Master of Science Geomatics

Straightening and simplifying a multi-view stereo mesh of a city

Yuxuan Kang

June 2017

TU Delft

STRAIGHTENING AND SIMPLIFICATION OF A MULTI-VIEW STEREO MESH OF A CITY

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Yuxuan Kang

June 2017

Yuxuan Kang: *Straightening and simplification of a multi-view stereo mesh of a city* (2017)

ISBN 999–99–9999–999–9

The work in this thesis was made in the:

3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors: Dr. Hugo Ledoux
Dr. Abdoulaye Diakité
Co-reader: Dr.ir. Martijn Meijers

ABSTRACT

With fast development of both hardware and software, 3D real-world scene data, for example multi view stereo mesh, can be acquired efficiently. And since 3D city data are more realistic and more useful in some applications, they begin to appear in the market.

Although the generation of 3D city data like multi view stereo mesh can be fast, the data might contain many measurement errors. Because of these measurement errors, some regular objects such as building, ground etc sometimes are not exactly flat. Points which should be on the same plane have small deviations from the plane they belong to, which makes the flat surface bumpy. One way to solve this problem is to control the quality of data acquisition and data processing. Unfortunately, even the data error sources are known, it is not possible to eliminate all the errors, and accurate but expensive data acquisition equipment sometimes are not affordable. In that case, processing existing data is much more economic and time saving compared with collecting data again.

This MSc thesis aims at solving above problem. It provides a methodology on straightening planar parts of multi view stereo mesh of the city. After straightening the mesh, this thesis also tries to simply the mesh by removing redundant vertices and faces in the mesh, so that the data will be clean and the data storage can be reduced.

ACKNOWLEDGEMENTS

With this thesis done, it is the end of my master study. In these two years, many important people appeared in my life and gave me hands both in study and in life.

First I would like to express my sincere gratitude to my mentors Dr. Hugo Ledoux and Dr. Abdoulaye Diakité. They provided me a lot of help during these months, especially when I was confused and lost, they always guided me back on track in time. And I would like to thank my co-reader Dr. Martijn Meijers for taking time to give advices and feedbacks on my thesis. I am also thankful to Kaixuan Zhou who provided advices to this thesis. This is not the first project I did with Dr. Hugo Ledoux and 3D geoinformation group, I am really grateful to the members of 3D geoinformation group who have helped me, Prof. Jantien Stoter, Ravi Peters, Filip Biljecki and Tom Commandeur.

Besides, I would like to thank CycloMedia Technology, Inc and Mickael Jonsson for providing data for this thesis.

And great thanks go to all the teachers and all my fellow students from Geomatics, Yueqian Xu, Balázs Dukai, Maya Tryfona, Matthijs Bon, Xander Duijn, Simon Griffioen, Martijn Vermeer, Fanny Bot, Oscar Willems etc. Thank you for making me feel home in Geomatics family and in the Netherlands. I really value the time we spent together and things you taught me.

I would like to thank my friend Yuwen Deng and Yiyang Liu, thank you for your support when I got depressed and distracted. And Lu Dai, thank you for your accompany and encouragement, and I will miss the time when we studied together for advanced algorithms. I would like to say thanks to all the new people I met here: Shiwei Bao and Yikai Lan, I remember the days we studied together, Shuaidong Yu, I remember the days we performed together, Bo Ma, I remember the days we hung out together. I remember and I will keep remembering, because these memories are the best gifts for me. Although I cannot list all of you here, I am truly thankful to everyone who appeared in my life in last two years.

I am so grateful to my parents and my families who support me to study abroad. Without their unconditional support, I cannot finish my study successfully. Although because of the time difference, we did not talk to each other very often, but I missed you all the time. So I really want to take this opportunity to express my love to them and I hope this thesis will be a satisfying gift to you.

Yuxuan Kang
Delft, June 2017

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

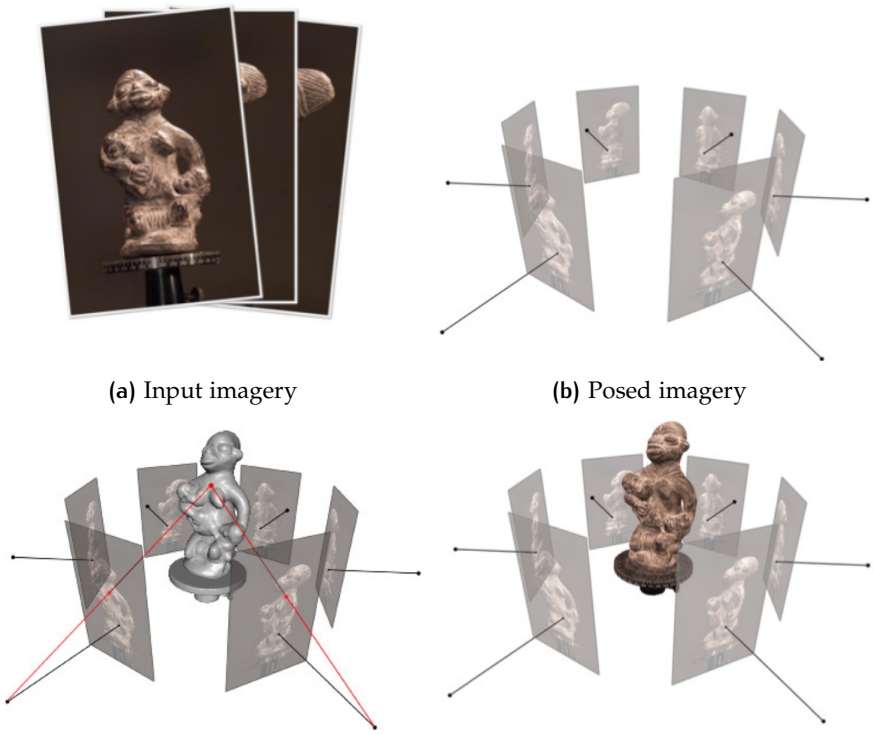
Due to the fast development of hardware and software, nowadays computers can store and handle huge data compared to the situation decades ago. The needs for 3D models are growing and expanding rapidly in a variety of fields. Compare with traditional 2D GIS, 3D indeed has the same functionality: 1) data capture, 2) data structuring 3) data manipulating 4) data analysis and 5) data prensentation [Zlatanova, 2000]. However, the difference is that 3D representation of objects is closer to reality which means it will provide more realistic feelings for 3D GIS users. As many people may have experienced, when they try to locate themselves in a city, there are always some difficulties of linking 2D maps with real city objects while people can easily link them to 3D models according to the shape, size, texture and other information. Moreover, the way of using 3D is also changing. Visualization used to be almost the only function for 3D city model, but now the function of 3D model is extended. Now there are two types of applications of 3D city model: Non-Visualization and Visualization-Based applications. For Non-Visualization application, for example there are estimation of the solar irradiation, energy demand estimation, aiding positioning, determination of the floorspace and classifying building types etc. For Visualization-Based applications, there are Geo-Visualisation and visualisation enhancement, visibility analysis, estimation of shadows cast by urban features, estimation of the propagation of noise in an urban environment, 3D cadastre, visualisation for navigation etc [Biljecki et al., 2015].

Although 3D modelling has many advantages, the fact that 3D model is hard to maintain cannot be denied. 3D models often suffer from low accuracy. These models with low quality will lead to serious error or crashing of the downstream applications [Zhao et al., 2013]. Besides, if the data has low quality, it may contain many noises so that flat objects are no longer exactly flat. Thus many studies are focusing on how to improve the quality of different kinds of 3D models with different approaches.

1.1 BACKGROUND

There are various ways for 3D representations. Among them, boundary representation is widely used in many applications. Boundary representations explicitly store topological information as faces, edges, and vertices [Lienhardt, 1991]. In computer graphics, this kind of representation is often referred as polygon mesh such as triangle mesh, and triangle mesh is often used to represent the surface. Moreover, they can be rendered efficiently with additional bitmap texture information [Wiemann et al., 2016].

The triangle mesh can be generated from different data sources as well. First, the spatial data can be collected with two kinds of sensors: active sensors such as laser scanners or passive sensors like cameras. With these two kinds of sensors, the triangle mesh can be generated based on point cloud



(a) Input imagery **(b)** Posed imagery

(c) Reconstructed 3D geometry **(d)** Textured 3D geometry

Figure 1.1: Example of a multi-view stereo pipeline from (a) to (d) [Furukawa et al., 2015]

data [Wiemann et al., 2016] or images using photogrammetry methods [Remondino and El-Hakim, 2006].

The images used for generating 3D mesh is called multi view stereo imagery. It makes it possible to generate dense meshes from the images acquired from different sources [Rouhani et al., 2017]. The generation of multi view stereo mesh is based on photogrammetric method. Furukawa et al. [2015] concluded the overall approach of generating multi view stereo mesh:

- Collect images
- Compute camera parameters for each image
- Reconstruct the 3D geometry of the scene from the set of images and corresponding camera parameters.
- Optionally reconstruct the materials of the scene

As shown in Figure 1.1, first multi view stereo images can be posed (Figure 1.1) according to the parameters of the cameras. Tie points can be found in overlapping images. If the parameters of the cameras are known, by intersecting corresponding image rays (two red lines in Figure 1.1 (c)), the 3D coordinates (x, y, z) can be calculated by image coordinates of two tie points (x_1, y_1) and (x_2, y_2). By proper sampling, points with some density can be acquired and by triangulating these points, a multi stereo mesh can be generated.

The data can be collected efficiently, with cameras mounted on cars or drones. Besides, existing airborne images can be reused to generate meshes

Figure 1.2: Multi view stereo mesh of city of Amsterdam : Amsterdam 3D (CycloMedia Technology, Inc)

as well. Thus, it is really helpful for creating and updating city models in a short time. A big advantage of image based triangle meshes is that images contain abundant texture information. The textures not only contribute to the realistic scenes but also to some analysis. For example, each vertex can be enriched with texture information. It provides more references together with other features for classification and segmentation of the mesh [Rouhani et al., 2017].

