# Building Model Reconstruction from Point Clouds Derived from Oblique Imagery 

by<br>Ming Li

B.S. Wuhan University, 2011


#### Abstract

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Chester F. Carlson Center for Imaging Science

College of Science Rochester Institute of Technology


May, 2015

Signature of the Author $\qquad$

Accepted by

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

CERTIFICATE OF APPROVAL

## M.S. DEGREE THESIS

The M.S. Degree Thesis of Ming Li has been examined and approved by the thesis committee as satisfactory for the thesis required for the M.S. degree in Imaging Science

Dr. John P. Kerekes, Thesis Advisor

Dr. Carl Salvaggio

Dr. David Messinger

Date

# Building Model Reconstruction from Point Clouds Derived from Oblique Imagery 

by
Ming Li
Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science Degree
at the Rochester Institute of Technology


#### Abstract

The increasing availability of high resolution airborne imagery increases the accuracy of building modelling of urban scenes. This high accuracy of building modelling offers a strong reference for disaster recovery and asset evaluation. With the advantage of having more façade information, this thesis builds on previous efforts in building reconstruction from airborne oblique imagery.

Based on previous work, this thesis presents two schemes to construct building models from point clouds derived from oblique imagery. With the assumption that buildings are in a cubic-shape, the first scheme consists of three different steps. Plane estimation aims at identifying dominant surfaces; edge extraction helps in detecting and simplifying in-plane edges in each identified surfaces; model construction finishes the job of assembling the surfaces and edges together and producing a model in a universally accepted format. We find this scheme works well with complete point clouds that cover all sides of the building. A second method is proposed to handle


the complications when the point clouds do not cover all sides of the building. The main structure of the building is estimated using minimum bounding box on the dominant planes. The rest of the estimated planes are then attached to the main structure. The process can produce a water-tight building model.

The schemes are tested on point cloud data sets from multiple sources, including both image derived and lidar derived point clouds. The surface based approach and minimum bounding box based approach both show the capability of reconstructing models, while both of them have disadvantages. The limitations such as density of point clouds; fitting accuracy; and future work, including increasing efficiency and robustness, are also discussed.

## Acknowledgements

Pursuing my graduate studies at RIT has been one of the most important steps of my life and I have enjoy every day of it. Throughout the highs and lows, a lot of people have helped me along the way, and I would like to thank them.

First and foremost, my thesis advisor, Dr. John Kerekes. This thesis would not have been possible without the support and advice from Dr. Kerekes. His advice helped me get through the difficult days of the research, and his support guided me into the right career path. He is a more than just an advisor to me.

I am extremely grateful to my thesis committee; Dr. David Messinger and Dr. Carl Salvaggio. They were very helpful in influencing the direction of my dissertation work. They asked thought-provoking questions and made a number of very useful suggestions. I am grateful for their expertise and willingness to give advice.

This work has been completed with the support of Pictometry International. In addition to their greatly appreciated financial support, Dr. Yandong Wang from Pictometry International offered generous technical support and consultation throughout the years of the thesis work.

Many thanks go to my team in the project: Jie Zhang and Ming Zhang. Without them, the project wouldn't have progressed so quickly. I am truly appreciate their time and willingness to discuss the work with me.

I would like to thank RIT and Center for Imaging Science for the financial support for my graduate study. I would also like to thank Sue Chan for all the hard work to keep me on track with all of the paper work and registration. I'd also like to thank the many professors I had during my course work. I am grateful to have
joined a school with such a wealth of knowledge and talent.
I want to thank my family and friends. My parents' and my sister's support have helped me get through many obstacles in life. Their unyielding support keeps me working hard every day. It is my greatest honor to have such good friends like Jiashu Zhang, Bin Chen, Fan Wang, Mike, Brian etc. Thank you for keeping me sane and entertained during my studies.

## Contents

List of Tables ..... x
List of Figures ..... xi
1 Introduction ..... 1
1.1 Project Objectives ..... 2
1.2 Contributions to Knowledge ..... 4
1.3 Thesis Overview and Organization ..... 4
2 Background ..... 5
2.1 Oblique Imagery ..... 5
2.2 Point Cloud Data ..... 6
2.3 Computer Vision Theories ..... 8
2.3.1 Projective Geometry ..... 8
2.4 Parameter Estimation ..... 12
3 Previous Work ..... 17
4 Data ..... 20
4.1 Pictometry Data ..... 20
4.2 DIRSIG Data ..... 22
4.3 Other Point Cloud Data ..... 25
5 Single Surface Based Method ..... 26
5.1 Plane Estimation ..... 28
5.1.1 Classical RANSAC algorithm ..... 28
5.1.2 Region Growing ..... 31
5.1.3 Proposed Method ..... 33
5.2 Edge Extraction ..... 37
5.2.1 Determine in-plane Edges ..... 38
5.2.2 Line Simplification ..... 41
5.2.3 Line Regulation ..... 43
5.3 Model Extraction ..... 47
6 Minimum Bounding Box Based Method ..... 50
6.1 Minimum Bounding Box ..... 52
6.2 Proposed Approach ..... 53
6.2.1 Searching for Dominant Planes ..... 54
6.2.2 Histogram based Clustering ..... 55
6.2.3 Model Construction ..... 61
7 Results and Discussion ..... 63
7.1 Results on Adaptive RANSAC Algorithm ..... 63
7.1.1 Computational Efficiency ..... 64
7.1.2 Fitting Accuracy ..... 65
7.2 Results on the Edge related Approach ..... 68
7.2.1 Edge Identification ..... 69
7.2.2 Model Construction ..... 71
7.3 Results on Minimum Bounding Box related Approach ..... 72
7.4 Model Accuracy Validation ..... 75
8 Conclusions and Future Work ..... 80
Bibliography ..... 83

## List of Tables

5.1 The result of RANSAC estimation in terms of points on each plane. . 30
5.2 Processing time (seconds) comparison between classical and modified RANSAC algorithm in seconds.36
7.1 Processing time (seconds) comparison between original and adaptive RANSAC algorithm for DIRSIG building. ..... 64
7.2 Processing time (seconds) comparison between original and adaptive RANSAC algorithm for Airborne Oblique Imagery Data ..... 65
7.3 Fitting Accuracy Comparison betweeen RANSAC and Adaptive RANSAC for DIRSIG Building ..... 66
7.4 Fitting Accuracy Comparison (Meters) betweeen RANSAC and Adap- tive RANSAC for Airborne Oblique Imagery Data ..... 66
7.5 Ground Truth Corner Points vs Projected Corner Points ..... 77
7.6 RMS Error of four sides of the building ..... 79

## List of Figures

2.1 Oblique Imagery of RIT campus ..... 6
2.2 Point Cloud of RIT campus derived from oblique imagery ..... 7
2.3 Central Projection Geometry Example.[1] ..... 11
2.4 Line Fitting Comparison between Least Square and RANSAC.[2] ..... 13
4.1 Samples of Pictometry airborne oblique imagery ..... 21
4.2 Point Cloud generated from Pictometry oblique imagery ..... 22
4.3 Another Point Cloud generated from Pictometry oblique imagery ..... 23
4.4 Samples of DIRSIG generated images. ..... 24
4.5 Point Cloud generated from Pictometry oblique imagery. ..... 24
4.6 Examples of lidar based point clouds ..... 25
5.1 The overall scheme of the project workflow ..... 27
5.2 Example of a building that fits the assumption (Wallace Library of Rochester Institute of Technology) and its corresponding point cloud. ..... 27
5.3 Made-up point cloud example and plane estimation result of the data ..... 29
5.4 Reconstruction example of automobile C-pillar by Region Growing Algorithm[3] ..... 33
5.5 Results of Convex Hull algorithm on L-shape data. (a) the sample "L" shape data; (b) ideal edge extraction result; (b) edge extraction result from convex hull ..... 39
5.6 Example of Alpha Shapes in 2D[4], where blue dots represents point set S , green circle represents scoop with radius $\alpha$ and red line segments represent identified shapes ..... 40
5.7 Example of 2D Douglas-Peucker line simplification algorithm[5] ..... 42
5.8 Edge map of 2D imagery ..... 45
5.9 Vector Description of Edge Point Shifting. (A)the vector formed by the target points and its neighbors. (B) The theory of determining direction of shifting. ..... 46
5.10 An example of the OBJ file format. ..... 48
6.1 The problem with Pictometry data set, While points cover roof and front wall, walls on the other sides are missing. ..... 51
6.2 A preliminary model of library. ..... 51
6.3 An example of minimum bounding box, where blue points are the input points in 3D and red lines are the minimum bounding box ..... 52
6.4 Minimum Bounding Box algorithm results on (a) Pictometry data and (b) DIRSIG data ..... 53
6.5 Dominant plane search on different data sets (a) RIT library data set, (b) identified 2 dominant planes in red and green, (c) DIRSIG data set, (d) identified 5 dominant planes in different colors55
6.6 Two different situations of planes, in (a), 12 surface share the same plane, in (b) only one surface on the estimated plane ..... 56
6.7 Principal Component Analysis of one point set. (a)(b)(c) are the points projected into the orthogonal dimensions,(d)(e)(f) are the cor- responding histograms of the points in each dimension. ..... 58
6.8 Selecting cluster number by histogram (a) histogram of the input data in one dimension, (b) histogram after selecting local maxima, (c) his- togram after eliminating low maxima, (d) histogram after eliminating shallow maxima ..... 59
6.9 Noise reduction based on histogram. ..... 60
6.10 Clustering result of one plane. Each color represents one cluster ..... 61
7.1 Plane fitting result of a door as circled in red (top view). (a) result from original RANSAC, (b) result from adaptive RANSAC ..... 66
7.2 Surface Extraction results for both traditional and modified RANSAC algorithm for library building. (a) the airborne image of library build- ing; (b) the point cloud of the building; (c) classic RANSAC algorithm result; (d) modified RANSAC algorithm result ..... 67
7.3 Consensus Set Comparison of (a) traditional and (b) modified RANSAC algorithm on the point clouds of rooftop ..... 68
7.4 Edge detection results for one single surface. (a)edge points(red)detected by alpha-shape algorithm; (b) edges from Douglas-Peukeralgorithm; (c) Modified edges after edge correction from 2D image;
(d) modified edges after line simplification on corrected edge points ..... 70
7.5 Edge identification results on DIRSIG and lidar data sets. ..... 71
7.6 Model construction result on one DIRSIG derived point cloud data set 1. ..... 72
7.7 Model construction result on DIRSIG derived point cloud data set 2. ..... 73
7.8 Model construction result on oblique imagery derived point cloud data set. ..... 73
7.9 An example of missing information on the reconstructed model box ..... 74
7.10 An example of DIRSIG image with projected corner points ..... 76
7.11 Samples of DIRSIG generated images ..... 78

## Chapter 1

## Introduction

3D building models are becoming increasingly essential among urban planning, disaster management, emergency response, and other applications. Due to the rapid development of cities and the requirement of up-to-date information, semi- or completely automated modelling has emerged as an active research field. With the aid of computer vision techniques, this field of study has experienced a boost in recent years.

