            seed_list.push(current_neighbors[i]);
          }
        }
      }
    }

    num_seg_pts = num_seg_pts + static_cast<int>(current_region.size());
    seg_regions.push_back(current_region);
  }
  return;
}
```

Appendix C

DIRSIG ODB File

```
DIRSIG_ODB = 1.0

OBJECT {
    OBJ_FILENAME = cube.obj
    UNITS = METERS
    INSTANCES {
        INFO = 4.04007577896, −10.9473018646, 0.338058382273,
            1.05978679657, 1.05978679657, 1.05978679657, 0.0, −0.0, 0.0
        INFO = −7.5735783577, −22.2937908173, 0.338058382273,
            1.08196341991, 1.08196341991, 0.551539897919, 0.0, −0.0, 0.0
    }
}

OBJECT {
    GDB_FILENAME = Dogwood_1.gdb
    UNITS = METERS
    INSTANCES {
        INFO = −27.2771320343, −6.22774839401, −0.103100538254,
            2.51196026802, 2.51196026802, 2.51196026802, 0.0, −0.0, 0.0
        INFO = −10.2956752777, −9.51372528076, −0.103100538254,
            1.28399705887, 1.28399705887, 1.28399705887, 0.0, −0.0, 0.0
        INFO = 3.38845062256, 4.95235157013, 0.450744509697,
            1.36396813393, 1.36396813393, 1.36396813393, 0.0, −0.0, 0.0
        INFO = −13.944776535, 16.8119277954, −0.103100538254,
            1.25114524364, 1.25114524364, 1.25114524364, 0.0, −0.0, 0.0
        INFO = 8.52807044983, −29.0171470642, −0.103100538254,
            2.06550955772, 2.06550955772, 2.06550955772, 0.0, −0.0, 0.0
    }
}

OBJECT {
    OBJ_FILENAME = house.obj
    UNITS = METERS
    INSTANCES {
        INFO = 10.035027504, 10.6866502762, −2.07783985138, 1.56635761261,
             1.56635761261, 1.56635761261, 0.0, −0.0, 0.0
        INFO = −38.0517349243, −8.63604354858, −2.04403448105,
            1.52244532108, 1.52244532108, 1.52244520187, 0.0, −0.0,
            −179.730717971
    }
}

OBJECT {
    OBJ_FILENAME = terrain.obj
    UNITS = METERS
    INSTANCES {
        INFO = 0.0, 0.0, 0.0, 5.0, 5.0, 1.0, 0.0, −0.0, 0.0
    }
}

SPHERE {
      RADIUS = 4.0
      CENTER = −25, 13, 0.0
      MATERIAL_IDS = 4
}

CYLINDER {
    POINT_A = 13, 15, 0
    POINT_B = 13, 15, 5
    RADIUS = 4.0
    CAP_A = TRUE
    CAP_B = TRUE
    MATERIAL_ID = 4
}