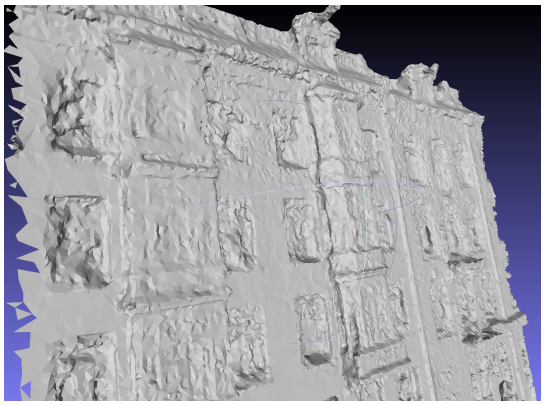
For this MSc project, the experiment data was collected by CycloMedia Technology, Inc in 2015. The datasets are multi view stereo meshes from the city of Amsterdam (Figure 1.2). The data cover 900km roads and 100km^2 area. They are collected by cameras mounted on cars. However, the roof parts will be missed by this means, so the roof data is from aerial images.

1.2 PROBLEM STATEMENT

Although the generation of multi view stereo mesh can be fast, the quality of the mesh cannot be guaranteed. There are many data error sources from data collection to the output meshes. For example, the quality of the mesh depends on the inner and exterior orientation of the camera. Because in order to get accurate meshes, the distortion of the images should be corrected, the accuracy of localizing and orientating the camera should be guaranteed. Moreover, in data processing procedure, overlapping images are matched in order to generate stereo. The accuracy of imaging matching is also a factor that has influence on the output meshes.

As shown in Figure 1.3, two subfigures show the same area of a building facade. Without texture, it is clear to see that the facade is bumpy. Due to the data errors, points often have small deviations from the actual planes they belong to. Thus these points are on either side of the plane. However, based on our knowledge about the real world, man-made objects usually have regular shapes. The facade of the building should be straight and flat. Thus bumpy objects sometimes are misleading and have a negative influence on the recognition of the objects. Moreover in many 3D city modelling, buidings are often modelled by CityGML, and CityGML model defines buildings as regular objects in different level of details (LoD) [Fan et al., 2009]. So

(a) Test data with texture



(b) Test data without texture

Figure 1.3: Bumpy facade in triangle mesh

straight and flat city object representation is more consistent with different formats of city models.

One way to solve this problem is to control the quality of data acquisition and processing. Unfortunately, although the data error sources are known, it is impossible to eliminate all the errors. And in reality, not all applications can afford accurate but expensive equipment and some applications focus more on the speed of data acquisition instead of accuracy. So the solution to improve the quality of the data is not feasible sometimes. Processing existing data is much more economic and time saving compared to re-collecting data, thus how to improve poor-quality data becomes a challenging topic.

1.3 RESEARCH QUESTIONS

As the problem is stated in the previous section, the aim of this MSc thesis is to straighten the bumpy objects in the triangle meshes based on plane fitting methods, so that the regular objects in the mesh can have straight and flat shapes. As explained in Section 1.1 and 1.2, the data have low quality. In order to detect planes from such data, RANSAC algorithm is adopted. A most important feature for RANSAC is that it can deal with data with a large portion of gross error, this will formally dicussed in Section 3.4.

The research aims to answer the following question: *"Can RANSAC algorithm based method yield similar or better result than existing approaches for*

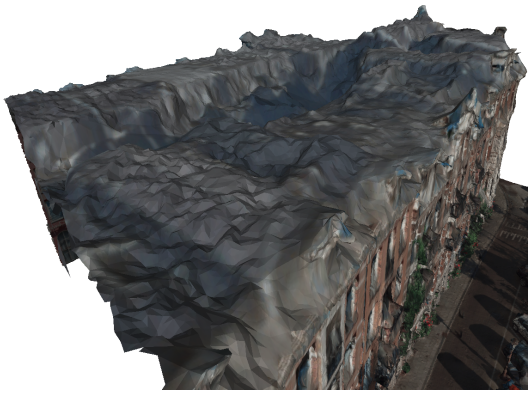


Figure 1.4: Low-quality roofs

straightening multi view stereo mesh". In order to answer the research question, there are several sub research questions need to be anwsered as well:

- What methods are currently used? What are the advantages and disadvantages?
- How can some plane constraints be used for straightening meshes?
- How can geometry/topology/texture information be used?
- Is it feasible to simplify the straightened meshes regarding data storage and attach textures to simplified meshes?

1.4 RESEARCH SCOPE

The thesis focuses especially on the man-made objects, trying to restore regular shapes of these objects and ignore unnecessary details so that the representation of the objects can be simplified. This thesis mainly works on planar areas in the city models. Since there is no semantic information in the data, according to the observation, these planar areas are often: facades of the buildings, ground, windows and doors, balconies etc. These planar objects should be straightened in the output mesh. During the procedure, many redudant details might be ignored so that the output mesh is a more simplified representation of the objects compared with the original mesh. However, some important features should be retained, such as some big windows and balconies on the facade.

By observation of the data, there are some vegetations. The vegetations are natural objects and they are often not in regular shapes. Similarly non-planar areas will remain the same in the output mesh.

Roofs of the buildings are regular so they are supposed to be straightened as well. However, the data that this thesis mainly works with has really poor-quality roofs. The roofs are generated by airborne images, due to the equipment, flight conditions etc, the quality of the roofs becomes low. Figure 1.4 shows part of the roof areas, it is clear to see that roofs are not planar in these data. Thus it is not possible to deal with these parts either. But the workflow and method can be applied to any similar data with better quality so that the roofs can be straightened as well.

1.5 RESEARCH RELEVANCE

In recent years, there are more and more ways of data acquisition. All the data collection methods have different purposes. Some of them aim at high accurate measurement, the others aim at less accurate but more convenient and fast data acquisition. Compared with 2D data, 3D data is more difficult to maintain, regarding in many aspects such as geometry, topology, texture etc. Different studies focus on soloving problems caused by low quality, for example the research from Zhao et al. [2014] is to set up a framework for geometric repair of CityGML models. Similarly, this thesis focuses on setting up a workflow for straightening multi view stereo mesh.

Some studies about how to straighten the meshes have been done with different approaches, they are formally discussed in Chapter 2. However, some methods have complex math models and steps. This research has a more understandable workflow, and it is tested on poor-quality data, which means it is more portable to data with better quality. Besides, the data comes from real application which means the research can not only have theratical values but also practical values.

1.6 READING GUIDE

There are in total 6 chapters. Chapter 2 organises some related works. Chapter 3 explains conceptual framework, workflow and principles of some algorithms related to this research. The details of implementation and experiments are presented in Chapter 4. Then following Chapter 5 will focus on the analysis of the result and make comparisons with other study of this topic. Finally, conclusions will be drawn and the research question will be answered in Chapter 6. Moreover, some existing problems, recommendations and protential future works will be given in the final chapter as well.

2 RELATED WORK

Several relevant aspects of this thesis have been studied by other researchers. The following chapter gives an overview of the related studies. Some related works cover more than one aspect, so they will be discussed more than once with different perspectives in different sections.

There are four parts in this chapter. First, Section 2.1 discusses some enriched features used in different studies for segmentation or classification. Second, Section 2.2 provides an overview about different approaches for segmentation. Section 2.3 introduces some related studies about plane detection. Finally, Section 2.4 focuses on how to regularize and refine the fitted planes. The regularization of the plane is to make the fitted plane more regular considering the relations to the other planes. The refinement of the plane is to improve the quality of the fitted plane.

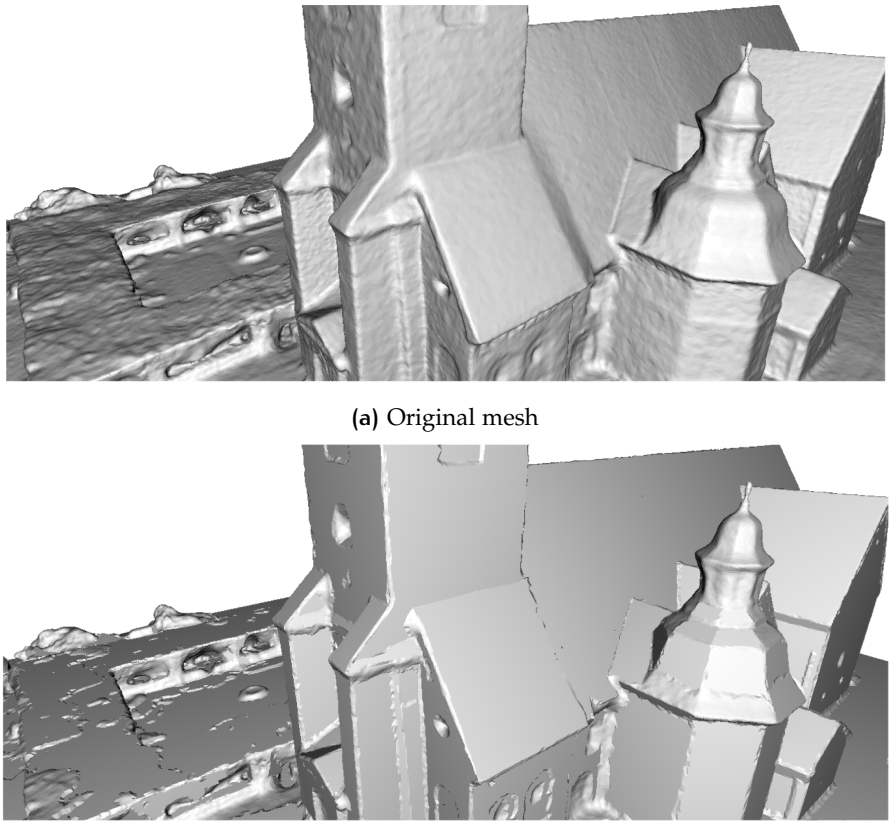
In 3D city modelling, there are not many studies fully focus on multi view stereo mesh. Jonsson [2016] did a similar topic on multi view stereo mesh, so in the sections of this chapter, the study from Jonsson [2016] will be discussed first and in details. Besides, there are plenty of research about point cloud. Due to some similarities between point cloud and multi view stereo mesh, for exmple they both contain dense points, the studies about point cloud are also relevant to this thesis. Moreover, there are many studies about segmentation, classification on images from (multispectral) remote sensing. Since multi view stereo mesh in many cases contains also spectral information (normally referred as texture), the studies about segmentation and classification of images are also considered relevant.

2.1 FEATURE ENRICHMENT

For multi view stereo meshes, the primitives are vertex, edge and face. Among them, vertex contains geometry information while topology information is formed by edges which connect adjacent points. In order to segment the points well, only using geometry and topology information is usually not enough. So some other features are enriched to the points. In this section, some common features that can be enriched to points or other segmentation unit such as triangle will be discussed.

As mentioned before, Jonsson [2016] also did research in detection and correction of planar regions in triangle meshes. Figure 2.1 shows his result.