For decades, several different approaches based on various computer vision techniques have been developed. In this thesis, the focus is on the point-cloud based method. Generally, this method can be divided into two steps, point cloud extraction and model extraction. For point cloud extraction, the commonly used computer vision structure from motion (SfM) work flow is an adaptation of the well-known Bundler software written by Noah Snavley[6]. The imagery data goes through Scale Invariant Feature Transform (SIFT), Bundler, Patch-based Multi-view Stereo (P-

MVS), and produces camera information and point clouds in the relevant coordinate system[7]. In model extraction, various methods have been developed based on the variety of data sources and building shape.

Our primary data source is airborne oblique imagery. Compared to traditional nadir view imagery or active sensing such as lidar (Light Detection And Ranging) data, one of the most obvious advantages of oblique imagery is the information on building facades[8]. With this information in hand, we will be able to extract information on the building sides which is not possible to achieve with nadir data. This thesis project mainly focuses on model extraction from 3 dimensional point cloud data extracted from oblique airborne imagery. Due to the limited accessability of oblique imagery, the point clouds generated are not as dense as expected, and several sides are missing. The challenge of the project is thus to reconstruct a complete building model from the incomplete point cloud.

### 1.1 Project Objectives

As stated above, the ultimate goal of this thesis project is to extract building models from point cloud data in a semi- or completely automated process. To achieve this goal, the task is separated into several tasks that can be easily handled. These tasks together will accomplish the ultimate goal of constructing a building model. These several tasks are listed as follows:

1. Develop or adapt a method to estimate surfaces in the 3 D point cloud. The intent of this task is to estimate dominant surfaces in the point cloud and
identify corresponding points that belong to the related surface. Several estimation methods have been proposed in previous work. However, the approach is a case-bycase task due to the variety of data features and building structures. Large efforts are made to adapt an algorithm to suit our unique data set. Additional difficulties and issues in the estimation process are also discussed.

## 2. Detect edges in the estimated surfaces and adjust boundaries ac-

 cordingly. In order to get a building model from the surfaces estimated, one needs to outline the edges of each surface. The goal of this task is to detect the edges of the surfaces, approximate the boundaries and then adjust the boundaries based on the general geometry of the building structure. Edge detection in point cloud data is a relatively difficult process due to the randomness of the points. Thus, the effort has been mainly put into the edge approximation and linear regression. Because of the low density of point cloud data, another regularization process is proposed to make the edges align with the geometry of buildings.3. Construct building models. This task is to finish the ultimate objective of this research, that is to connect surface edges to form a building structure model. And then it will produce the model in a widely used format in the industry. Moreover, adding texture information of each surface can be a secondary goal of this task. This texture information can be extracted from the airborne imagery, including spatial detail and color information.

### 1.2 Contributions to Knowledge

This research provides a baseline workflow to reconstruct 3-dimensional building models from oblique imagery derived point cloud data. Although several approaches were proposed related to this topic, this task is aimed at the unique data we have. In the effort, a new adaptive RANSAC algorithm is proposed. Two different reconstruction approaches are developed to achieve the goal of reconstructing models from point clouds derived from oblique imagery. Also, this research demosntrates the possibility of produce 3-dimensional models from oblique airborne imagery.

### 1.3 Thesis Overview and Organization

The rest of this thesis is organized as follows. Basic concepts and background information in relation to this research are provided in Chapter 2. Chapter 3 introduces the previous work that is similar to the research we are conducting. Chapter 4 presents the data sets examined in the thesis work. Chapter 5 and 6 introduces the baseline of two proposed approaches and a detailed description of the algorithms used in the process. Results and discussions are shown in Chapter 7 as well as a description regarding the accuracy of the process. Chapter 8 includes a summary of the research and suggested future work.

## Chapter 2

## Background

### 2.1 Oblique Imagery

Oblique imagery is a type of aerial photography that is captured at a non-vertical angle with respect to the ground. Apart from orthographic imagery which mostly captures information from a nadir view, oblique imagery contains information on the sides as well as the top of buildings. It resembles closely how viewers see the landscape. Currently, oblique imagery are systematically captured in several cities by multiple companies including Pictometry[9]. Several applications of oblique imagery have been proposed. Hhle proposed to use a single oblique image to estimate object height[10]. Xiao et al. used multiple oblique images to detect buildings[8]. In 2009, Gerke discussed the possibility of 3D point cloud generation based on oblique imagery. The overlapping and multi-viewing features of oblique imagery make it possible to extract 3D point clouds[11].

The oblique images used in the research come from Pictometry International. Their aircraft flew over Rochester Institute of Technology (RIT) and captured the entire campus. Each image is about 4900x3200 pixels in size. Because of the GPS and IMU onboard, each picture is geo-referenced. In total, 11 oblique images are used to generate the point clouds used in this thesis. Figure 2.1 is an example of the oblique images captured by Pictometry.


Figure 2.1: Oblique Imagery of RIT campus

### 2.2 Point Cloud Data

A point cloud is a set of vertices representing multi-dimensional structure, and is most commonly used in 2D and 3D data. In 3D space, usually point cloud data is defined by $\mathrm{X}, \mathrm{Y}$ and Z geometric coordinates comprising an external surface of an
object. When color information like RGB components are available, the data turns 4D.

Point clouds can be generated from hardware like 3D scanners, stereo cameras, or from computer software. In this research, the source is airborne imagery and previous work has produced the point cloud structure of the entire scene [12]. Figure 4.2 shows the point cloud of the RIT campus generated from 10 airborne oblique images. In the data, geometric coordinates and RGB information are included, as well as a normal vector for each point.


Figure 2.2: Point Cloud of RIT campus derived from oblique imagery

### 2.3 Computer Vision Theories

Computer Vision is a discipline that tries to perceive our 3D world based on one or more 2D images. Different from traditional photogrammetry which acquires precise measurements of the scene, computer vision tends to pursue a more general understanding of the scene that requires less precise measurements. This difference makes the application of computer vision different from photogrammetry. Computer vision develops more into areas such as object recognition, motion detection, model construction, etc. Techniques in computer vision largely rely on pinhole camera theory to build and understand 3D object models. In this section, we will introduce some fundamental computer vision concepts that are used in this thesis.

### 2.3.1 Projective Geometry

This section will briefly introduce several fundamental concepts in terms of projective geometry that are widely used in modern computer vision technologies and also essential in this thesis work. A thorough discussion of all computer vision concepts is beyond the scope of this thesis. A more detailed description can be found in Hartley and Zisserman[1]. The rest of this section is primarily taken from this book. No further reference is presented in the rest of this discussion.

## Homogeneous Coordinates

The representation of points, lines, and planes in Euclidean space is the most popular method used. For instance, a point in 2D Euclidean space is presented as
$(x, y)$. It also can be considered as a vector representation of the point, $\mathbf{x}=(x, y)^{T}$ However, geometric entities like points and lines are treated differently in projective representations. Homogeneous coordinates are used, which represent entities only up to an arbitrary scaler multiplier. It means that a homogeneous representation of an entity is not unique. Any entities $\mathbf{x}$ and kx point at the same thing. In this sense, an arbitrary homogeneous vector representative of a point in 2D projective space is $\mathbf{x}=(k x, k y, k)^{T}$, where k is a non-zero scaler. It represents the point $(x, y)$ in 2D Euclidean space.

A line is naturally represented by vector $(a, b, c)$ in accordance to the equation $a x+b y+c=0$ in 2D space. However the correspondence between lines and vectors is not one-to-one. Just like points, any vectors ( $k a, k b, k c$ ) with a non-zero scalar $k$ states the same line. This equivalence class of vectors offers us the homogeneous representation of lines in 2 D projective space, $\mathbf{l}=(a, b, c)^{T}$ With line representation, one can easily tell that a point $\mathbf{x}$ lies on the line $\mathbf{l}$ only if $\mathbf{x}^{T} \mathbf{l}=0$

In the same manner, in 3D projective space, a point is expressed as $\mathbf{x}=(k x, k y, k z, k)^{T}$ representing the point $\mathbf{x}=(x, y, z)^{T}$ in Euclidean space. A plane in 3D space can be described in the equation $a x+b y+c z+d=0$. Correspondingly, it can be represented in vector form as $(a, b, c, d)^{T}$ where $(a, b, c)^{T}$ describes the plane normal. Similar to lines in 2D space, any vector $(k a, k b, k c, k)^{T}$ with non-zero scalar $k$ describes the same plane. Therefore, a homogeneous representation of a certain plane is $\pi=(a, b, c, d)^{T}$. Again a point $\mathbf{x}$ is on the plane $\pi$ only if $\mathbf{x}^{T} \pi=0$. Up to this point, we can perform a linear projective transform in 3D homogeneous space, $\mathbf{X}^{\prime}=\mathbf{H X}$, where $\mathbf{H}$ is a projection matrix that has 15 degrees of freedom. A plane
under the same projection is transformed to be $\pi^{\prime}=H^{-T} \pi$.

## Central Projection

With the introduction above, we can start to describe the basic geometry of a pinhole camera model. Here we assume the image plane is in front of the projection center as seen in Figure 2.3. In this simple model, the projection center, $\mathbf{O}$, is the origin of the local coordinate system; the plane $Z=f$ is the image plane. Under the pinhole camera model, a point in 3 D space $\mathbf{X}=(X, Y, Z)^{T}$ is mapped to the point $\mathbf{x}=\left(x_{c}, y_{c}\right)^{T}$ where a line connecting the point $\mathbf{X}$ and the origin meets the image plane. By similar triangles, one can easily calculate that $\mathbf{x}=\left(x_{c}, y_{c}\right)^{T}=$ $(f X / Z, f Y / Z)^{T}$. In the manner of homogeneous representation, the calculation can similarly be presented in matrix multiplication.

$$
\left[\begin{array}{c}
x_{c}  \tag{2.1}\\
y_{c} \\
1
\end{array}\right]=\left[\begin{array}{c}
f X / Z \\
f Y / Z \\
1
\end{array}\right]=\left[\begin{array}{cccc}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{array}\right]\left[\begin{array}{c}
X \\
Y \\
Z \\
1
\end{array}\right]
$$

One thing to note here is that this equation assumes the coordinate origin of the image plane is set at the principal point. In practice, it may not be the case. So for the purpose of generalization, another mapping which adds shift of principal point


Figure 2.3: Central Projection Geometry Example.[1]
is needed here. This leads to the following solution.

$$
\left[\begin{array}{c}
x_{c}  \tag{2.2}\\
y_{c} \\
1
\end{array}\right] \sim\left[\begin{array}{c}
f X+Z p_{x} \\
f Y+Z p_{y} \\
1
\end{array}\right]=\left[\begin{array}{cccc}
f & 0 & p_{x} & 0 \\
0 & f & p_{y} & 0 \\
0 & 0 & 1 & 0
\end{array}\right]\left[\begin{array}{c}
X \\
Y \\
Z \\
1
\end{array}\right]=K[I \mid \mathbf{0}] \mathbf{X}
$$

In a more general case, points in space are expressed in the world coordinate system other than camera coordinate system. These two systems are related through a rotation and a translation. In order to use the equations developed above, one
simply needs to calculate the coordinate position of the point $\mathbf{X}$ in camera coordinate system by the formula $X_{\text {cam }}=R(\mathbf{X}-\mathbf{C})$ where R is a $3^{*} 3$ rotation matrix representing the orientation of the camera coordinate frame, and $\mathbf{C}$ is the center of camera coordinate in the world coordinate frame. It can be expressed in homogeneous coordinates as

$$
X_{c a m}=\left[\begin{array}{cc}
R & -R \mathbf{C}  \tag{2.3}\\
0 & 1
\end{array}\right] \mathbf{X}
$$

This equation along with the equation (2.2) will offer the general pinhole camera mapping as follow.

$$
\begin{equation*}
\mathbf{x}=K R[I \mid-\mathbf{C}] \mathbf{X} \tag{2.4}
\end{equation*}
$$

One can see that a pinhole camera model, $\mathrm{P}=\mathrm{KR}[\mathrm{I} \mid-\mathrm{C}]$ has 9 degrees of freedom. The parameters in K are internal parameters describing internal orientation of the camera, and the parameters in R and $\mathbf{C}$ describes external parameters representing orientation and positions of the camera in the world coordinate system.