CYLINDER {
    POINT_A = 10, -3, 2
    POINT_B = 20, 3, 2
    RADIUS = 2.0
    CAP_A = TRUE
    CAP_B = TRUE
    MATERIAL_ID = 4
}
```

Bibliography

[1] Li, Z., Zhu, Q., and Gold, C., [*Digital terrain modeling: principles and methodology*], CRC (2004).

[2] Hartley, R. and Zisserman, A., [*Multiple view geometry in computer vision*], vol. 2, Cambridge Univ Press (2000).

[3] Wolf, P. and DeWitt, B., [*Elements of Photogrammetry with Applications in GIS*], McGraw-Hill (2000).

[4] Leberl, F., Irschara, A., Pock, T., Meixner, P., Gruber, M., Scholz, S., and Wiechert, A., "Point clouds: Lidar versus 3d vision," *Photogrammetric Engineering and Remote Sensing* **76**(10), 1123–1134 (2010).

[5] Fischler, M. A. and Bolles, R. C., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM* **24**, 381–395 (June 1981).

[6] Zuliani, M., "Ransac for dummies," *With examples using the RANSAC toolbox for Matlab and more* (2009).

[7] Torr, P. and Zisserman, A., "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding* **78**(1), 138–156 (2000).

[8] Boykov, Y. and Veksler, O., "Graph cuts in vision and graphics: Theories and applications," *Handbook of Mathematical Models in Computer Vision* , 79–96 (2006).

[9] Ford Jr, L., Fulkerson, D., and Ziffer, A., "Flows in networks," *Physics Today* **16**, 54 (1963).

[10] Boykov, Y., Veksler, O., and Zabih, R., "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **23**(11), 1222–1239 (2001).

[11] Boykov, Y. and Kolmogorov, V., "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **26**(9), 1124–1137 (2004).

[12] Kolmogorov, V. and Zabin, R., "What energy functions can be minimized via graph cuts?," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **26**(2), 147–159 (2004).

[13] Sun, S. and Salvaggio, C., "Complex building roof detection and strict description from lidar data and orthorectified aerial imagery," in [*Proceedings of IEEE, IGARRS 2012, Analysis Techniques: Image Processing Techniques, Feature Detection in Images*], IEEE (July 2012).

[14] Lach, S. and Kerekes, J., "Robust extraction of exterior building boundaries from topographic lidar data," in [*Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*], **2**, II–85, IEEE (2008).

[15] Luebke, D., "A developer's survey of polygonal simplification algorithms," *Computer Graphics and Applications, IEEE* **21**(3), 24–35 (2001).

[16] Douglas, D. and Peucker, T., "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization* **10**(2), 112–122 (1973).

[17] Zhao, Z. and Saalfeld, A., "Linear-time sleeve-fitting polyline simplification algorithms," in [*Proceedings of AutoCarto*], **13**, 214–223 (1997).

[18] Hu, J., You, S., and Neumann, U., "Approaches to large-scale urban modeling," *Computer Graphics and Applications, IEEE* **23**(6), 62–69 (2003).

[19] Musialski, P., Wonka, P., Aliaga, D., Wimmer, M., van Gool, L., Purgathofer, W., Mitra, N., Pauly, M., Wand, M., Ceylan, D., et al., "A survey of urban reconstruc-

tion," in [*Eurographics 2012-State of the Art Reports*], 1–28, The Eurographics Association (2012).

[20] Haala, N. and Kada, M., "An update on automatic 3d building reconstruction," *ISPRS Journal of Photogrammetry and Remote Sensing* **65**(6), 570–580 (2010).

[21] Maas, H.-G. and Vosselman, G., "Two algorithms for extracting building models from raw laser altimetry data," *ISPRS Journal of Photogrammetry and Remote Sensing* **54**(23), 153 – 163 (1999).

[22] Weidner, U., "An approach to building extraction from digital surface models," *International Archives of Photogrammetry and Remote Sensing* **31**, 924–929 (1996).

[23] Brunn, A. and Weidner, U., "Extracting buildings from digital surface models," *International Archives of Photogrammetry and Remote Sensing* **32**(3 SECT 4W2), 27–34 (1997).

[24] Morgan, M. and Tempfli, K., "Automatic building extraction from airborne laser scanning data," *International Archives of Photogrammetry and Remote Sensing* **33**(B3/2; PART 3), 616–623 (2000).

[25] Wang, O., Lodha, S., and Helmbold, D., "A bayesian approach to building footprint extraction from aerial lidar data," in [*3D Data Processing, Visualization, and Transmission, Third International Symposium on*], 192–199, IEEE (2006).

[26] Lafarge, F., Descombes, X., Zerubia, J., and Pierrot-Deseilligny, M., "Automatic building extraction from dems using an object approach and application to the 3d-city modeling," *ISPRS Journal of Photogrammetry and Remote Sensing* **63**(3), 365 – 381 (2008).

[27] Matei, B., Sawhney, H., Samarasekera, S., Kim, J., and Kumar, R., "Building segmentation for densely built urban regions using aerial lidar data," in [*Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*], 1–8, IEEE (2008).

[28] Carlberg, M., Gao, P., Chen, G., and Zakhor, A., "Classifying urban landscape in aerial lidar using 3d shape analysis," in [*Image Processing (ICIP), 2009 16th IEEE International Conference on*], 1701–1704, IEEE (2009).