The most relavant feature used in this MSc thesis is curvature. In 2D space, each point on a curve has a corresponding tangent vector, then curvature is the measurement on the rate of change of this tangent vector. So it describes how much a curve deviates from a straight line. Extend this notion to 3D, since each point on the curved surface has non-unique tangent vectors in different directions, the curvature coresponds to chosen tangent vector, which means it is not unique either. Among all the curvatures, the two principle curvatures are defined as the maximum and minimum cur-



(a) Original mesh

(b) Flattened mesh

Figure 2.1: Result from Jonsson [2016]

vatures. These two values are input to the defined probablity function to caculate the probablities of a point is a planar point. As a sequent step, Markov random field (MRF) is used to make decision on labelling planar points. More detailed theoratical definition of curvature of a surface is explained by Taubin [1995], while [Douros and Buxton, 2002] provides more insights on 3D surface curvature estimation with quadric surface patches. Curvature feature Jonsson [2016] used is a good indicator for planarity. The points in planar areas have low curvature, thus planar areas can be detected well by this feature. However, curvature estimation can achieve decent results when the data have good quality. If the data contain too much noise, curvature might not be estimated accurately.

In point cloud related studies, the normal is one of the most important feature apart from geometry of the point [Sampath and Shan, 2010]. Besides, they also mentioned that the eigen values of the covariance matrix formed by neighboring points can indicate whether a point is planar or not. If the normalized eigen value is small enough, the point should be planar.

Many studies of point cloud also tend to combine point cloud data with another data for information enrichment, for example, Demir and Baltsavias [2012] used a combination of point cloud data and multispectual image to automatically model building roofs. Although the study does not enrich the features of point cloud directly, it does use additional features for point classification. First, it uses slope as feature to separate ground and non-ground points, then NIR (Near InfraRed), R (red), G (green), B (blue) are used for classification to distinguish buildings, bare ground, roads, shadows, grass

and trees. Additionaly, in order to better classify trees, point cloud vertical density can also be a feature since it is generally much higher at trees than at open terrain or buildings. Their study gives an inspiration of combining color information and geometry information flexibly to better process points. A more related work was carried out by Martinovic et al. [2015]. The same as the study of Demir and Baltsavias [2012], the data is a combination of point cloud and image. For Random Forest (RF) classifier, the following features are used: mean RGB values of the point as seen in images, LAB values of the mean RGB, normal of the point, spin-image descriptor [Johnson and Hebert, 1999], height of the point above estimated ground plane and so forth, in total 132-dimensional feature space.

In many mesh related studies, classification is a neccesity, thus features are important factors in the research of this area. Verdie et al. [2015] classified based on superfacet instead of individual triangle facet. These superfacets are clusterd based on shape operator matrix [Cohen-Steiner and Morvan, 2003] and image analysis. The features are enriched on each triangle facet instead of point: the relative elevation, planarity derived from surface variation [Pauly et al., 2002], and horizontality. These features are normalized within 0 and 1. Then the features for each superfacet are caculated as area-weighted sum of the features for each facet, also the same procedure takes on the normal of the superfacet. For more examples, Valentin et al. [2013] used surface curvature, singular values extracted from principal component analysis (PCA), shape diameter feature (SDF), shape contexts (SC) and spin images as geometric features for Conditional Random Field (CRF) energy model.

In conclusion, considering the data of this project, normal features is considered necessary and it can be enriched on each vertex based on normal estimation method or based on normals of the faces. There are too much noise in the data, thus near planar parts may become bumpy, so that curvature and planarity are assumed to be less distinctive. The data is also textured so color information can also be used since facade, windows and other objects are usally in different colors.

2.2 SEGMENTATION

The segmentation of mesh is a necessary part in many applications of computer graphics [Katz et al., 2005]. Besides, segmentation can also be appied to different kinds of data such as image, point cloud etc. Because of the different formats of the data, the segmentation can base on points in point cloud, pixels in images, vertices or faces in mesh etc. Although the data are different, the idea of segmentation can be identical. The aim of the segmentation is to integrate the basic unit such points, pixels etc into segments, thus more anlayses can be applied on the segments instead of individual basic unit.

In this section, in order to get an complete overview, all different forms of segmentation on different kinds of data are discussed

First, there are some mesh related studies which apply segmentation methods. Jonsson [2016] segmented the mesh by region growing and it relies on the label assigned to each vertex. Planar segments are kept for plane fitting and non-planar segments are disgarded. Besides, in order to find both big planes such as building facade and small planes like windows and chimney sides, plane fitting approach is carried on different scales by

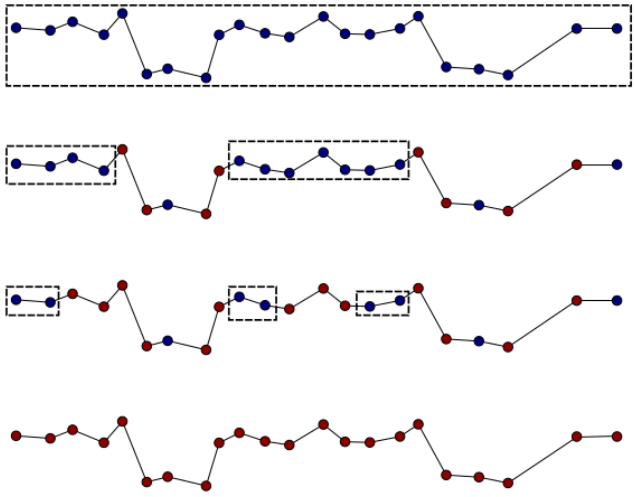


Figure 2.2: Segmentation on different scales [Jonsson, 2016]

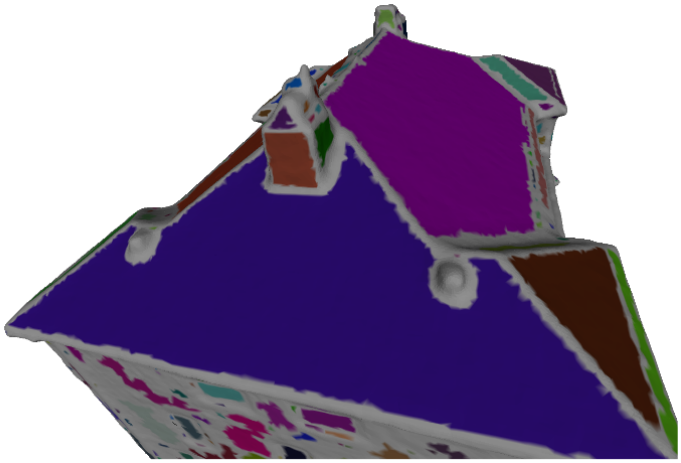


Figure 2.3: Curvature segmentation result [Jonsson, 2016]

using scaling factor σ. This scaling factor is a parameter to control the labeling for points. Figure 2.3 shows segmentation on different scales from largest scale (top) to smallest scale (bottom) , and the blue points are labelled as planar points. In the largest scale in Figure 2.3, all the points will be segmented as a big cluster, thus a big plane will be fitted and the fitted planes become smaller and smaller by decreasing the scale. It provides an idea for this project that plane fitting can be carried on different scales to get different size of planes of the buildings.

Verdie et al. [2015] also applied region growing, but the region growing is not based on points but on face segments of the mesh. First faces of the mesh are oversegmented into superfacet based on region growing by comparing ths similairy of shape operator matrix. The shape operator matrix is estimated for each triangle facet on a local spherical mesh neighborhood with radius R. Then they use Markov Ramdom Field to label superfacet from the geometric attributes computed per superfacet. After labelling, they identify a set of nearly planar superfacets by selecting the ones labelled as roof or facade. For each of them, least square is applied for plane fitting.

In many point cloud related studies, region growing methods assign label to each point, for example the research from Elberink and Vosselman [2009], Sampath and Shan [2010] and Sun and Salvaggio [2013], while Orthuber and Avbelj [2015] proposed a different TIN-based region growing where the label is assigned to each triangle instead of point. So TIN-based region growing starts from a seed triangle, iteratively include more homogenous triangles into a segment. Since for triangle meshes, each face is also a triangle so this TIN-based region growing can be applied the MVS mesh as well. Further more, in order to imporve the performance of region growing, Chauve et al. [2010] designed a seeding strategy that points of better planarity have priority to be seed points. This will make region growing more efficient.

Despite of region growing based segmentation, there are some other studies of point cloud which adopt other segmentation methods i.e clustering[Filin and Pfeifer, 2006]. **?** mentioned an edge-based method which can determine the edges in the data set and connect them to form regions. Besides they concluded some types of available clustering [Jain et al., 1999][Berkhin, 2006][PEH, 2007] : hierachical methods including bottom up and top down, partitioning methods such as k-means, model based methods and density based methods. Among them, they used fuzzy k-means clustering and the number of clusters are estimated by calculating likelyhood. Alharthy and Bethel [2004] also used clustering method but the clustering is also based on region growing.

Woo et al. [2002] gives another perspective for point cloud segmentation. Unlike traditional point cloud segmentation which is based on point, this method is based on subdivision of 3D grid. A original 3D grid is created with relatively large voxel size and the size of the voxel decreases by iteratively subdividing the voxel. A voxel is not divided until the deviation of points in this voxel is low enough. Similarly base on voxel, Tseng and Hung [2016] applied split-and-merge segmentation [Wang and Tseng, 2010][Wang and Tseng, 2011] using octree structure.