### 2.4 Parameter Estimation

Almost all computer vision problems involve parameter estimation, such as line fitting, motion analysis, and in our case, surface reconstruction. Traditional estimation approaches have strong premises. For instance, least square estimation (LSE) confines into a single population model[13] and assumes the noise distributed in a single pattern such as Gaussian. When the assumptions are not met, these approaches can turn out to have major error.

In most computer vision cases, the complicated structure separates the data into multiple populations and creates gross outliers. The sensitivity of traditional estimators to outliers makes it not ideal for these cases. The idea can be summarized in the example [2] below.


Figure 2.4: Line Fitting Comparison between Least Square and RANSAC.[2]

In LSE estimation, the estimator includes gross error points in the estimation. By doing so, the estimated line leans towards the gross error point, and eliminates points on the ideal line. After several iterations, the line as indicated in Figure 2.4 is closer to the gross error point than points on the correct line.

This situation urged the computer vision community to shift focus to robust es-
timators. Ideally, robust estimators remove the effect of outliers on final estimation. Several robust estimation techniques have been developed over recent years. The two most important techniques that were developed independently are the hough transforms [14] and RANdom SAmple Consensus (RANSAC)[2]. Hough transforms are basically a voting procedure. In the so-called parameter space, each data point votes for the parameters with the acceptable small fitting residual. Then the space is searched to locate a maxima. One disadvantage of hough transforms is that the voting space increases exponentially which makes it computationally impractical in many cases [13].

This research thus utilizes the RANSAC technique as the primary approach to handle parameter estimation problems. RANSAC offers another perspective in removing outliers. Instead of trying to use as much data as possible for estimation as in a least square approach, RANSAC tries to find the parameter with the least outliers. It first starts by estimating parameters with minimum data points necessary and then evaluates the points that are within a predefined error as inliers. The algorithm iterates the previous process until a minimum number of inliers is achieved or a maximum number of iterations is reached. The estimation with the maximum number of inliers is considered as the ultimate estimation. In detail, RANSAC algorithm can be explained in the pseudo code below.

```
Algorithm 1 RANSAC
    Definition:
    S:The data set that need to be estimated
    n : Minimum number of points needed to estimate the parameters
    k : Maximum number of iteration allowed
    d : Minimum number of inliers to accept an estimation
    Start
1. Randomly Select n points to estimate a model using these points
2. Determine consensus set Si of points that are within the error threshold
if The size of \(S i\) is larger than d or the iteration exceeds k then Re-estimate model using \(S i\)
end if
if The size of \(S i\) is smaller than d and iteration is smaller than k then return to step 1
After certain trials, return the largest set of \(S i\) and re-estimated model
end if
```

One thing worth noting is that there are only four parameters that need to be specified. N is determined by the model that one wants to estimate. The parameter k should be large enough so that there is a high probability of acquiring a large consensus set.

In the example above, the RANSAC result shows its advantage over the least square approach. It identifies the gross error point as outlier. The fitting result is
better than least square estimator and includes the minor error point as inlier.
In this research and previous work, RANSAC has been used in multiple cases. It is used to fit planes to 3 -dimensional data points in this research. Moreover, it is able to estimate multiple planes by analyzing remaining outliers from a previous step. In an early stage of the project, RANSAC was used to eliminate poorly matched points from SIFT results. In this thesis, we develop an adaptive RANSAC algorithm to efficiently estimate planes.

## Chapter 3

## Previous Work

As mentioned earlier, surface reconstruction is a case-by-case project. The methods are differentiated by the type of target, the type of data, density of point clouds, the availability of other useful information of the target, etc. There is no absolutely effective algorithm that can reconstruct all cases. However, there are several directions that reconstruction research has explored.

One of the most popular data sources in building reconstruction is lidar data. It offers a high density point cloud that can be easily identified. Taking advantage of this fact, Turner et al. [15] reconstructed a single surface by using robust least square interpolation. Normal vectors were utilized when trying to reconstruct complicated rooftop structures in Verma et al.'s work [16]. However the drawback of aerial lidar point clouds is that it is almost impossible to reconstruct side walls because it mostly contains only nadir view information. Frequently, algorithms tend to extrude rooftop outlines and extend them to the ground [16]. Recently researchers started to use 2D
imagery to acquire more information and to assess the accuracy as a reference. By referencing with 2D imagery, it will ease the work of edge identification. Using lidar point clouds and a building topographic map, Rey-Jer You and Bo-Chen Lin [17] successfully outlined edges and registered the clouds with a 2 D topographic map. Further, Wang et al. [18] used information from 2D imagery to refine edges in a region growing process along with lidar data and retrieved texture of the surfaces in the model.

2D imagery alone is another principle source for data reconstruction. Researchers started to use 2D imagery to construct building models before lidar data was available. It developed along with the improvement in multi-view geometry theory. Carlos Tomasi and Takeo Kanade [19] proposed an early method of utilizing affine fabrication to extract 3D features from multiple 2D frames. Later on, because of the increasing popularity of different types of digital imagery, new extraction methods were developed. Again, it becomes a case-dependent problem. In 1998, Frere et al. proposed an early method based on edge detection results of 2D imagery in nadir view [20]. This approach has the same limitation as lidar data. It cannot offer side information. This research area enjoyed a tremendous boost in the last two decades with multiple directions to approach the problem. Most recently, Maurer et al developed a method which utilizes multiple overlapping images from an aerial vehicle platform and publicly available GIS information to create geo-referenced 3D model of buildings [21]. An approach combining probabilistic volumetric estimation with smooth signed distance estimation was proposed by Calakli et al. [22] to produce a detailed model of large urban scenes.

Although the algorithms mentioned above successfully produced building models, most of them still requires a human-involved process such as selecting matching points. The goal of this thesis and related work is to find a fully automated approach to produce point clouds from 2D oblique imagery and generate 3D building models.

## Chapter 4

## Data

Point cloud data are a major component of computer vision data types, and have been widely used in the scope of 3D and 2D applications. In the scope of this thesis, point clouds from multiple sources are used to test and validate the algorithm. Meanwhile, the focus is still on the point clouds generated from oblique imagery.

### 4.1 Pictometry Data

Pictometry Data includes oblique imagery from five different perspectives, north, south, east, west, and nadir respectively. Figure 4.1 shows some samples of the collected imagery. The site in the imagery is the campus of the Rochester Institute of Technology (RIT), including various buildings, parking lots, and vegetation. The resolution of the images are $3248 x 4872$, taken at the altitude of approximately 1400 m .

The point cloud data is generated from Jie Zhang's work[12]. It follows the work


Figure 4.1: Samples of Pictometry airborne oblique imagery
flow established at RIT. The imagery goes through feature detection and matching algorithms, and finally reprojects back to 3D space and forms the point cloud data. Figure 4.2 below gives an example of the point cloud data of RIT campus from one perspective.

Several other Pictometry data sets are provided. Figure 4.3 is another point cloud sample generated from airborne images of the height of 800 m .


Figure 4.2: Point Cloud generated from Pictometry oblique imagery

### 4.2 DIRSIG Data

In order to validate the robustness of the approach, the algorithms need to be tested on multiple data from different sources. Another dataset that is used in the research is provided by Katie Salvaggio [23]. The data set was created with RITs Digital Imaging and Remote Sensing Image Generation (DIRSIG) software[24]. It provides high-fidelity radiometric data and also 3D location and surface normals for each pixel in an image scene. Figure 4.4 shows an example of the scene that is generated from DIRSIG. It includes multiple buildings with different structure and also vegetation. The images of the simulated scene were taken at the altitude of 800 m above ground, with a focal length of 125.09 mm . The camera is set to be slightly tilted, thus offering an oblique view of the scene.

The data set comes with minimum and maximum range, corresponding hit co-


Figure 4.3: Another Point Cloud generated from Pictometry oblique imagery
ordinates, and normal coordinates. Using this information, a point cloud data set can be created with a free space based algorithm. Figure 4.5 shows a sample of the point clouds generated. One set of point cloud corresponds to one single image with each pixel corresponding to a point in the 3D coordinate system.

Because of the fact that the data comes from ground truth images with known 3D information, the point cloud generated is noise free. It can serve as a benchmark data set for 3D reconstruction testing. By combining point clouds of different angles that cover four sides of a building, one complete point cloud data set of a building is accomplished.


Figure 4.4: Samples of DIRSIG generated images.


Figure 4.5: Point Cloud generated from Pictometry oblique imagery.

### 4.3 Other Point Cloud Data

For the purpose of testing the robustness of some parts of the algorithm, point cloud data from other sources are used as well. Specifically, lidar point clouds are used here. Lidar can produce a much denser point cloud with clear edges. The lidar point cloud used in the research is the point cloud of RIT campus. It is from a nadir view, and thus it includes only the rooftop of each building. Although it is not suitable for the entire algorithm, it is a good source to test the edge related part of the algorithm. Figure 4.6 shows parts of the point clouds that are used in the thesis.


Figure 4.6: Examples of lidar based point clouds

## Chapter 5

## Single Surface Based Method

As stated above, this thesis seeks the feasibility of surface reconstruction of building models based on point clouds derived from oblique imagery. In order to finish this goal, the project was divided into several smaller tasks that are easier to handle. These tasks includes plane estimation, edge extraction, and model construction. Figure 5.1 demonstrates the overall scheme developed for the thesis project. Although the method developed here is specifically for point clouds generated from oblique imagery, most of the algorithms can also be applied to other types of point cloud data such as lidar data.

Before demonstrating the tasks, a few assumptions are made to simplify the problem. First, based on observation, the buildings to be reconstructed are cubical-shape with flat surfaces. This is a fact for most of the buildings in an urban scene. Under this assumption, it is easier to estimate surfaces with simple parameters. Second, all buildings are assumed to have clear, sharp edges. This assumption can allow us to


Figure 5.1: The overall scheme of the project workflow
easily isolate edges. Combined with the previous premise, the edges we are looking for are mostly straight lines which are also easy to represent by parameters. With these assumption, we rule out buildings with complicated structures such as curved edges, or spherical surfaces. It will ease our work tremendously in terms of plane estimation and edge regulation. Fig 5.2 below shows an example of the building we are processing and its corresponding point cloud.


Figure 5.2: Example of a building that fits the assumption (Wallace Library of Rochester Institute of Technology) and its corresponding point cloud.