[29] Moons, T., Frère, D., Vandekerckhove, J., and Van Gool, L., "Automatic modelling and 3d reconstruction of urban house roofs from high resolution aerial imagery," in [*Computer Vision ECCV'98*], 410–425, Springer (1998).

[30] Kim, Z., Huertas, A., and Nevatia, R., "Automatic description of buildings with complex rooftops from multiple images," in [*CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*], **2**, II–272–II–279 (2001).

[31] Werner, T. and Zisserman, A., "New techniques for automated architectural reconstruction from photographs," *Computer VisionECCV 2002* , 808–809 (2002).

[32] Nevatia, R. and Price, K., "Automatic and interactive modeling of buildings in urban environments from aerial images," in [*Image Processing. 2002. Proceedings. 2002 International Conference on*], **3**, 525–528, IEEE (2002).

[33] Rau, J., Chen, L., and Wang, G., "An interactive scheme for building modeling using the split-merge-shape algorithm," *International Archives of Photogrammetry and Remote Sensing* **35**(B3), 584–589 (2004).

[34] Snavely, N., Seitz, S., and Szeliski, R., "Photo tourism: exploring photo collections in 3d," in [*ACM Transactions on Graphics (TOG)*], **25**(3), 835–846, ACM (2006).

[35] Snavely, N., Seitz, S., and Szeliski, R., "Modeling the world from internet photo collections," *International Journal of Computer Vision* **80**(2), 189–210 (2008).

[36] Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R., "A comparison and evaluation of multi-view stereo reconstruction algorithms," in [*Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*], **1**, 519–528, IEEE (2006).

[37] Elaksher, A. and Bethel, J., "Reconstructing 3d buildings from lidar data," *International Archives Of Photogrammetry Remote Sensing and Spatial Information Sciences* **34**(3/A), 102–107 (2002).

[38] Rottensteiner, F., "Automatic generation of high-quality building models from lidar data," *Computer Graphics and Applications, IEEE* **23**(6), 42–50 (2003).

[39] Verma, V., Kumar, R., and Hsu, S., "3d building detection and modeling from aerial lidar data," in [*CVPR 2006. Proceedings of the 2006 IEEE Computer Society Conference on*], **2**, 2213 – 2220 (2006).

[40] Zhou, Q.-Y. and Neumann, U., "Fast and extensible building modeling from airborne lidar data," in [*Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*], *GIS '08*, 7:1–7:8, ACM (2008).

[41] Zhou, Q., Ju, T., and Hu, S., "Topology repair of solid models using skeletons," *Visualization and Computer Graphics, IEEE Transactions on* **13**(4), 675–685 (2007).

[42] Zhou, Q. and Neumann, U., "A streaming framework for seamless building reconstruction from large-scale aerial lidar data," in [*Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*], 2759–2766, IEEE (2009).

[43] Dorninger, P. and Pfeifer, N., "A comprehensive automated 3d approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds," *Sensors* **8**(11), 7323–7343 (2008).

[44] Poullis, C. and You, S., "Automatic reconstruction of cities from remote sensor data," in [*CVPR 2009. Proceedings of the 2011 IEEE Computer Society Conference on*], 2775–2782, IEEE (2009).

[45] Toshev, A., Mordohai, P., and Taskar, B., "Detecting and parsing architecture at city scale from range data," in [*Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*], 398–405, IEEE (2010).

[46] Sampath, A. and Shan, J., "Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds," *Geoscience and Remote Sensing, IEEE Transactions on* **48**(3), 1554–1567 (2010).

[47] Duda, R., Hart, P., and Stork, D., "Pattern classification," *New York: John Wiley, Section* **10**, l (2001).

[48] Berkhin, P., "A survey of clustering data mining techniques," *Grouping multidimensional data* , 25–71 (2006).

[49] Verdie, Y., Lafarge, F., and Zerubia, J., "Generating compact meshes under planar constraints: An automatic approach for modeling buildings from aerial lidar," in [*Image Processing (ICIP), 2011 18th IEEE International Conference on*], 877–880, IEEE (2011).

[50] Garland, M. and Heckbert, P., "Surface simplification using quadric error metrics," in [*Proceedings of the 24th annual conference on Computer graphics and interactive techniques*], 209–216, ACM Press/Addison-Wesley Publishing Co. (1997).

[51] Lafarge, F. and Mallet, C., "Building large urban environments from unstructured point data," in [*Computer Vision, 2011 IEEE International Conference on*], 1068–1075, IEEE (2011).

[52] Lafarge, F. and Mallet, C., "Creating large-scale city models from 3d-point clouds: A robust approach with hybrid representation," *International Journal of Computer Vision* , 1–17 (2012).

[53] Fruh, C. and Zakhor, A., "Constructing 3d city models by merging aerial and ground views," *Computer Graphics and Applications, IEEE* **23**(6), 52–61 (2003).

[54] Frueh, C. and Zakhor, A., "Constructing 3d city models by merging ground-based and airborne views," in [*Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*], **2**, II–562, IEEE (2003).