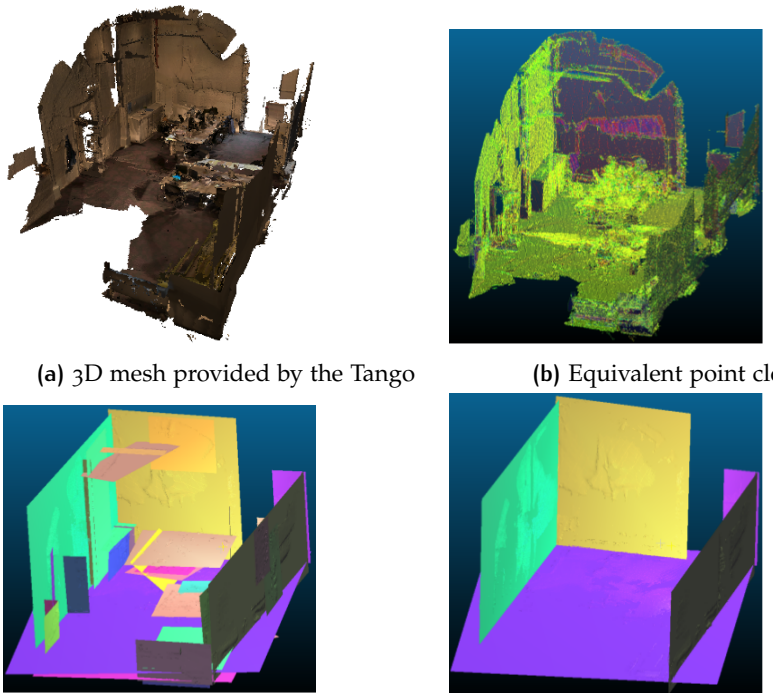
Rouhani et al. [2017] concluded the simplest form of mesh segmentation may be seen as unsupervised clustering problem based on geometic criteria [Shlafman et al., 2002]. Besides, there are some deterministic approaches such as region growing [Koschan, 2003], spectral analysis [Zhang et al., 2010] and some probabilistic approaches such as MRFs or CRFs. In general, region growing seems to be the most popular method for segmentation both on points and image pixels.

For this project, since the mesh contains both point geometry and image texture, region growing can be suitable for mesh segmentation. There are also some studies which labelling the mesh segment, but for this project, thereg can be semantic enrichment but it is not a focus.

2.3 PLANE DETECTION

Building is one of the most important type of object in 3D city modeling. And it normally consists of several planes, for example walls, flat roofs etc [Fan et al., 2009]. The following section will provide some related works with the aspect of plane dectection.

RANSAC is a widely used method for model fitting. It was proposed by Fischler and Bolles [1981] in image analysis and automated cartography. They mentioned that RANSAC was capable of intepreting/smoothing data



(a) 3D mesh provided by the Tango

(b) Equivalent point cloud

(c) Extracted patches

(d) Patches corresponding to walls and floor

Figure 2.4: Planar patch detection using RANSAC [Diakité and Zlatanova, 2016]

with a significant percentage of gross errors. Roth and Levine [1993] did a statistic research showing that this percentage can be even 50%. Since shapes are often parametric model, many studies use RANSAC to detect shapes.

Gallup et al. [2010] applied RANSAC on depth map to detect planes. Schnabel et al. [2007] used RANSAC for point cloud shape detection. The aims are not only planes, but also cylinders, cones. Each point fixes only one parameter of the shape. Diakité and Zlatanova [2016] applied RANSAC on TANGO tablet scanning data to detect indoor planar patches, the result is shown in Figure 2.4.

Tarsha-Kurdi et al. [2008] proposed an extended RANSAC algorithm for roof detection from point cloud. The extension of RANSAC includes two parts: improve the data quality and improve RANSAC algorithm. The algorithm can be improved in two aspects:

- The shape does not necessarily include as many points as possible. Before the algorithm iteratively replaces the shape with less points with the one with more points, standard deviation needs to be considered in order to decide whether it should be replaced.

- If a detected plane does not meet the requirement, the points that fit this plane should be reassigned to original cloud.

It provides some ideas for this project about how to control the quality of the shapes detected by RANSAC.

Besides RANSAC, the Hough Transform [VC, 1962] is a classic shape detection method for detecting parameterized objects, typically lines and circles. It is often applied on image data, but it is not a necessity. For example,

Borrmann et al. [2011] used Hough Transform to detect planes in 3D point clouds. Duda and Hart [1972] explained that, in 2D space, each figure point in original space is a straight line in Hough space and if there are many lines intersect at one point in Hough space, the intersection point indicates a line in original space. This notion can be extended to 3D. Borrmann et al. [2011] explained a method that transform each point to a Hough space defined by Duda and Hart [1972], where each point in original space is a surface in this Hough space. If the intersection curves between different surfaces intersect at one point, this point is a indication of a plane in original space. Tarsha-Kurdi et al. [2007] explained more details about the implementation of 3D Hough Transform algorithm, and how to apply it on automatic detection of 3D building roof planes.

In conclusion, there are several methods which are capable of detecting shapes in 3D applications, for example, Hough Transform, RANSAC and some other ways such as tensor voting [Kim et al., 2009]. Tarsha-Kurdi et al. [2007] compared Hough Transform and RANSAC in their experiment. The conclusion is RANSAC algorithm provides not only results in a shorter time but also the quality of the result is higher. As explained in Section 1.3, the data this project uses have low quality, and RANSAC has strong ability to deal with data containing much noise. So for this project, RANSAC will be adopted as plane fitting method, and it is also a key factor for this project. More principles and details of RANSAC will be give in Chapter 3.

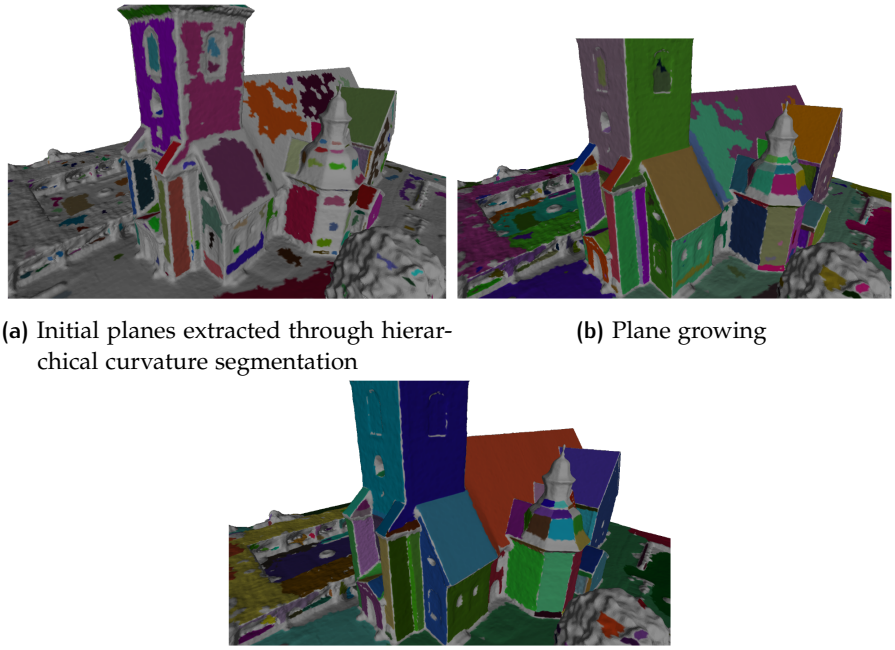
2.4 PLANE REGULARIZATION AND REFINEMENT

Since plane fitting is a key component, the quality of the fitted plane is vital. In city model, buildings are usually relatively regular shaped objects, thus some rules can be applied to get better models.

Jonsson [2016] did plane refinement by plane growing and merging. First, planes are placed restrictively according to the segmentation result, which means the initial planes will be small. Second the plane will grow to include more triangles into the plane. Then the planes will be merged combining small planes, if the total vertex-to-plane projection error for the combined plane is not too high. Figure 2.5 shows the result for each step.

Kada and Wichmann [2013] detected planar segments from point cloud then each planar segment can be mapped to planar half-space [Mäntylä, 1988]. A single 3D building model can be described as the collection of planar half-spaces (Figure 2.6). Based on this half-space modeling, Wichmann and Kada [2014] proposed 3D building adjustment by regularizing planar half-space. There are two scales for the adjustment: local and global adjustment. Local adjustment is for regularizing the shape of one single building based on the fact that the component of most buildings are symmetric and regular (i.e. with 90° corners). Thus there are three steps:

- Slope adjustment: half-spaces with similar slope are adjust to their average value.
- Orientation adjustment: half-spaces with similar x-y directions are adjust to their average angular value by rotating around z axis.
- Position adjustment: vertical half-spaces are moved be sysmetric and regular.



(a) Initial planes extracted through hierarchical curvature segmentation

(b) Plane growing

(c) Planar region merged

Figure 2.5: Plane refinement [Jonsson, 2016]

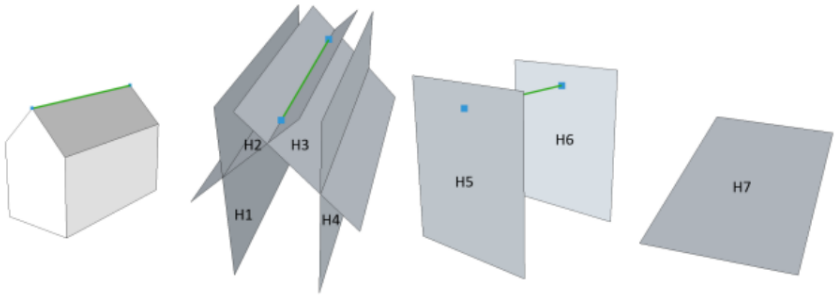


Figure 2.6: Pitched roof building described by half-spaces from H1 to H7 [Kada and Wichmann, 2013]

Global adjustment uses the similar concept but it is applied on all the buildings. Because buildings in a region often follow some patterns (i.e the facades of the buildings might all face towards south in an area).

The thesis adopts the method from Verdie et al. [2015], a detection-then-regularizaiton strategy. First the geometric relationships between different planes are detected, including parallelism, orthogonality, Z-symmetry, coplanarity. Then the orthogonal and z-symmetric planes will be re-orientated and the coplanar planes will be re-positioned. More details will be discussed in Section 3.5.

3 CONCEPTUAL FRAMEWORK TO STRAIGHTEN CITY MESH

The following chapter provides the conceptual framework to straighten the mesh, mainly focusing on the theoratical aspects of the project. First Section 3.1 shows the workflow and a diagram is given to illustrate the steps involved. In this section, the reason for such a design is explained as well. Then the following sections describe some important methods and algorithms used in the workflow. Section 3.2 introduces the method used for estimating normal for each point. Section 3.3 characterises texture information and how it can be enriched to each point in the mesh. Section 3.4 introduces the principle and some characteristics of RANSAC algorithm which is used for plane detection. Besides, the explanation of its parameters will be given in this section as well. Section 3.5 explains the method adopted in this project to regularize the planes. Section 3.6 introduces two kinds of snapping operations. Section 3.7 gives detailed steps and criteria for segmenting the mesh. Section 3.8 provides details about how to split the segment after local fitting in order to detect and remove spikes. After the mesh is straightened, Section 3.9 explains the way to simplify the mesh so that the data storage can be reduced.