### 5.1 Plane Estimation

As mentioned earlier, this estimation algorithm is modified from the RANSAC algorithm. In order to increase the performance of the algorithm to this specific data, a few modification are made to the classical RANSAC algorithm. A few aspects of region growing theory are adopted here.

### 5.1.1 Classical RANSAC algorithm

When dealing with plane estimation, the RANSAC algorithm will produce a set of parameters that describes the plane and a consensus set of points that are classified to the plane. According to the basic plane representation in 3D space, we have the following equation.

$$
\begin{equation*}
A x+B y+C z+D=0 \tag{5.1}
\end{equation*}
$$

The set of parameters from RANSAC are called Theta $=[\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}]$.
Figure 5.3 below displays an example of a made-up point cloud and the result of plane estimation. The point cloud contains 6 surfaces with an average of 4000 points on each surface. Random noise is intentionally added to the data set.


Figure 5.3: Made-up point cloud example and plane estimation result of the data

The figure on the right of figure 5.3 shows the extracted planes from the point cloud. All planes that were set up are successfully identified. Table 5.1 below shows the comparison of ground truth and estimated consensus set. It demonstrates the accuracy of the RANSAC algorithm. All points are assigned to planes with small margins. The existence of outliers in the point cloud does not affect the overall accuracy of the estimation.

Table 5.1: The result of RANSAC estimation in terms of points on each plane.

|  | Ground <br> Truth | Estimation |
| :--- | :--- | :--- |
| Plane 1 | 6815 | 6830 |
| Plane 2 | 4388 | 4408 |
| Plane 3 | 4337 | 4370 |
| Plane 4 | 2444 | 2426 |
| Plane 5 | 2228 | 2208 |
| Plane 6 | 1690 | 1660 |
| Total | 21902 | 21902 |

Although RANSAC is already capable of extracting multiple planes, it still has several problems when dealing with real-life targets. Most of the time, buildings are not a simple cube with flat surfaces. They may have multiple layers in one orientation. The randomness of estimation may result in highly deviated planes in order to fit more points onto the plane. Furthermore, when dealing with larger sets of data, the random estimating process may go to exhaustion and require high processing capability. In order to reduce the effect of these problems, region growing theory is adopted to augment the basic RANSAC algorithm.

### 5.1.2 Region Growing

Region growing was originally a region-based image segmentation method. The basic idea is to examine the neighboring pixels of the selected point and determine whether it belongs to the region. It iterates until a certain criterion is met such as the region is not spreading any more. Region growing has been introduced to surface reconstruction by several researchers because of its advantages such as minimizing the memory usage when dealing with large data sets. Vieira and Shimada proposed a surface reconstruction scheme based on the theory of region growing [3]. In their method, the data is first partitioned into smaller grids, and then it tries to expand the region from an initial point which is called the seed point. It approximates a surface based on a small neighborhood near the seed point. Then further neighbors are checked whether they are compatible with the surface. If so, they are added to the region. A new surface will be approximated based on this new region. Repeat this region growing process until the region stops increasing. A final surface is then extracted. The detailed steps are explained in the pseudo code in Algorithm 2.

Figure 5.4 shows one test result from Vieira's work. Three different steps were shown in the image. With a dense point cloud, it produced very detailed reconstruction results in a time period of 24 seconds. However, the surface fitting algorithm implemented in this region growing scheme is better for spherical or higher degree surfaces. In order to connect the region growing idea to our data set, it is combined with the RANSAC algorithm.

```
Algorithm 2 Surface Extraction using Region Growing
    Definition:
    X : The data set that need to be estimated
    x : A point in the data set X
    \(b_{\text {old }}\) : Initial Estimated surface before region growing
    \(b_{\text {new }}\) : Updated surface after region growing
    b : Final surface estimation
    \(R_{b, \text { old }}\) : The region before growing
    \(R_{b, \text { new }}\) : The region after growing
    \(R_{b}\) : Final region
    Start
    Partition X into a cubical grid
    For each x , calculate and store k-nearest neighbors
    For each x , calculate surface variation based on k-nearest neighbors
    Sort x in order of increasing surface variation
    if x is labelled as used in estimation then
        Skip to next point
    end if
    Initial estimation of the surface using the first point, and store it in \(b_{\text {new }}\)
    while \(R_{b, \text { new }}>R_{b, \text { old }}\) do
        \(R_{b, \text { old }}=R_{b, \text { new }}\)
        \(b_{\text {old }}=b_{\text {new }}\)
        Region growing and update \(R_{b, \text { new }}, b_{\text {new }}\)
    end while
    if \(R_{b, \text { new }}<R_{b, \text { old }}\) then
        \(R_{b}=R_{b, \text { old }}\)
        \(\mathrm{b}=b_{\text {old }}\)
    else
        \(R_{b}=R_{b, \text { new }}\)
        \(\mathrm{b}=b_{\text {new }}\)
    end if
```



Figure 5.4: Reconstruction example of automobile C-pillar by Region Growing Algorithm[3]

### 5.1.3 Proposed Method

In the proposed algorithm, we mainly inherit the idea of using seed points from Vieira's algorithm. A small neighborhood of the seed point is used to estimate the surface. Intuitively, this algorithm works most efficiently when the seed point lies in the interior of a large group of points that are most likely in the same surface [3]. Under the assumption that surface estimation in regions that have less variation is potentially more successful, a decision is made to pick seed points based on surface variation.

The surface variation is evaluated at each point by principal component analysis (PCA). PCA has been widely used to compute local properties of point clouds such as point normals [25]. Let N be the k-nearest neighbors of a point $x$ in the data set. This technique is performed by calculating the covariance matrix of point $x$ and its
neighbors N . The covariance matrix C here can be defined as:

$$
\begin{equation*}
C=\sum_{p \in N}(p-\bar{p})(p-\bar{p})^{T} \tag{5.2}
\end{equation*}
$$

where $\bar{p}$ is the 3D centroid of $N$ neighbors in Euclidean space. This $3 \times 3$ matrix is symmetric, positive semi-definite and has three real eigenvalues, $\lambda_{0}, \lambda_{1}, \lambda_{2}$. Their corresponding eigenvectors, $v_{0}, v_{1}, v_{2}$, form an orthogonal basis of 3 dimensional space. Each eigenvalue $\lambda_{i}$ measures the variation in the direction of corresponding eigenvector $v_{i}$. Specifically, $v_{0}$ approximates the surface normal at point $x$, assuming $\lambda_{0} \leq \lambda_{1} \leq \lambda_{2}$. And the plane decided by $v_{1}$ and $v_{2}$ is recognized as the tangent plane at point $x$.[26] Thus, $\lambda_{0}$ measures the variation in the orientation of surface normal, as well as how the points variate from the tangent plane. So surface variation of point $x$ in the $k$-nearest neighbors can be defined as:

$$
\begin{equation*}
\sigma_{k}(x)=\frac{\lambda_{0}}{\lambda_{0}+\lambda_{1}+\lambda_{2}} \tag{5.3}
\end{equation*}
$$

The less the surface variation, the more likely that the points lie on the same plane. When $\sigma(x)=0$, it means point $x$ and its $k$-nearest neighbors are on the same plane [26]. After the surface variation of every point in the data set is evaluated, candidate seed points can be selected by searching for those with the least surface variation.

With the selected seed point and its nearest neighbors, the next step is to estimate a surface from these points. As mentioned earlier, RANSAC is applied here instead of Bézier surface estimation. However, a modification is made to the classical RANSAC scheme. In order to use the seed point region, the process of randomly
selecting points to perform initial estimation in the classical RANSAC algorithm is abandoned. Instead, a seed point and its neighboring region is chosen to be the library for initial estimation. Once a surface is finalized in this process, all points in the consensus set are labelled. Then another seed point is picked from the unlabelled points in the same fashion. The process will iterate until there is not enough points to estimate a plane. By doing so, it will offer the algorithm a better chance to locate the plane quickly rather than randomly selecting points to estimate planes, since the points fed to the algorithm are already the ones that are most likely to be on the same plane. Furthermore, because we use a limited number of points to estimate the surface, the maximum number of iterations can be easily calculated. For instance, if we choose to insert one seed point and its 20 nearest neighbors into RANSAC, the maximum number of iterations possible is $C_{21}^{3}=1140$. In this way, the number of iterations for each RANSAC run in multiple surface estimations can be tremendously reduced by setting a finite number for the maximum iterations, while in the case of classical RANSAC algorithm this is usually set to be infinite. A reduction in iterations means less processing time, and much more efficiency as well.

Taking the example of a cubical point cloud shown in Figure 5.3 again, both classical and adaptive RANSAC are performed on the data with the same parameter settings on a consumer laptop (Intel Core i5 $2.50 \mathrm{GHz}, 4 \mathrm{G}$ RAM). Both algorithms return decent result in terms of estimation. However, the gap in processing time between the two algorithms is large, as one can see in Table 7.6.

The detailed algorithm scheme is shown in the pseudo code below (Algorithm 3).

Table 5.2: Processing time (seconds) comparison between classical and modified RANSAC algorithm in seconds.

| Plane | 1 | 2 | 3 | 4 | 5 | 6 | Total |
| :---: | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Classical <br> RANSAC | 10.806 | 5.886 | 5.822 | 2.630 | 1.370 | 0.019 | 26.513 |
| Adaptive <br> RANSAC | 3.680 | 5.234 | 2.216 | 6.942 | 2.645 | 0.019 | 18.52 |

There is one more thing to note before the end of this section. The surface variation is not an intrinsic feature of a surface. It depends on the number of neighbors taken into account. Thus, a reasonable neighbor size is a key parameter to be considered in this algorithm. Failing to choose the right size will either lose the generalization or lose the efficiency of the algorithm. For instance, if 50 nearest neighbors are considered, then the maximum iterations possible is 20825 . It most likely will not reduce the iterations compared to the classic RANSAC algorithm.

To make the algorithm more flexible, one more parameter is designed to have a better fit. Because of the existence of noise, the result might be contaminated due to the smaller sampling size. In order to control the accuracy, another parameter is used to compensate the noise. In the estimation round using seed points and its neighbors, $\alpha$ is defined as the percentage of points from the library lying on the plane.

```
Algorithm 3 Surface Extraction using modified RANSAC
    Definition:
    X : The data set that need to be estimated
    x : A point in the data set X
    \(R_{x}\) : The region formed by x and its \(k\) nearest neighbors
    Start
    Perform \(k\) nearest neighbor(knn) search for each point
    For each x, calculate surface variation based on \(R_{x}\)
    Sort x in order of increasing surface variation
    if x is labelled as used in estimation then
        Skip to next point
    end if
    Run RANSAC using points in \(R_{x}\) as parameter estimation library
    Label the points in the consensus set of the estimated surface
```


### 5.2 Edge Extraction

Primarily, oblique imagery offers points on the sides of buildings. This advantage of oblique imagery offers a straightforward approach for edge refinement. With dominant surfaces in hand, one can easily identify the edges by calculating the intersections of every two planes. With high precision of surface estimation, this intuitive approach would have high precision as well. However, due to the limitation of available data, information on several sides cannot offer matching points to produce enough points. Thus it is hard to sharply determine the parameters of such surfaces. Such inaccuracy of plane parameters will cause the aberrate intersections. Thus such an intuitive approach may not be the best option for this thesis. In order to find an alternative approach, the cubic-shape assumption is chosen to be applied here. This means that the the missing sides are oriented straight down from the
boundary of the rooftop surface. Thus the missing side surface can be represented by a flat surface straight down from related rooftop edge. In this way, the geometry can be estimated by identifying in-plane edges in each plane.