[55] Wang, L. and Neumann, U., "A robust approach for automatic registration of aerial images with untextured aerial lidar data," in [*CVPR 2009. Proceedings of the 2009 IEEE Computer Society Conference on*], 2623 –2630 (june 2009).

[56] Ding, M., Lyngbaek, K., and Zakhor, A., "Automatic registration of aerial imagery with untextured 3d lidar models," in [*CVPR 2008.Proceedings of the 2008 IEEE Computer Society Conference on*], 1–8, IEEE (2008).

[57] Lowe, D., "Three-dimensional object recognition from single two-dimensional images," *Artificial intelligence* **31**(3), 355–395 (1987).

[58] Mastin, A., Kepner, J., and Fisher, J., "Automatic registration of lidar and optical images of urban scenes," in [*CVPR 2009. Proceedings of the 2009 IEEE Computer Society Conference on*], 2639–2646, IEEE (2009).

[59] Mastin, D. A., *Statistical methods for 2D-3D registration of optical and LIDAR images*, Master's thesis (2009).

[60] Pauly, M., [*Point primitives for interactive modeling and processing of 3D geometry*], Hartung-Gorre (2003).

[61] Muja, M. and Lowe, D. G., "Fast approximate nearest neighbors with automatic algorithm configuration," in [*International Conference on Computer Vision Theory and Application VISSAPP'09)*], 331–340, INSTICC Press (2009).

[62] Muja, M. and Lowe, D. G., "Fast matching of binary features," in [*Computer and Robot Vision (CRV)*], 404–410 (2012).

[63] Rusu, R. B., *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, PhD thesis, Springer (2009).

[64] Dey, T., Li, G., and Sun, J., "Normal estimation for point clouds: A comparison study for a voronoi based method," in [*Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*], 39–46, IEEE (2005).

[65] Berkmann, J. and Caelli, T., "Computation of surface geometry and segmentation using covariance techniques," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **16**(11), 1114–1116 (1994).

[66] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., "Surface reconstruction from unorganized points," *SIGGRAPH Comput. Graph.* **26**, 71–78 (July 1992).

[67] Rusu, R., Marton, Z., Blodow, N., Dolha, M., and Beetz, M., "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems* **56**(11), 927–941 (2008).

[68] Secord, J. and Zakhor, A., "Tree detection in urban regions using aerial lidar and image data," *Geoscience and Remote Sensing Letters, IEEE* **4**(2), 196–200 (2007).

[69] Charaniya, A., Manduchi, R., and Lodha, S., "Supervised parametric classification of aerial lidar data," in [*Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*], 30–30, IEEE (2004).

[70] Koch, B., Heyder, U., and Weinacker, H., "Detection of individual tree crowns in airborne lidar data," *Photogrammetric engineering and remote sensing* **72**(4), 357 (2006).

[71] Anguelov, D., Taskarf, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A., "Discriminative learning of markov random fields for segmentation of 3d scan data," in [*Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*], **2**, 169–176, IEEE (2005).

[72] Sedlacek, D. and Zara, J., "Graph cut based point-cloud segmentation for polygonal reconstruction," *Advances in Visual Computing* , 218–227 (2009).

[73] Golovinskiy, A. and Funkhouser, T., "Min-cut based segmentation of point clouds," in [*Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*], 39–46, IEEE (2009).

[74] Isenburg, M., Liu, Y., Shewchuk, J., Snoeyink, J., and Thirion, T., "Generating raster dem from mass points via tin streaming," *Geographic Information Science* , 186–198 (2006).

[75] Rabbani, T., van Den Heuvel, F., and Vosselmann, G., "Segmentation of point clouds using smoothness constraint," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **36**(5), 248–253 (2006).

[76] Chaudhuri, D. and Samal, A., "A simple method for fitting of bounding rectangle to closed regions," *Pattern recognition* **40**(7), 1981–1989 (2007).

[77] Lin, H., Gao, J., Zhou, Y., Lu, G., Ye, M., Zhang, C., Liu, L., and Yang, R., "Semantic decomposition and reconstruction of residential scenes from lidar data," *ACM Transactions on Graphics,(Proc. of SIGGRAPH 2013)* **32**(4) (2013).

[78] Chew, L. P., "Constrained delaunay triangulations," *Algorithmica* **4**(1-4), 97–108 (1989).

[79] Kazhdan, M., Bolitho, M., and Hoppe, H., "Poisson surface reconstruction," in [*Proceedings of the fourth Eurographics symposium on Geometry processing*], (2006).

[80] Zhou, Q.-Y. and Neumann, U., "2.5d dual contouring: A robust approach to creating building models from aerial lidar point clouds," in [*Computer Vision ECCV 2010*], 115–128 (2010).

[81] "http://meshlab.sourceforge.net/."

[82] "http://www.danielgm.net/cc/."

[83] "http://dirsig.org/."

[84] Schnabel, R., Wahl, R., and Klein, R., "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum* **26**, 214–226 (June 2007).

[85] Forbes, A. and National Physical Laboratory, G. B., [*Least-squares Best-fit Geometric Elements*], NPL report DITC, National Physical Laborarory (1991).