3.1 METHODOLOGY

In general, the method consists of two scales: global scale and local scale. The aim of global scale is to detect large planes for example building facade, ground, roof etc. Global fitting takes precedence of local fitting because these parts of city objects will make a sketch of the city model. That is to say, assume that these planes are already detected, the frame of the city models is known. Besides, no matter in point cloud data or multi view stereo mesh, these large planes contain most of the points, it is much easier to detect these parts than small details. For these reasons these parts should be processed first, and the leftover points in this stage will be ones that describe the details of the models. In the following local scale, these leftover points from global scale will be the input. The aim of local scale is to detect small planes that fit well in local area, in order to straighten small objects such as windows, balconies etc.

Figure 3.1 shows the workflow of the project. The loop formed by the green boxes are global fitting and local fitting operations. First the texture information will be enriched to each vertex in the mesh. Then the normal of each vertex is estimated by the normal of its incident faces. After these two steps of information enrichment, the points are input to Random Sample Consensus (RANSAC) algorighm to detect planes, and right after it is regularizing the detected planes. With all these planes, points will be snapped to their corresponding planes. Snapping operation is to project the point to the plane to get the intersection point, then relocate the point to this intersection point. After snapping, the global fitting is done, then the points

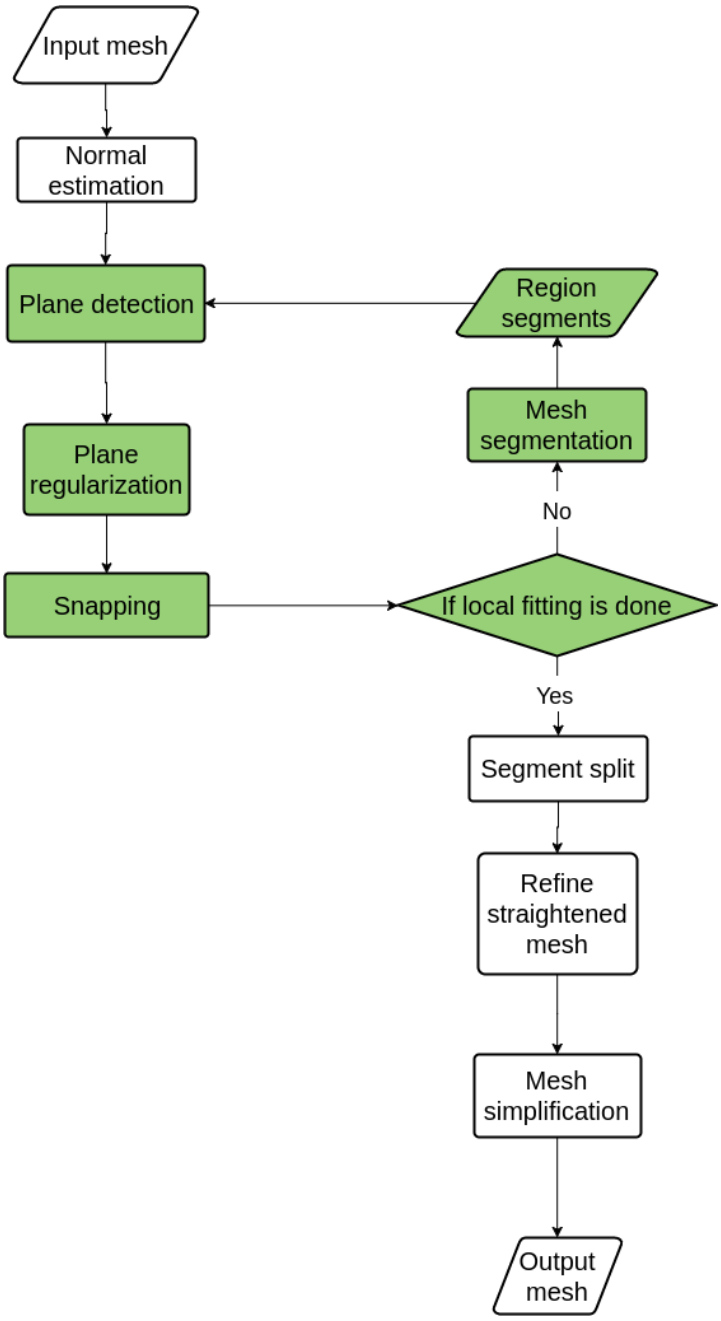


Figure 3.1: Workflow diagram of straightening and simplifying Multi View Stereo mesh

which are not snapped in global fitting will be segmented into region segments. In local fitting, each segment will be input seperately to RANSAC to get smaller planes, so different paramters are used. Then the same as global fitting, it is followed by plane regularization and snapping. After local fitting, most points are already snapped to one plane. Points snapped to the same plane will become a segment in this stage. In the next step, the segments will be splitted into smaller segments defined as spikes. These spikes will be removed by snapping operation again. The following step is mesh simplification. Since many parts of the mesh are already straightened

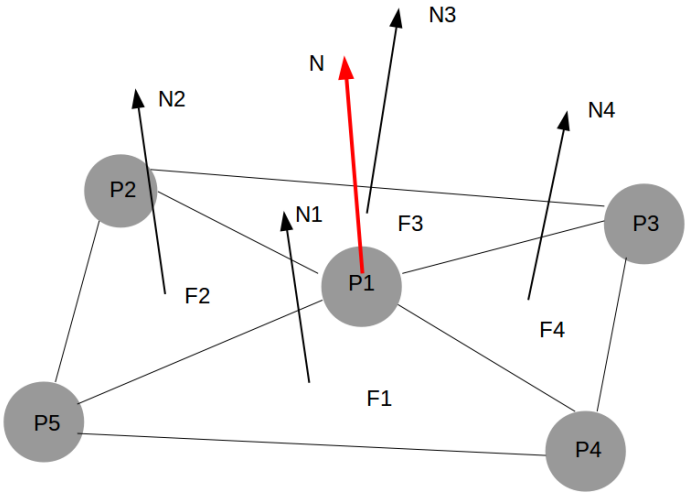


Figure 3.2: Normal estimation by the normal of incident faces

before this step, the vertices can be thinned according to the flatness of the neighborhood.

3.2 NORMAL ESTIMATION

By definition, the normal of a point in a continous surface should be the normal of the tangent plane of the point. In surface sampled by points, the normal of the point is normally estimated by its neighbor points, and there are various ways for estimating normal [Klasing et al., 2009] such as plane singular value decomposition (SVD) [Hoffman and Jain, 1987][Huang and Menq, 2001] and plane principle component analysis (PCA).

However, this project adopts a simple and efficient normal estimation method. Because compare to point cloud , multi view stereo mesh has topology information, thus it is not neccessary to find nearest neighbors. Considering topology information, each vertex shared by several faces, the normal can be estimated by all the normal of incident faces. The normal of a vetex n_i can be calculated as follows:

$$n_i(x,y,z) = \sum_{j=1}^{n} n_j(x_j,y_j,z_j) \quad j \in \{ \text{ all in incident faces of vertex } i \ \}$$

$$|v| = 1$$

As shown in Figure 3.2, the normal of P_1 is N, and the normals of the incident face F_1, F_2, F_3, F_4 are N_1, N_2, N_3, N_4 respectively. N can be estimated by the normal addition from N_1 to N_4. With the method above, each point will contain normal information after this step, which is neccessary for plane detection.

3.3 TEXTURE INFORMATION ENRICHMENT

Apart from normal information, texture information can also be enriched for each vertex. UV mapping is a texture mapping method for projecting

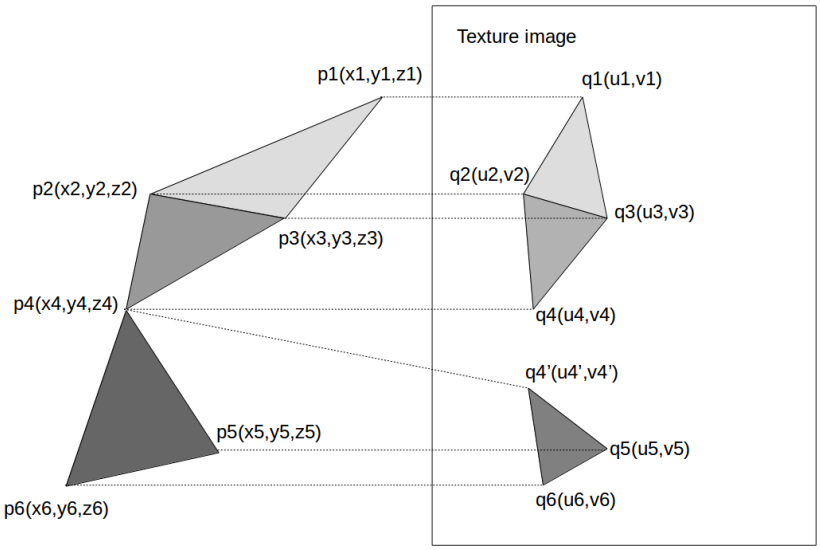


Figure 3.3: UV mapping from triangle mesh to texture image

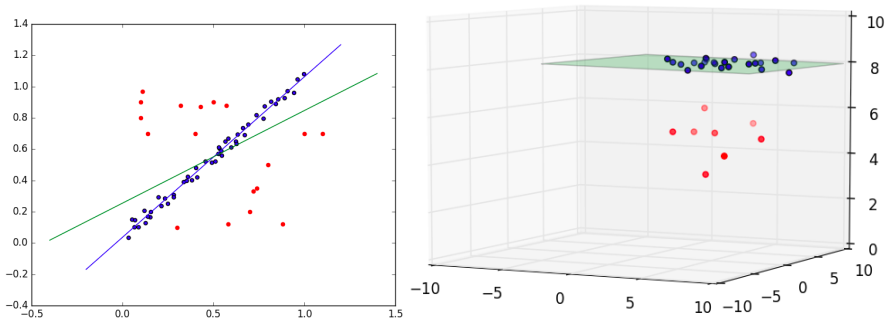
2D image to 3D model surface. For a triangle mesh, UV coordinates can be generated for each vertex. The triangle mesh is unfolded at the seams with each triangle laying on a flat page. Thus each triangle has three UV coordinates for three vertices, and these three UV coordinates define a triangle part on the texture image which can be automatically linked to its corresponding triangle in the mesh (Figure 3.3). Besides, UV coordinates are optionally applied per face [Murdock, 2008] so that a shared vertex can have different UV coordinates for its related triangles. Thus adjacent triangles can be totally separated in texture image. For example in Figure 3.3, $p_4(x_4, y_4, z_4)$ are linked to $q_4(u_4, v_4)$ and $q'_4(u'_4, v'_4)$, thus triangle $T(p_2, p_3, p_4)$ and triangle $T(p_4, p_5, p_6)$are positioned in different areas in texture image.