### 5.2.1 Determine in-plane Edges

In previous efforts of solving this problem, several different approaches have been proposed by researchers. One popular approach makes the assumption that buildings are all convex hulls. This assumption turns the problem into a task of finding the minimum bounding box of the points in the plane. When buildings are exactly a cube, the convex hull algorithm works well. Unfortunately, not all buildings are simply cubical. When encountering a complicated structure such as "L" shape or "U" shape, it cannot well identify all edges. An example is shown in Figure 5.5. With the "L" shape structure shown in the exmaple, convex hull fails to recognize the edges at concave areas.

Here we decide to use another approach that looks similar to convex hull, which is Alpha Shapes. It is a generalization of convex hull [27]. Unlike the convex hull algorithm, Alpha Shapes are not confined to convex structures. It can accurately locate concave areas, and even holes in the structure, such as windows in a surface.

Edelsbrunner[27] described the concept of Alpha Shapes analogously for intuitive understanding. Thinking of the target as a huge mass of chocolate chip ice cream, where the chocolate chips stand for point set $S$, and ice cream as $R^{3}$ space. Using a spherical-shape ice cream scoop, we can carve out all the ice cream in the reach without bumping into the chips. Thereby, we can even carve out holes inside the ice


Figure 5.5: Results of Convex Hull algorithm on L-shape data. (a) the sample "L" shape data; (b) ideal edge extraction result; (b) edge extraction result from convex hull
cream. Eventually, we will end up with a shape bounded by caps, arcs and points. After straightening all round surfaces and connecting the points with line segments, an intuitive alpha shapes description of the points $S$ will be in hand. A 2D example of alpha shapes is shown in Figure 5.6. Here the parameter $\alpha$ can be considered as the radius of the spherical scoop. It is obvious to see that when $\alpha$ is too small we will be able to carve out all ice cream without touching any chips. Thus it will keep all the points in $S$ when $\alpha \rightarrow 0$. In the same sense, when $\alpha$ is too large, it will prevent the scoop from moving between two points, especially in concave areas. We will end up with the convex hull of the set $S$. Hence, the alpha shapes of S is the convex hull with $\alpha \rightarrow \infty$. Decreasing value of $\alpha$ will produce decreasing sets of
shapes and eventually developing cavities.


Figure 5.6: Example of Alpha Shapes in 2D[4], where blue dots represents point set S , green circle represents scoop with radius $\alpha$ and red line segments represent identified shapes.

In order to use the alpha shapes algorithm for boundary extraction, a preprocessing step is needed for the point cloud data. To ease the computational intensity and simplify the process, it is better to operate alpha shapes in 2D space. Thus, a projection of points onto 2D plane is performed before the actual edge extraction. Then, a standard 2D alpha shapes algorithm can be performed on the projected 2D data. In the end, we will obtain a group of boundaries including internal cavities. In order to pave the way for further algorithms, a rearrangement
process is performed. The output of the alpha shape algorithm is a list of point pairs that each pair of points are connected. In order to better process the edges in the following procedures, we reorder the points into an array in which the neighbors are connected.

### 5.2.2 Line Simplification

After the stage of the alpha shapes algorithm, we have a primary estimate of the edges. However, the output from alpha shapes usually has irregular geometry because of the fact that alpha shapes only identifies points on an edge, but not how they behave geometrically. This irregularity makes the result undesirable for final edges. Hence, here a line simplification process is necessary to produce less noisy results geometrically.

The main goal of this line simplification process is to remove redundant points and straighten the edges by lines between critical points. Over the years, there have been various algorithms developed in this area. Several researchers have compared different simplification algorithms under different circumstances and they all came to the conclusion that the Douglas-Peucker algorithm shows superior results, especially for less complicated lines [28][29][30]. Our lines here are simple straight lines with no curves or high order complications, thus the Douglas-Peucker simplification algorithm is used in this thesis to finish the task of generalizing lines.

The Douglas-Peucker algorithm, also called the Ramer-Douglas-Peucker algorithm, was independently suggested by Urs Ramer [31] in 1972 and by David Douglas and Thomas Peucker [32] in 1973. It is a recursive process that tries to find few-
er points to represent similar curves. Given a set of vertices, the algorithm first identifies the first and last points as points to be kept. With these two points as end points, it then checks whether the distance of the farthest point from the line segment is larger than a predefined threshold $\epsilon$. If not, then all points between end points that are unlabelled are discarded. If it is larger than $\epsilon$, then this point is labelled to be kept. The algorithm then recursively calls itself with labelled points. In the end, it will output a new approximation of the structure with reduced number of points. Figure 5.7 below shows a simple example of Douglas-Peucker algorithm [5].


Figure 5.7: Example of 2D Douglas-Peucker line simplification algorithm[5]

There is a set of 10 points that needs to be simplified, $P_{0}$ to $P_{9}$. In the first iteration, $P_{5}$ has the longest distance from the line segment formed by $P_{0}$ and $P_{9}$. Since the distance is larger than the threshold $\epsilon, P_{5}$ is then identified to be kept. In
later iterations, $P_{6}$ is also identified as a critical point, and the rest are discarded. The final simplified curves are formed by four critical points shown in part D of the figure.

Up till this point, the algorithm can produce simplified edges of a certain surface extracted from point clouds. However, none of the algorithms described above relates the data to the real geometrical structure of a building. Whether the edges extracted truly complies with the edges in reality is still yet to be tested.

### 5.2.3 Line Regulation

As stated above, no algorithms mentioned so far take the real geometry of the building into account. Alpha shapes only searches for points on edges in the given data, not considering whether those points lie on actual edges of the building. The Douglas-Peucker algorithm generalizes the identified edge points to a smoother, geometrically reasonable shape based only on the location of the points in the entire set of points. Again, it cannot guarantee that the outputs are the real edges of the building. In order to make sure the extracted points are on actual building edges, an algorithm is developed to correct the points in point cloud coordinates based on positions in 2D images.

The basic idea of this correction scheme is to shift the identified edge points around until they are on the edges in 2D imagery. In order to finish this task, several projective geometry techniques are applied here. For the purpose of projecting the points back to 2D imagery, the camera matrix of the image is used. A camera matrix consists of intrinsic parameters such as focal length and extrinsic parameters like
camera rotation and translation. In early work, the camera matrix for each image is extracted from the process of bundle adjustment. In Noah Snavely's documentation, he describes the process in detail [6]. The method to locate a 3D point in a related 2D image is adapted from his documentation as well.

In the bundler process, the camera matrix extracted is a $5 \times 3$ matrix in the following format such that the first row represents the focal length $f$ followed by two radial distortion coefficients $k_{1}, k_{2}$. The next 3 rows represents the $3 \times 3$ rotation matrix $R$. The last row stands for the translation vector $t$. Taken point $X$ from 3D point cloud, it first converts the points from world coordinates to camera coordinates by:

$$
\begin{equation*}
P=R * \mathbf{X}+\mathbf{t} \tag{5.4}
\end{equation*}
$$

After equalizing z coordinates to 1 for each point, we can then convert the points to pixel locations by:

$$
\begin{equation*}
P^{\prime}=f * r(p) * P \tag{5.5}
\end{equation*}
$$

where $r(p)$ is a scaling factor to compensate radial distortion:

$$
\begin{equation*}
r(p)=1.0+k_{1} *|P|^{2}+k_{2} *|p|^{4} \tag{5.6}
\end{equation*}
$$

Note that the image pixel coordinate's origin locates at the center of the image, and positive x axis points towards right, and y upwards. To adjust the coordinates to different programming environment, such as Matlab, a translation process is needed where the origin of the coordinate is top left corner.

To make things easier, instead of projecting points to a color image, a preprocessed edge map of the image is used. In this way, we do not need to deal with multi-channels when shifting the points around. An edge map is a simple gray-scale image in which higher pixel value means more likely it is on the edge. It is easier to set a threshold to identify edge points. Figure 5.8 below shows an example of an edge map for one of the images. Although we can project 3D points back onto


Figure 5.8: Edge map of 2D imagery

2D world, the inverse is unfortunately troublesome. The reason is obvious. Based on single position in a 2D image, the depth information is lost. Thus it is hard to locate the point in 3D coordinates again. To avoid this challenge, we choose to shift the point in 3D coordinates and project it back to 2D edge map to check whether it is on the edge. We then iterate until the point is on the edge. The main challenge is then to determine the proper direction to shift the point. Basically we only need to decide one of two orthogonal directions to move the point. Fortunately, thanks to the unique features of oblique imagery, several orthogonal planes can be extracted. Then, those two orthogonal directions are decided by the three dominant planes
that are orthogonal to each other. The normals of two planes will serve as the two shifting directions for points on the third plane. After close observation, every point has the tendency of shifting against the concave formed by line segments with two neighbors. The process is better explained in vector form.


Figure 5.9: Vector Description of Edge Point Shifting. (A)the vector formed by the target points and its neighbors. (B) The theory of determining direction of shifting.

Figure 5.9 shows the basic theory of edge point shifting based on 2D imagery. $V_{1}$ and $V_{2}$ represent the vectors formed by the point and its neighbors. $D_{1}$ and $D_{2}$ represent the two dominant shifting directions. After normalizing $V_{1}$ and $V_{2}$, the combination of these two vectors are calculated as $V$. Using the dot product, the angles between $V$ and $D_{1}, D_{2}$ are determined. Based on the angles, if one's absolute value is smaller than a predefined $\epsilon$, the the point should move in the direction that produces that particular result. In the other case, if both values are larger than $\epsilon$, then the point should move in both directions. In the example above, $\theta_{1}$ is obviously
smaller than $\theta_{2}$, and smaller than $\epsilon$, then the point should move in the direction of $D_{1}$.

When the shifting direction is determined, then the point is shifted in the direction at a reasonable interval, and then is projected back to 2D edge map to check whether or not it is on edge. If not, then it needs to shift more until it satisfies the condition of reaching a threshold of pixel value. This process iterates for every point that is produced by the algorithm in the previous stage. In this way, the points are guaranteed to be on the edge of the building and therefore have decent precision.

In summary of this section, the edge identification process consists of three steps. First, alpha shapes is applied to identify edge points from the points. Then, the Douglas-Peucker line simplification algorithm is used to simplify the lines and makes the lines geometrically reasonable. At last, a shifting scheme is performed to correct the error between identified edges and real building edges.

### 5.3 Model Extraction

With edges and plane parameters in hand, now we are ready to construct building models. In order to produce an output that can be used in various situations, a universally accepted format is used here to represent the models. Here we propose to use wavefront OBJ file format.

OBJ format is a simple data format that represents 3 D geometry alone. It includes the positions of vertices, normals and the faces that makes each polygon defined as a list of vertices. Since OBJ format doesn't require a unit for the data, it
can also contain scale information. An example of OBJ format file is shown in the Figure 5.10.


Figure 5.10: An example of the OBJ file format.