UV coordinates range is normalized to [0,1]. In order to get RGB information for each vertex, there should be a coordinate transformation from UV coordinates to image coordinates. For this project, the origin of UV coordinates is bottom left corner and the origin of image coordinate is top left corner, equation 3.1 is the transformation from UV coordinates to image coordinates. From image coordinate (x, y), the RGB value of the image can be acquired.

$$\begin{cases} x = \lfloor u \times width + 0.5 \rfloor \\ y = \lfloor (1 - v) \times height + 0.5 \rfloor \end{cases} \tag{3.1}$$

3.4 RANDOM SAMPLE CONSENSUS ALGORITHM

Random sample consensus (RANSAC) [Fischler and Bolles, 1981] is a widely used method for parameter estimation of a mathematical model. The most important feature for RANSAC is that it can deal with data with a large portion of gross error. There are two important terms for RANSAC: inlier and outlier. Compared to least squares which fits model based on all the



(a) Blue line is fitted by RANSAC and green line is fitted by least squares

(b) Plane fitting by RANSAC

Figure 3.4: Model fitting on data with inliers (blue points) and outliers (red points)

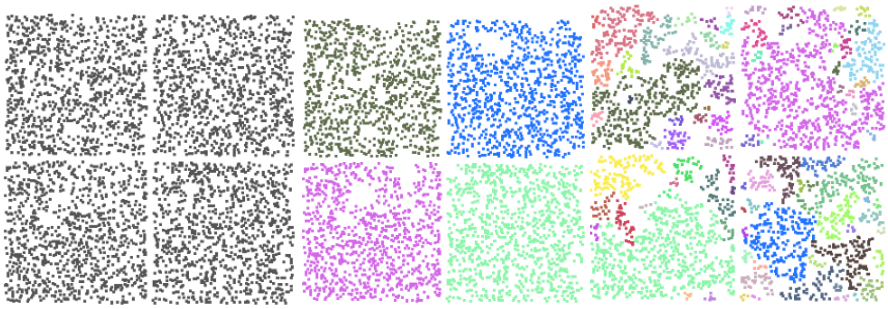
input data, RANSAC will detect which points can fit a model best, and the other points will be ignored. So RANSAC is also an outlier detection method. Figure 3.4 (a) shows the result from RANSAC and least squares in line fitting in 2D space, it is clearly to see the result from least squares will be influenced by outliers and RANSAC fits a better line. The same with plane fitting shown in subfigure (b). Because of this characteristic, RANSAC is capable of robustly dealing with data containing more than 50% of outliers [Schnabel et al., 2007].

RANSAC can be described as the following steps:

1. Randomly choose a set of points as inliers. The number of points should be just enough for estimating the model.
2. Estimate the parameters of the model based on these inliers.
3. Check all the other points whether they can fit to the estimated model within a error tolerance. If so, add all these points (inliers) to consensus set.
4. If there are enough points in consensus set, use all these points to compute a new model, otherwise repeat the previous steps.
5. After predetermined number of trials, if a consensus set with more points cannot be found, return the model fitted by the largest consensus set.

There are some parameters required for RANSAC. In the context of plane detection, the following parameters are needed for this project:

- epsilon (ϵ): It defines the absolute maximum tolerance Euclidean distance between the point and the plane. Only points within the epsilon will be included as inliers.
- normal threshold (σ): The threshold of the deviation between the estimated normal of the point and the normal of the plane. The deviation is described as dot product of two normals, and the closer the dot product is to 1, the smaller deviation is between the two normals. For inliers, the deviation should not exceed this threshold.
- cluster epsilon (E): Clustering of the points into connected components covered by a detected shape is controlled via parameter E, the



(a) A large value for cluster epsilon E leads to detecting a single planar shape

(b) A moderate value for cluster epsilon E yields the detection of four squares

(c) A small value for cluster epsilon E leads to oversegmentation

Figure 3.5: Parameter cluster epsilon E controls the connectivity of the points covered by a detected shape. The input point set is sampled on four coplanar squares.[Oesau et al., 2017]

influence of this paramter is shown in Figure 3.5. A large cluster epsilon will lead to relatively large planes and some details will not be distinguished while small cluster epsilon yield more detailed plane detection but might lead to oversegmentation as well. Figure 3.6 shows results from different cluster epsilon E.

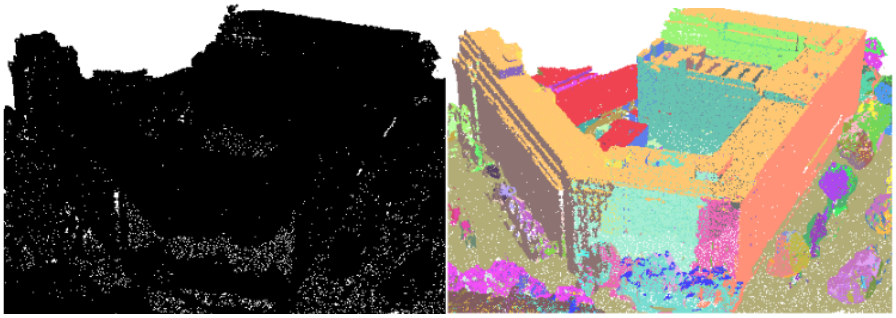
- probability (p): RANSAC cannot always ensure the model is estimated by the largest consensus set, this parameter defines the probability of missing the largest plane. A lower probability provides more reliable results but it leads to more iterations and runtime.

- min points (n): It defines the minimum points used for estimating the model. However this parameter is not strict, based on the chosen probability, planes may also be detected by a lower number of points.

The parameters of RANSAC are important for detecting planes. For this project, there are two scales, the parameters should be set differently for them.

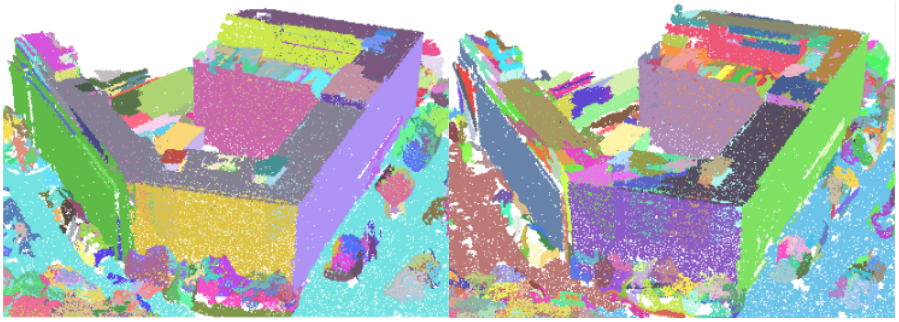
For global fitting, the aim is to detect large planes such as facade, thus n should be a large number. Besides, in order to avoid oversegmentation in this stage, E should be relatively large as well. Because in global fitting, the detected planes are main planes, the quality of them should be guaranteed. And the points on the facade are normally regular than the other parts, σ should be close to 1 and ϵ should be small enough to strictly throw out outliers and avoid generating spikes. It is meant that all the window points etc will be regarded as outliers, then they will be detected in the local fitting. p should be small so that it allows more iterations to get more reliable detections.

For local fitting, small planes such as windows should be detected thus n and E should be small. For this data, the normal of the points in small parts have more deviations, so σ should be relaxed but ϵ should still be small to avoid spike problem. p should be small in order to ensure the quality of the detected planes.



(a) Input point set

(b) cluster epsilon E set to 2.0 (one color per detected shape)



(c) cluster epsilon E set to 0.5

(d) cluster epsilon E set to 0.25

Figure 3.6: Impact of cluster epsilon E over level of details of the detection [Oesau et al., 2017]

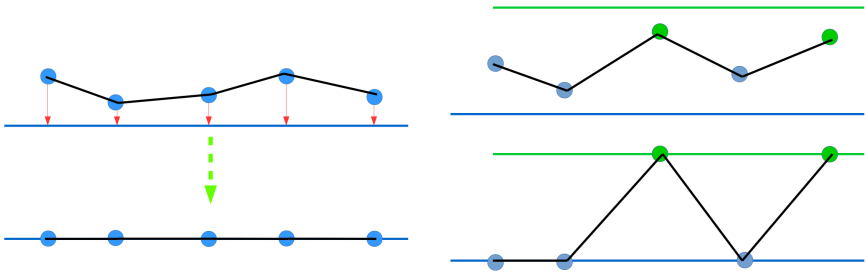
3.5 PLANE REGULARIZATION

With the proper design of the parameters, RANSAC can detect generally reasonable planes, however, it only considers the distribution of the points instead of considering the city model as a whole. For example for a building model, there should be some constraints with the plane components [Wichmann and Kada, 2014]. For example, Figure 3.7 shows a problem caused by unregularized planes. After plane detection, points as inliers will be projected on the plane and moved to the location of projected point, this operation is referred as snapping in this project (Figure 3.7 (a)). However, if two planes are near coplanar as shown in Figure 3.7 (b), there are two planes fitted on these points, two green points are inliers of the green plane and three blue points are inliers of the blue plane. After snapping, they will be projected to their corresponding planes, then some spikes will appear. Thus, plane regularization is neccessary.

For this project, the method from Verdie et al. [2015] is adopted. They defined 4 types of geometric relationships between planes as follows:

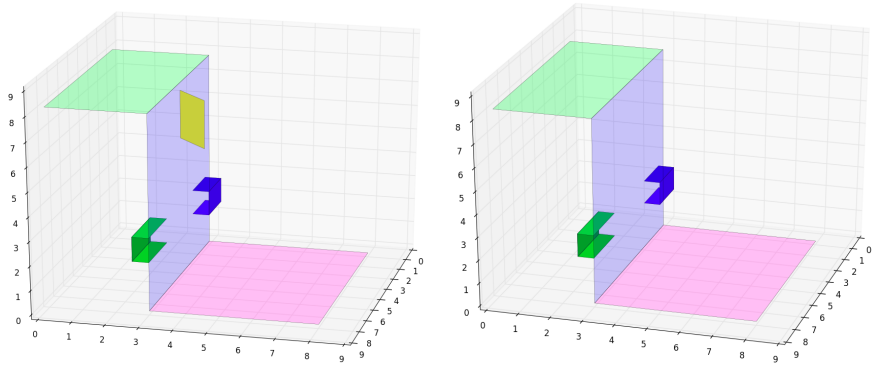
let P_1, P_2 be two planes, n_1,n_2 be their normal vectors respectively, c_1 be the centroid of points fitted to P_1 and c_2 be the centroid of points fitted to P_2.