In this format, it starts with listing all vertices that are needed for this model. It follows the order of $(x, y, z)$ coordinate order without any units. The same case is for normals, if this information is available. Face definition in an OBJ file has several different representations. The most simple one is shown in line 1 and 2 of the face definition in the example. It simple identifies all the vertices that lies in the same face, and lists them in a counter-clockwise order. The other one shown in the example attaches vertex normals to each vertex in the surface. There are many more elements in the format such as texture coordinates, parameter space vertices, etc. These are beyond the scope of this thesis.

In our case, in order to generate a standard OBJ file, we simply need to list all
edge points as vertices, and then list all surfaces identified in earlier stages. Since the data comes from oblique imagery, it should cover all sides, so a simple stitching process is sufficient to put the model together.

## Chapter 6

## Minimum Bounding Box Based

## Method

In a more realistic case, more often than not, the point cloud is not complete and does not cover all sides of the building. Just like the case in the Pictometry data, ten images were facing north, and only four were facing the other three directions. This fact results in missing walls on three sides while maintaining two complete surfaces, the rooftop and the surface facing south. The problem is shown in Figure 6.1

With this defect, the approach described in Chapter 5 may not work well. That approach assumes that the points cover all sides of the building. Without this assumption, just like in our case, the approach is only able to stitch two surfaces together while the rest of the walls are left empty. Figure 6.2 shows one result of such work.

In order to compensate the missing information on some of the walls, another


Figure 6.1: The problem with Pictometry data set, While points cover roof and front wall, walls on the other sides are missing.


Figure 6.2: A preliminary model of library.
approach based on a minimum bounding box is proposed here.

### 6.1 Minimum Bounding Box

The technique of minimum bounding box is mostly used in geometry. The goal of this technique is to find the bounding box that encloses a set of points and also has the smallest measure. The measure here can be area, volume, or perimeter of the box. In most cases, another constraint that is taken into consideration is the orientation. Particularly in model reconstruction, the box must have the right orientation for further processing. While finding a minimum bounding box is not a complicated task, making orientation alignment is the most difficult step in this algorithm. Figure 6.3 gives an example of minimum bounding box on 3D data points.


Figure 6.3: An example of minimum bounding box, where blue points are the input points in 3D and red lines are the minimum bounding box

The algorithm used here constructs a convex hull of the point set and utilizes properties of the convex hull such as face edges, face normals, and face orientations to align bounding box orientations to the data set. Figure 6.4 shows minimum bounding box results on real data. Figure 6.4(a) shows a result of the algorithm. By using minimum bounding box, the missing walls are compensated with a flat surface of the box.

(a)

(b)

Figure 6.4: Minimum Bounding Box algorithm results on (a) Pictometry data and (b) DIRSIG data

### 6.2 Proposed Approach

Utilizing the idea of minimum bounding box, here a new approach is proposed to make up the missing walls in the practical point cloud data. Again, the assumption for this approach stays the same as the previous approach. The building consists of flat surfaces only. This assumption allows us to use a bounding box to simulate the missing walls by casting a flat surface to the missing walls.

### 6.2.1 Searching for Dominant Planes

The approach starts with the adaptive RANSAC algorithm described in Chapter 5 to estimate multiple planes from the point cloud. With the estimated surfaces, the next step is to find the dominant surfaces. The dominant surface is defined as the surface that covers one side of the building, and that has the most inlier points. To find the dominant planes, the estimated planes are first sorted in a descending order based on the number of inliers. By default, the plane with the most inliers is labelled as one dominant plane automatically. In the sorted order, each of the remaining planes is checked if it is orthogonal to two unparallel planes, and can only be parallel to one of the labelled dominant planes. After this process is done, all the dominant planes are labelled and ready to use. Figure 6.5 shows some results of dominant plane identification.

Generally, one building has five dominant surfaces depicting four sides and one rooftop. However, due to the missing information, one building in our data does not necessarily contain five dominant surfaces. In Figure 6.5, the point cloud in (a) obviously only contains two dominant surfaces while the one in (b) identifies five surfaces as dominant.

The labelled dominant surfaces are then used to construct the basic main structure of the building. Using the inliers from the dominant planes only, a cubic structure can be acquired by performing the minimum bounding box algorithm on the points. This structure acts as the base structure of the building, and later processes are built on this structure.


Figure 6.5: Dominant plane search on different data sets (a) RIT library data set, (b) identified 2 dominant planes in red and green, (c) DIRSIG data set, (d) identified 5 dominant planes in different colors

### 6.2.2 Histogram based Clustering

After the main structure is constructed, the next step is to assemble the remaining surfaces to the box. This part of the approach is the most difficult part due to the complicated structure of the remaining planes. Some of the estimated planes contain only one surface while some contain more than one surface which shares the same plane. Figure 6.6 shows an example of the complication. In (a), the estimated plane has 12 surfaces while in (b) there is only one surfac lying on the estimated plane.

Minimum bounding box only finds the bounding box that can enclose all the input points. It cannot preserve these details of small surfaces in one estimated plane. In order to fully reconstruct the details, we need to separate the surfaces lying on the same plane into different clusters.


Figure 6.6: Two different situations of planes, in (a), 12 surface share the same plane, in (b) only one surface on the estimated plane

The method we proposed here to cluster the points is histogram based K-means clustering. K-means is a clustering method that has been widely used for decades. It was first proposed by McQueen [33] in 1967 as a local search algorithm that partitions $n$ points into $k$ clusters. It works in the following way. The points are first seeded with k initial cluster centers. Then it assigns every remaining data point to its closest center, and then recalculates the new centers as the means of their assigned points. This process of assigning data points and adjusting centers is repeated until the means are stabilized.

The number of clusters, $k$, affects the result of clustering. An inappropriate
choice of $k$ may yield a terrible result. However, in the k-means algorithm implementation in many data analysis software packages, the number of clusters is set as an input parameter. This means one has to know how many clusters exist in the data before processing it. In the interests of automation, the number of clusters is expected to be determined automatically without any human involvement. In order to achieve that goal, a preprocessing step is essential before the clustering.

The first step in this process is to perform a principal components analysis on the data points. In this way, the points can be projected into three dimensions based on the variation in the distribution of the points. Then a processing of searching for maxima in the histogram of each dimension is conducted. The number of clusters is decided by the number of maxima in each dimension. The search for maxima is explained as follows.

After projecting the points to each dimensions, a statistical histogram of the distribution is computed for each dimension. Figure 6.7 demonstrates the points of Figure 6.6(a) projected into the PCA dimensions and their corresponding histograms. Visually, a local peak in a histogram means that the related region has the most points in the local neighborhood. The points in this peak region can then be identified as belonging to one cluster. So the problem now breaks down to search for the number of maxima in the histogram.

In order to correctly find the number of peaks, several rules have been utilized to eliminate false peaks. Here we use an example histogram shown in Figure 6.8(a) to better illustrate the process. Let $H$ be the histogram of the points in one dimension, and $p$ be one bin of the histogram. The first rule is to find all the local maxima


Figure 6.7: Principal Component Analysis of one point set. (a)(b)(c) are the points projected into the orthogonal dimensions,(d)(e)(f) are the corresponding histograms of the points in each dimension.
in the histogram. $N_{p} 1=\{p \mid H(p)>H(p-1), H(p)>H(p+1)\}$, is the set that includes all local maxima bins that have more points than the one before and after them. The result is shown in Figure $6.8(\mathrm{~b})$. In the set of $N_{p} 1$, there might be some bins that are peaks among low bins. This means some of them might contain few points that cannot be identified as clusters. Thus, another rule is set to reduce the maxima by eliminating extremely low peaks. The set of this rule is named as $N_{p} 2=$ $\left\{N_{p} 1 \mid H N_{p} 1(p) \geq \alpha * \max (H)\right\}$ as shown in Figure 6.8(c), where $\alpha$ is a factor that is decided by the size of the input points. Similarly, among the high peaks, there is a probable case that several high peaks are close to each other. And these high peaks actually describe only one single cluster. In this case, the peak we are looking
for should be distinctive peak in relation to its neighbors and also its neighboring peaks. So one more rule, $N_{p} 3=\left\{N_{p} 2 \mid \operatorname{sum}\left\{H\left(N_{p} 2(p)\right), H\left(N_{p} 2(p+1)\right\} / 2>\beta *\right.\right.$ $\operatorname{sum}\left\{H\left(N_{p} 2(p): H\left(N_{p} 2(p+1)\right)\right)\right\} /\left(N_{p} 2(p+1)-N_{p} 2(p)+1\right)$, is set to eliminate the shallow peaks to avoid this situation, where $\beta$ is a factor based on the size of the input points. The result is shown in Figure 6.8(d). Thus, after the process, the number of peaks in this dimension is set to be 1 .


Figure 6.8: Selecting cluster number by histogram (a) histogram of the input data in one dimension, (b) histogram after selecting local maxima, (c) histogram after eliminating low maxima, (d) histogram after eliminating shallow maxima

After all three dimensions are analyzed by the histogram peak detection algo-
rithm, the number of clusters $k$ is then determined by multiplying the numbers of peaks of three dimensions together. The reason that we are able to do so is that PCA projects the points into three orthogonal dimensions that are not correlated. One more thing to note here is that we can perform noise reduction while searching for local maxima. During the search in each dimension, the bins with extremely low points and isolated from the groups of points are considered as misidentified points and thus removed from the input data. The example is shown in Figure 6.9.


Figure 6.9: Noise reduction based on histogram.

After the histogram process, the number of clusters is determined. Thus, a kmeans clustering is performed on the noise reduced data. One result of the clustering is shown in Figure 6.10. The clustered data is then treated as multiple estimated surfaces sharing the same plane primitives.


Figure 6.10: Clustering result of one plane. Each color represents one cluster

One thing to point out here is that, since we are trying to achieve automatic reconstruction, there should be as little human involvement as possible. One possible human involvement is the user-determined parameter settings that is based on the size and quality of the point clouds. The single-plane based algorithm, for example, requires a few parameters such as the alpha parameter in alpha shapes, distance parameter in line simplification. The bounding box based algorithm, on the other hand, requires no parameter input from the user. The most obvious parameter, the number of clusters in the k-means clustering process, is done automatically. This fact allows the bounding box based algorithm to achieve a better automation.

### 6.2.3 Model Construction

The next step is to attach the remaining surfaces to the main structure. Again, the minimum bounding box approach is used here to form a model for the surface. The minimum bounding box produces eight corner points that describe the box.

These eight points are then used to find the main structure plane that this surface should be attached. The sum of the distances of the corner points to each of the dominant surfaces is computed. The dominant surface with the least distance is the main structure plane for which we are looking. Then the four corner points that are closer to the identified main structure plane are projected onto the plane. The remaining four points are projected to the targeting surface. To this step, the targeting surface is attached to the main structure through a bounding box. And when all the remaining surfaces are attached, a building model is accomplished. The OBJ file is then generated using the corner points of all bounding boxes. Thanks to the simplicity of bounding boxes, the OBJ file is created by simply listing all the surfaces using four corner points.

## Chapter 7

## Results and Discussion

As mentioned earlier, both approaches introduced in previous chapters are used for the purpose of reconstructing single building models. The following sections are in the order of the process explained in the previous chapters so that one can easily see the effect of each algorithm on the data.

### 7.1 Results on Adaptive RANSAC Algorithm

The adaptive RANSAC algorithm is used in both approaches. It is important to discuss its efficiency before presenting the results in both approaches.

The performance of the adaptive RANSAC algorithm along with the strategy to solve problems occurring in multiple primitive estimation was tested with real data and compared to the original RANSAC algorithm. The comparison is mainly conducted in two aspects, computational efficiency and fitting accuracy [34].

### 7.1.1 Computational Efficiency

Hypothesis testing in RANSAC is an iterative process. Generally a counting of iterations would be adequate to characterize the efficiency. However, due to the fact that our modified algorithm includes a nearest neighbor search which is not processed during each iteration, an elapsed processing time is used here to evaluate the efficiency. As shown in Table 7.1, almost half the time in our modified algorithm was consumed in finding the first plane. At that time, the nearest neighbor search was performed and surface variation was calculated. Even so, it was obvious the total time was much shorter than the original RANSAC, especially in the first two runs. In the consideration of iterations, most of the modified algorithm runs were finished within 50 iterations as expected. As stated in Chapter 2, the decrease in the estimation pool reduces the number of iterations and thus reduces the processing time. Table 7.1 and Table 7.2 show a significant drop in total elapsed time compared to original RANSAC.

Table 7.1: Processing time (seconds) comparison between original and adaptive RANSAC algorithm for DIRSIG building.

| Plane | 1 | 2 | 3 | 4 | 5 | Total |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Classical RANSAC | 10.806 | 5.822 | 2.630 | 1.370 | 0.019 | 20.627 |
| Adaptive RANSAC | 6.942 | 2.216 | 3.680 | 2.645 | 0.019 | 13.286 |

Table 7.2: Processing time (seconds) comparison between original and adaptive RANSAC algorithm for Airborne Oblique Imagery Data

| Plane | 1 | 2 | 3 | 4 | 5 | Total |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Classical RANSAC | 61.184 | 44.786 | 29.582 | 12.190 | 7.743 | 155.424 |
| Adaptive RANSAC | 44.534 | 33.560 | 32.231 | 13.407 | 12.038 | 135.77 |

### 7.1.2 Fitting Accuracy

A direct visualization of the fitting accuracy is shown in Figure 7.1 where the blue dots denote the original point cloud. The point cloud depicts a warehouse door. The grey line represents the estimated primitive. Figure 7.1(a) is the fitting result from original RANSAC. The primitive fits better in the center while deviated on the edge. Figure 7.1(b) shows better fitting results from our modified algorithm. The points are evenly distributed in all areas of the plane. From the top view, the plane fit by our algorithm looks much thinner than the result from the original algorithm.

To better illustrate the fitting accuracy, a point to plane distance is calculated at each inlier point. An average distance error is achieved for each estimated plane. The results are shown in the Table 7.3 and Table 7.4 for the two data sets. The result again indicates improvements of accuracy from modified algorithm in some cases.

Besides computational efficiency, our adaptive RANSAC algorithm shows better results of estimating detailed minor surfaces than the traditional algorithm. The results in Figure 7.2 show that our modified algorithm has superior performance than the traditional RANSAC algorithm. First, the traditional algorithm estimated

Table 7.3: Fitting Accuracy Comparison betweeen RANSAC and Adaptive RANSAC for DIRSIG Building

| Plane | 1 | 2 | 3 | 4 | 5 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Classical RANSAC | 0.0037 | 0.0088 | 0.0122 | 0.0113 | 0.0168 |
| Adaptive RANSAC | 0.0014 | $5.50 e^{-} 4$ | 0.0011 | 0.0023 | $5.86 e^{-} 4$ |

Table 7.4: Fitting Accuracy Comparison (Meters) betweeen RANSAC and Adaptive RANSAC for Airborne Oblique Imagery Data

| Plane | 1 | 2 | 3 | 4 | 5 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Classical RANSAC | 0.44 | 1.57 | 0.34 | 1.32 | 0.162 |
| Adaptive RANSAC | 0.16 | 0.33 | 1.64 | 0.25 | 0.89 |



Figure 7.1: Plane fitting result of a door as circled in red (top view). (a) result from original RANSAC, (b) result from adaptive RANSAC
at most three surfaces while our algorithm successfully identified five surfaces. This makes sense when thinking about the logic behind these two algorithms. RANSAC algorithm feeds the estimator random points for surface estimation. After two or
three dominant planes are extracted, the RANSAC algorithm is hard to locate other small planes that contains detailed layers of the building which have fewer points on the plane. However, with our algorithm, after dominant planes are found, it still feeds the algorithm with points that are most likely to be on the same plane for estimation. Thus, small detailed surfaces have a better chance to be identified.


Figure 7.2: Surface Extraction results for both traditional and modified RANSAC algorithm for library building. (a) the airborne image of library building; (b) the point cloud of the building; (c) classic RANSAC algorithm result; (d) modified RANSAC algorithm result

Another advantage shown in the result are the details in the extracted surfaces. Figure 7.3 shows the points recognized as located on the rooftop of the library
building in Figure 7.2(a). Compared to the actual rooftop, one can easily observe three different layers. The major layer is a U-shape surface with a flat tower in the center of the cave. While the traditional algorithm fails to carve out the details of the plane, the modified algorithm mostly shows the basic outline of the U-shape and the tower.


Figure 7.3: Consensus Set Comparison of (a) traditional and (b) modified RANSAC algorithm on the point clouds of rooftop

### 7.2 Results on the Edge related Approach

With the estimated planes from adaptive RANSAC, the approach introduced in Chapter 5 goes through every plane to extract and simplify edges, stitch them together to build a model from the edges and planes. The rest of this section will present the results of this approach on different data sets.

### 7.2.1 Edge Identification

As stated in previous chapter, this edge identification process consists of three steps with each step restraining the points towards the real edge of the building. To demonstrate the result, the rooftop of the library building is used as an example. The result is shown in Figure 7.4 below. In (a), red points are the identified edge points by alpha shapes. Again, it proves that alpha shapes has effective results on edge points detection; however, geometrically it does not comply with reality. (b) shows the results from Douglas-Peucker algorithm. It is obvious that lines are longer and critical points representing edges are tremendously reduced; but the actual geometry of the building is still not seen in this step. (c) presents the effects of 2D imagery correction. Zigzags that appeared in the last step are gone, lines are more aligned. Another Douglas-Peucker algorithm is performed to eliminate unnecessary points. It produces results in (d). Lines are straightened and they mostly comply with the geometry in this result.

The edge identification algorithm was also tested on DIRSIG data and lidar data. The fact that both data sets have much denser point cloud makes the algorithm more effective. Denser point clouds tends to have neat and less noisy edges. The results on both data sets shows the same. DIRSIG data includes almost no noise, the edges are clear cut and visually straight before processing. The algorithm simply reduces the number of points needed to describe the plane. Lidar data has redundant points on edges such that the edges are more obvious and identifiable. Thus, the edges extracted are straight and comply with geometry. Although the assumption of this approach claims that the structure of the building is cubical, which means


Figure 7.4: Edge detection results for one single surface. (a)edge points(red) detected by alpha-shape algorithm; (b) edges from Douglas-Peuker algorithm; (c) Modified edges after edge correction from 2D image; (d) modified edges after line simplification on corrected edge points
the shape of each surface is rectangular, the result on lidar data demonstrate that the algorithm can have a decent estimation of curves when the point cloud is dense enough.


Figure 7.5: Edge identification results on DIRSIG and lidar data sets.

### 7.2.2 Model Construction

When the edges are extracted, along with the planes, a model can be constructed by simply stitching them together. This construction scheme is based on the assumption that the point cloud covers all sides of the building. DIRSIG data fulfills the assumption perfectly. Thus the result here focuses on the DIRSIG data.

Figure 7.6(a) shows a point cloud data set from DIRSIG generated images. It covers all sides of the building while some minor regions are missing points. The approach has no problem connecting main structure surfaces together. However, the doors, and stools on the rooftop of one point cloud data failed to be reconstructed because there is no surface to connect them with the main structure. Instead of looking for additional planes to connect them, the method we used here is to simply project the edge points on these surfaces to the closest surface. In this way, the surfaces are forced to connect. The result is shown in Figure 7.6(b).

As discussed in Chapter 5, when the point cloud is missing major part of a


Figure 7.6: Model construction result on one DIRSIG derived point cloud data set 1.
building, this approach fails to construct a watertight model because there are not enough surface edges to stitch together.

### 7.3 Results on Minimum Bounding Box related Approach

Using another DIRSIG data set, the minimum bounding box approach is performed.
Figure 7.7 gives the result of the approach. As explained in Chaper 6, minimum bounding box compensates the missing sides, particularly in the main structure. It allows us to build a watertight model even when side information is missing. The door that is not connected to the main structure in the previous approach now is attached to the wall.

This approach is tested on another set of data, which is generated from oblique imagery. It is much noisier and denser than the RIT campus data set. The result is


Figure 7.7: Model construction result on DIRSIG derived point cloud data set 2.
shown in Figure 7.8. In the point cloud data, one side of the wall of the building is missing due to the trees near the building. The algorithm has no problem recovering the missing part of the wall. However, the error is obvious at the center of the frontal wall where there is a spherical surface. The bounding box enclosing this spherical wall is randomly placed and causes a major error.


Figure 7.8: Model construction result on oblique imagery derived point cloud data set.

This brings us to the disadvantages of this approach. The nature of bounding box confines its capability to reconstruct more complex models. Although projecting onto the estimated planes enables the approach to construct planes that not parallel to building orientation, the approach is still strictly confined in flat planes. Another example of this confinement is in the warehouse model in Figure 7.6. The stools on the rooftop are cylindrical. The approach can only replace them with cubical. The detail is lost.

Another drawback of this approach comes with details on the planes. The building in Figure 7.9 is an example. The walls on all 4 sides extend taller than rooftop. However, the bounding box cannot identify the design of the building. This detail is also lost in the reconstructed model. Because of the simplicity of the model from this approach, a large number of small details are not preserved.


Figure 7.9: An example of missing information on the reconstructed model box

### 7.4 Model Accuracy Validation

The point cloud is generated through a feature matching process. It is similar to a sampling process, so it is not capable of offering continuous measurement of the building. The reconstruction scheme we proposed is also aimed at reducing the number of vertices in the final building model. The result of the reconstruction should be evaluated against true measurement of the scene. It is almost impossible to have a qualitative evaluation of reconstruction results due to the fact that ground truth data is difficult to obtain, especially in urban and residential areas. However, accuracy evaluation is still necessary. Instead of obtaining ground truth in 3D, we evaluate the accuracy of reconstruction in 2D imageries where the reconstruction is built.

For this validation process, we use DIRSIG data set to test the result. As explained in earlier chapter, DIRSIG is created with known geometry and parameters, thus making it noise free in the point clouds. Any existing error would come from the process of reconstruction. Therefore, this data set is an ideal benchmark for reconstruction quality evaluation.

As stated before, the best way to evaluate the quality of a reconstruction is to compare the model with known ground truth. However, in most cases, ground truth is very difficult to obtain. Since our model is constructed mostly with vertices of corner points, it is easier for us to compare the corner points of the constructed model with real buildings. Theoretically, the comparison can happen in 2D and 3D space. However, 3D coordinates of real corners are harder to generate, so we propose to evaluate the quality of the constructed model in 2D space on the image plane.