- Parallelism. P_1 and P_2 are ϵ-parallel if $|n_1 \cdot n_2| \geq 1 - \epsilon$.
- Orthogonality. P_1 and P_2 are orthogonal if $|n_1 \cdot n_2| \leq \epsilon$.
- Z-symmetry. P_1 and P_2 are ϵ-Z-symmetric if $||n_1 \cdot n_z| - |n_2 \cdot n_z|| \leq \epsilon$. n_z is the vertical z-axis. Z-sysmmetry is mainly for regularizing roofs.



(a) Point snapping **(b)** Near coplanar planes problem

Figure 3.7: Snapping points and spikes problem caused by near coplanar planes



(a) Planes fitted in global fitting and local fitting **(b)** Remove unregularized planes

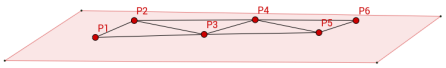
Figure 3.8: Regularization of small planes

- Coplanarity. P_1 and P_2 are d-ϵ-coplanar if they are ϵ-parallel and the $|d_\perp(c_1, P_2) + d_\perp(c_2, P_1)| < 2d$.

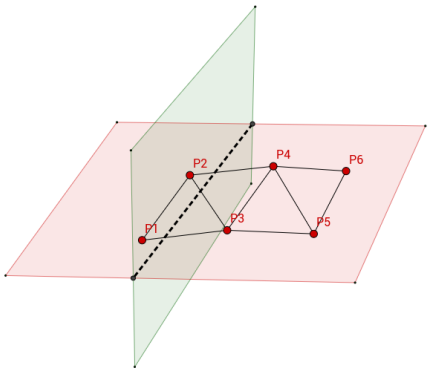
For this project, these four geometric relationships are reused. Based on them, near parallel, orthogonal, Z-symmetric and coplanar are made exactly parallel, orthogonal, Z-symmetric and coplanar respectively. [Oesau et al., 2017].

Apart from the rules explained above, another problem is that RANSAC does not consider if a detected plane is actually valid in reality. In local fitting, because the parameters of RANSAC are relaxed and sometimes it is difficult to get decent segmentation result, some unreasonable planes might be detected. Thus it is neccessary to regularize the detected planes in this stage. Figure 3.8 shows the simulating planes from global fitting and local fitting. There are three big planes fitted in global fitting, and several small planes fitted in local fitting. In order to remove unreasonable planes such as the yellow plane in Figure 3.8 (a), first the directions of the main planes are stored in a set N. Then all the small planes should be either parallel or orthogonal to one of the main direction in N. In Figure 3.8, there are three main planes with two main directions, clearly the yellow plane does not satisfy the constraints so it will be removed (Figure 3.8 (b)).

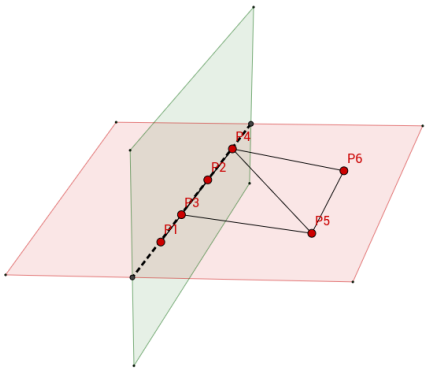
(a) Original mesh and fitted planes



(b) Snapping vertices to the corresponding plane



(c) Original mesh and two fitted planes



(d) Snapping vertices to the intersection line

Figure 3.9: Snapping operations

3.6 SNAPPING

Snapping is an important operation in this project. The advantage of snapping is that it is simple and it keeps the original topology of the mesh. There are two kinds of snapping operations:

- Snapping points to planes: After planes are detected by RANSAC, different points belong to different planes as inliers. These inliers are projected to the planes and they are moved to their projected positions. By snapping the points to the planes, bumpy surfaces can be straightened.
- Snapping points to intersection lines: In global fitting, there are some main planes detected. The intersection lines of these planes can be the corners. In order to sharpen the corners, the points near these intersection lines are projected to the lines and moved to their projected positions.

Figure 3.9 shows two types of snapping operations in this project. By snapping the vertices to their corresponding plane, the mesh will be straightened. And by snapping points to the intersection lines, the corners will be more regular and clean. The results are shown in Section 4.5.2.

3.7 MESH SEGMENTATION

After global fitting, the points on the large planes such as facade are snapped. The leftover points are windows, doors and other details on the main plane or some other noisy objects like vegetations or cars etc. In order to detect planes from these points and straighten them, first it is neccessary to segment them then detect planes on each segment.

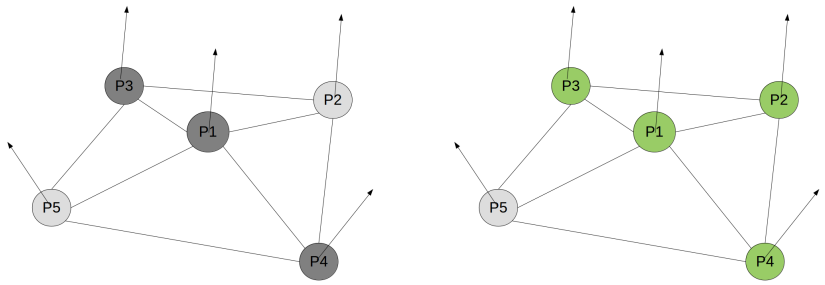
There are various methods to segment a mesh. As it is mentioned in Chapter 2, region growing is one of the most popular method used for segmentation, so it is adopted in this project. There are three kinds of information used in mesh segmentation. First, topology relationship is used for finding the connected points. It is explicit information for a mesh because the neighboring points of a vertex share edges with it, thus they are connected in the mesh. As explained in Section 3.2 and 3.3, every vertex in the mesh already contains normal and texture information. Based on these three kinds of information, Algorithm 3.1 shows the steps of region growing and it can be described as follows:

1. Add seed point p_{seed} to a region
2. For each point p_i in the region, find its unsnapped neighbor points p_j.
3. Compare the similarity of the normal between p_{seed} and p_j, if it is within a threshold, add p_j to the segment.
4. If not, compare the RGB distance between p_{seed} and p_j, if the distance is within a threshold, add p_j to the segment.
5. Repeat step 1 to step 4 until no more point is added to the segment
6. Start a new segment with another seed point.
7. Continue until every unsnapped point belongs to a segment.

Algorithm 3.1: Mesh segmentation: Region growing (Breath First Search)

```
Data: Points
Result: List of segments
while Points is not empty do
    segment.add(p_seed)
    Points.remove(p_seed)
    for point p_i in segment do
        for neighbor point p_j of p_i do
            if p_j not in segment & p_j is not snapped then
                if |n_seed · n_j| > threshold σ then
                    segment.add(p_j)
                    Points.remove(p_j)
                else if RGB(p_seed)-RGB(p_j) < threshold λ then
                    segment.add(p_j)
                    Points.remove(p_j)
        end
    end
    if segment.size > threshold n then
        SegmentList.add(segment)
end
return SegmentList
```

Manhattan distance is used for measureing RGB distance for this project since it is simple, fast and it performs the best for evaluating the similarity of texture compared with Euclidean distance, Vector Cosine Angle distance



(a) connected vertices with texture and normal information **(b)** Segment (green) after region growing

Figure 3.10: Region growing based on topology, normal and texture information

[Vadivel et al., 2003]. The definition of Manhattan distance in terms of texture is shown in Equation 3.2:

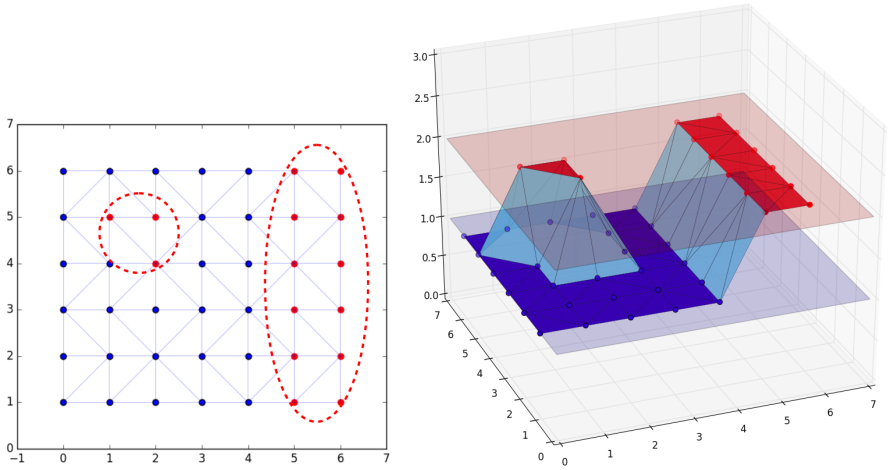
$$D(p_i, p_j) = |R(p_i) - R(p_j)| + |G(p_i) - G(p_j)| + |B(p_i) - B(p_j)| \quad (3.2)$$

Figure 3.10 shows an example result of this method. After P_1 is added to the segment as seed point, its adjacent points P_2 to P_5 will be regarded as candidates. Since P_2, P_3 have similar normal as P_1, they will be added to the segment. The normal deviation between P_1 and P_4 is relatively large, however their RGB distance is small, so P_4 will be added to the segment as well. Finally P_1 to P_4 will belong to the same segment, and P_5 will be excluded.

This region growing method takes both normal and texture into consideration. The advantage is that according to the observations to textured mesh, segments normally have similar colors. And there might be some errors when estimating normals so if region growing is only based on normals, some points which are supposed to be in the same segment will be separated for example P_1 and P_4 in Figure 3.10. However, normal should still be the prior criteria because it is an indicator to planarity and texture is not key factor for a segment. For example, some areas on the facade are in shadows, so they have totally different colors from the other part of the facade but all the facade points are supposed to be in the same segment. Moreover, the number of points that a segment contains should be larger than a threshold, otherwise no reliable planes can be fitted on the segment.