The building model we constructed contains the corner points and surfaces they form. If the corner points are correctly located, then the surface should be correctly reconstructed as well. Hence, by evaluating the accuracy of the location of the corner points, we will be able to evaluate the quality of the reconstructed model. As explained in Chapter 5, 3D points can be reprojected back to the image plane using known camera matrix. The DIRSIG data set comes with accuracy camera information and thus we are able to project the 3D corner points in world coordinate system back to 2D image coordinate system using equation (5.4) and (5.5). Figure 7.10 below shows an example of projected 3D corner points in a DIRSIG image.


Figure 7.10: An example of DIRSIG image with projected corner points

The original corner points can easy picked by hand in the image, as shown in the figure. Then a RMS error is calculated for the visible corners in the images against projected corner points using the formula below.

$$
\begin{equation*}
R M S=\sqrt{\frac{1}{N} \sum\left(\left(x-x_{c}\right)^{2}+\left(y-y_{c}\right)^{2}\right)} \tag{7.1}
\end{equation*}
$$

The nature of 2D images omits 3D information, thus one image is not enough to cover all the corner points in all sides of the building. Here we picked four images covering all four sides of the building, and repeated the process explained above to obtain RMS errors for all the corner points. The four images selected are shown in figure 7.11. The detailed corner points among ground truth and projections are shown in Table 7.5.

Table 7.5: Ground Truth Corner Points vs Projected Corner Points

| Ground <br> Truth X | Ground <br> Truth Y | Projected <br> X | Projected <br> Y |
| :--- | :--- | :--- | :--- |
| 556 | 432 | 551 | 428 |
| 583 | 432 | 589 | 428 |
| 583 | 429 | 589 | 425 |
| 628 | 429 | 632 | 425 |
| 628 | 432 | 631 | 428 |
| 656 | 433 | 659 | 428 |
| 557 | 528 | 560 | 530 |
| 655 | 528 | 659 | 530 |
| 628 | 529 | 625 | 530 |
| 582 | 529 | 580 | 530 |



Figure 7.11: Samples of DIRSIG generated images

Here we have used this method to validate the reconstruction results of two building models in DIRSIG scene. The results are shown in the table below. As one can see, the error measured in pixel units is reasonably low considering the size of the building. Converting the pixel units into meters using similar triangles, the error in the reconstruction in corner points is approximately within one meter. One thing to note here is that in these error calculations, the rooftop is included in all sides. This means the real error should be lower than the calculated value.

Table 7.6: RMS Error of four sides of the building

| Sides | RMS(Pixels) |
| :--- | :--- |
| 1 | 4.12 |
| 2 | 3.52 |
| 3 | 5.31 |
| 4 | 3.83 |
| Total | 4.29 |

## Chapter 8

## Conclusions and Future Work

The ultimate goal of this thesis is to reconstruct 3D building models from point clouds derived from aerial oblique imagery. The nature of oblique imagery gives us information on all the sides and enables us to construct a complete and watertight model of the building. However, the limited availability of images and complications of registration constrains the quality of the point clouds. Most of the work in this project is to compensate the drawbacks inherent in oblique image data sets and construct building models as close to reality as possible. Two approaches have been proposed to finish the task.

The first approach is a single surface based approach. It first estimates surfaces from point cloud data and processes one surface after another. We proposed a new modification to the traditional RANSAC algorithm so that it works more efficiently in the scope of this project. Instead of randomly feeding points to estimate planes, a seed point and its neighbors which are most likely to be on the same plane are
chosen to estimate planes. The algorithm is tested on multiple data sets from oblique imagery derived point clouds to lidar point clouds. The results show that our modified algorithm is computationally efficient and accurate. A chain of algorithms such as Alpha Shapes and Douglas Peucker algorithms are utilized to identify and simplify the edges and use the edge points as potential polygons. These algorithms are also tested on multiple data sets from different sources. The results prove that when the point cloud is dense enough, the method works efficient and align with geometry. These polygons and estimated surfaces are then used to stitch together to form a model. The results show that the approach works well on a dense and complete point cloud. When the point cloud is not complete, and not fully covering all sides, this method fails at attempting to generate a watertight model.

The second approach is based on minimum bounding box, and looking to compensate the defects due to the incomplete point clouds. In this approach, adaptive RANSAC is also applied to estimate planes. With the identified dominant planes, the main structure of the building is achieved. A histogram based clustering scheme is proposed to separate surfaces that land on the same plane. Then minimum bounding box is used to assemble small detail components to the main structure. This approach well compensated for the missing information of the point clouds by replacing it with a surface from the bounding box. When the method is tested on multiple point clouds, it shows decent computational efficiency and watertight models. The algorithm doesn't require any parameter input from the user, which reduces the human involvement and allows the algorithm to achieve better automation. However, a close inspection reveals that the method potentially loses details in the surfaces.

The results of the approaches on various testing data sets demonstrate that both of the methods are capable of reconstructing building models from point clouds. However, there are also limitations shown in the results. Both methods assume that the buildings only consist of flat surfaces without any higher degree primitives such as curves and spheres. This assumption confines the robustness of the approaches as the design of architectures has more smooth spheres. In order to extend the robustness of the approaches, future work may explore the possibility of applying minimum bounding spheres or cylinders to the approach. While the first approach does not rely on rectangular shapes as much as the minimum bounding box approach, it is highly dependent on the density of the point clouds. The approach gets inefficient when the point cloud is sparse. Further work is needed to increase the robustness while not applying more assumptions.

The current trend in 3D modelling is shifting to the application of real time reconstruction, and in smart phone reconstruction. These applications require a time and memory consumption within a reasonable limit. Although the approaches proposed here have not shown any high memory consumption, it is highly dependent on the size of the point cloud data. When dealing with real time processing, the RANSAC algorithm may increasingly become cumbersome, and the nearest neighbor search requires a high computation cost. So a less costly surface estimation method is suggested when applying the approaches to real time processing.

## Bibliography

[1] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
[2] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381-395, 1981.
[3] Miguel Vieira and Kenji Shimada. Surface extraction from point-sampled data through region growing. International Journal of $C A D / C A M, 5(1), 2009$.
[4] R.K. Martin. Using alpha shapes to approximate signal strength based positioning performance. Signal Processing Letters, IEEE, 18(12):741-744, 2011.
[5] Ziluan Liu, Yuehui Jin, Yidong Cui, and Qiyao Wang. Design and implementation of a line simplification algorithm for network measurement system. In Broadband Network and Multimedia Technology (IC-BNMT), 20114 th IEEE International Conference on, pages 412-416, 2011.
[6] N. Snavely. Bundler: Structure from motion (sfm) for unordered image collections. http://www.cs.cornell.edu/ snavely/bundler/, 2009.
[7] David Nilosek, Shaohui Sun, and Carl Salvaggio. Geo-accurate model extraction from three-dimensional image-derived point clouds, 2012.
[8] Jing Xiao, Markus Gerke, and George Vosselman. Building extraction from oblique airborne imagery based on robust faade detection. $\{I S P R S\}$ Journal of Photogrammetry and Remote Sensing, 68(0):56-68, 2012.
[9] Schultz S. Giuffrida F. Wang, Y. Pictometry's Proprietary Airborne Digital Imaging System and its Application in 3D City Modelling. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXVII,PartB1, 2008.
[10] Joachim Hhle. Photogrammetric measurements in oblique aerial images. Photogrammetrie, Fernerkundung, Geoinformation, (1):7-14, 2008.
[11] M. Gerke. Dense matching in high resolution oblique airborne images. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science 38(Part 3/W4), pages 77-82, 2009.
[12] Jie Zhang. Dense Point Cloud Extraction From Oblique Imagery. Master's thesis, Rochester Institute of Technology, 2013.
[13] Charles V. Stewart. Robust parameter estimation in computer vision. SIAM Rev., 41(3):513-537, September 1999.
[14] J. Illingworth and J. Kittler. A survey of the hough transform. Computer Vision, Graphics, and Image Processing, 44(1):87-116, 1988.
[15] E. Turner and A. Zakhor. Sharp geometry reconstruction of building facades using range data. In Image Processing (ICIP), 2012 19th IEEE International Conference on, pages 1785-1788, 2012.
[16] V. Verma, Rakesh Kumar, and S. Hsu. 3d building detection and modeling from aerial lidar data. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 2, pages 2213-2220, 2006.
[17] Rey-Jer You and Bo-Cheng Lin. A quality prediction method for building model reconstruction using lidar data and topographic maps. Geoscience and Remote Sensing, IEEE Transactions on, 49(9):3471-3480, 2011.
[18] Shugen Wang, Qiuyuan Gou, and Mingwei Sun. Simple building reconstruction from lidar data and aerial imagery. In Remote Sensing, Environment and Transportation Engineering (RSETE), 2012 2nd International Conference on, pages 1-5, 2012.
[19] Carlo Tomasi. Shape and motion from image streams under orthography: a factorization method. International Journal of Computer Vision, 9:137-154, 1992.
[20] D. Frere, J. Vandekerckhove, T. Moons, and L. Van Gool. Automatic modelling and 3d reconstruction of urban buildings from aerial imagery. In Geoscience and Remote Sensing Symposium Proceedings, 1998. IGARSS '98. 1998 IEEE International, volume 5, pages 2593-2596 vol.5, 1998.
[21] M. Maurer, M. Rumpler, A. Wendel, C. Hoppe, A. Irschara, and H. Bischof. Geo-referenced 3d reconstruction: Fusing public geographic data and aerial imagery. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 3557-3558, 2012.
[22] F. Calakli, A.O. Ulusoy, M.I. Restrepo, G. Taubin, and J.L. Mundy. High resolution surface reconstruction from multi-view aerial imagery. In 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on, pages 25-32, 2012.
[23] Katie N. Salvaggio and Carl Salvaggio. Automated identification of voids in three-dimensional point clouds, 2013.
[24] Rochester Institute of Technology. Dirsig - the digital imaging and remote sensing image generation model. http://www.dirsig.org.
[25] Edward Castillo and Hongkai Zhao. Point cloud segmentation via constrained nonlinear least squares surface normal estimates. Recent UCLA Computational and Applied Mathematics Reports, 2009.
[26] M. Pauly, Markus Gross, and L.P. Kobbelt. Efficient simplification of pointsampled surfaces. In Visualization, 2002. VIS 2002. IEEE, pages 163-170, 2002.
[27] Herbert Edelsbrunner and Ernst P Mücke. Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG), 13(1):43-72, 1994.
[28] Robert B McMaster. A statistical analysis of mathematical measures for linear simplification. The American Cartographer, 13(2):103-116, 1986.
[29] Ellen R White. Assessment of line-generalization algorithms using characteristic points. The American Cartographer, 12(1):17-28, 1985.
[30] Jill S Marino. Identification of characteristic points along naturally occurring lines: An empirical study, 1979.
[31] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing, 1(3):244-256, 1972.
[32] David H Douglas and Thomas K Peucker. 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.
[33] J. MacQueen. Some methods for classification and analysis of multivariate observations, 1967.
[34] Zhizhong Kang, Liqiang Zhang, Baoqian Wang, Zhen Li, and Fengman Jia. An optimized baysac algorithm for efficient fitting of primitives in point clouds. Geoscience and Remote Sensing Letters, IEEE, 11(6):1096-1100, June 2014.