3.8 SEGMENT SPLIT

After local fitting, most points are already snapped to one certain plane, while some still remain unsnapped. According to the plane they are snapped to, a label will be assigned to each vertex. Based on these labels, all the points are classified to different big segments. These segments are substantially formed by RANSAC algorithm, so points in the same segment have similar normals and they can fit to the same plane model. However, RANSAC does not consider the topology of the points, which means the points in the same segment may be separated in space. As shown in Figure 3.11, in previous plane fitting step, points are snapped to different planes, the blue points are snapped to blue plane and the red points are snapped to red plane. Based on the planes they are snapped to, there are two seg-



(a) Points belong to one segment are separated in 2D space

(b) Spikes problem in 3D space

Figure 3.11: Disconnected segment and spikes problem

ments the red one and the blue one at this stage. However, in Figure 3.11 (a), the red segment actually has two disconnected components, circled by dash lines. In Figure 3.11 (b), the 3 red points make a spike in the mesh, in order to remove the spike, it is neccessary to split the red segment into two separated segments.

Region growing is again adopted but with a different rule. The following is the description of the algorithm and pseudo code (Algorithm 3.2):

1. Add seed point to a region
2. For each point p_i in the region, find its neighbor points p_j.
3. Check if p_i and p_j have the samel label (snapped to the same plane), if so, add p_j to the region.
4. If not, mark p_j as an edge point, and record the label of p_j.
5. Repeat step 2 and 4 until no more point is added to the segment.
6. Start a new segment with another seed point.
7. Terminate until every point belongs to a segment.

After the algorithm, segments which contain disconnected components are split so that the points in one segment are all connected. The connectivity between different segments are checked when splitting the segments. Then a segment is defined a spike if:

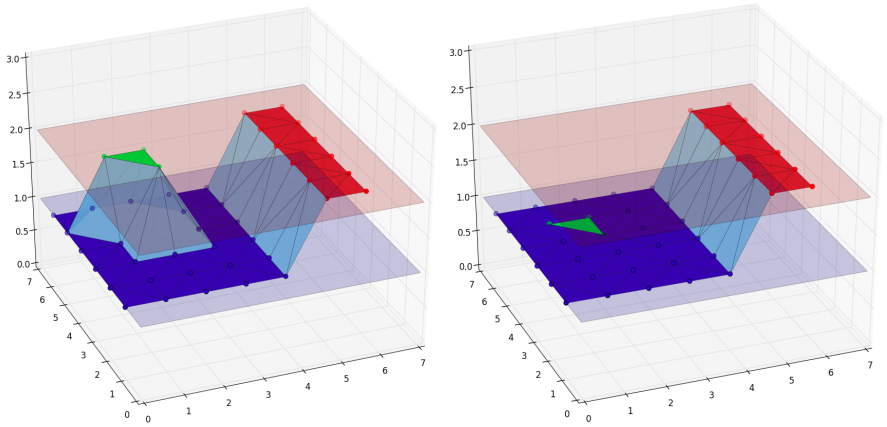
1. The number of points in the segment is less than a threshold n.
2. More than half of the edge points of this segment are connected to the same segment.

If more than half of the edge points of the segment i are connected to another segment j, merge these two segments and snap all the points in segment i to the plane corresponding to segment j.

As shown in Figure 3.12, the original red segment (Figure 3.11) is split into two segments, the green one and the red one. The green segment

Algorithm 3.2: Segment split

Data: Points
Result: List of segments
while *Points is not empty* **do**
  segment.add(seedpoint)
  Points.remove(seedpoint)
  for *point* p_i *in segment* **do**
    for *neighbor point* p_j *of* p_i **do**
      if p_j *not in segment* & $label(p_j) \neq label(p_i)$ **then**
        segment.add(p_j)
        Points.remove(p_j)
      else if p_j *is snapped* **then**
        $Connectivity(segment, label(p_j))$++
    end
  end
  SegmentList.add(segment)
end
return SegmentList



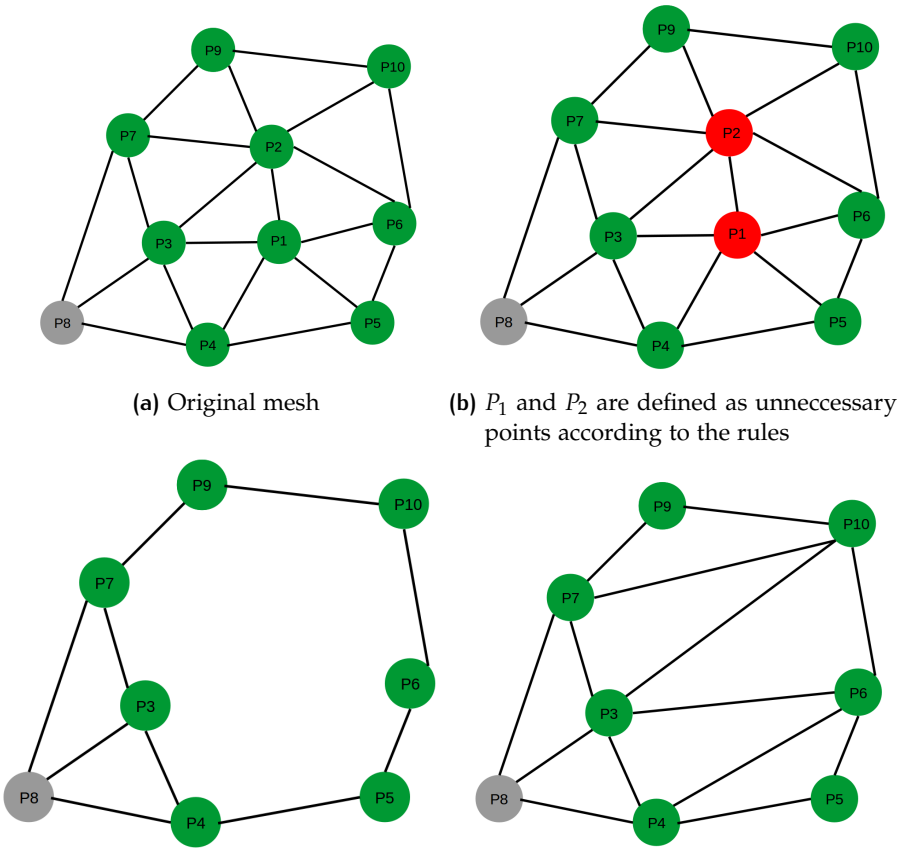
(a) Split red segment into two segments **(b)** Refine the mesh by removing the spike (green and red)

Figure 3.12: Split segment and remove spikes

only contains three points and all its edge points are connected to the blue segment, so it is an "island" on the blue segment. By definition the green segment is a spike and all the three points are snapped to the blue plane so that the mesh will be refined.

3.9 MESH SIMPLIFICATION

After straightening the multi view stereo (MVS) mesh, many points are already snapped to planes. Many triangles are on the same planes, so it is not necessary to keep all the triangles. In order to reduce the data storage, two steps are adopted to simplify the mesh: (1) remove unnecessary vertices (2) retriangulate the mesh. In removing vertices step, there are three rules:



(a) Original mesh

(b) P_1 and P_2 are defined as unneccessary points according to the rules

(c) Remove unneccessary points

(d) Retriangulate the mesh

Figure 3.13: The mesh simplification process: from (a) to (d)

- If a vertex and all its adjacent vertices in mesh are snapped to the same plane, it is defined as an unnecessary vertex thus it will be removed.
- If a vertex is not snapped to any plane, it will be kept.
- If a vertex has unsnapped neighbors, it will be kept.

Retriangulation of the faces is based on 2D constrained Delaunay triangulation instead of 3D [Shewchuk, 1997]. According to the rules in removing unnecessary vertices step, only one type of faces in all the original faces will be removed, that is the face with 3 vertices all snapped to the same plane, all the other faces can be kept. So among all the points that are kept, only points that are snapped need to be triangulate. The 3D coordinates of these snapped points are transferred to 2D coordinates in the plane they are snapped to, then they are triangulated in 2D space. These newly generated triangle faces are added to the face list, then the simplified mesh can be acquired.

In order to fill the empty spaces shown in Figure 3.13 with new triangles, it is required to get segments of edge points of these empty spaces, for example in Figure3.13, the green points should be put into the same segment, and these green points which are needed for triangulation are connected by edges that only have one incident face, so other edges with more than one incident will be removed first. Then, a edge based region growing is carried out. The region grows based on edge structure instead of vertex.

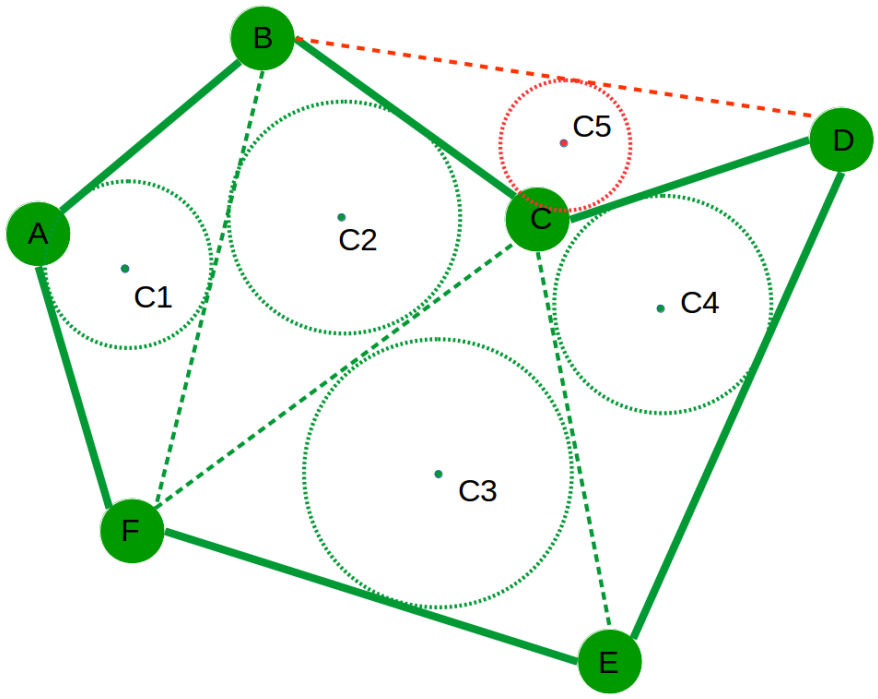


Figure 3.14: Incenter of new triangles

Edge based region growing starts from a seed edge and the edges grow in one direction. It includes the incident edge in every step until no more edges can be added into the segment. For example, in Figure 3.14, AB is a seed edge, then in next step, edge BC or AF will be included. Assume BC is included after AB, then CD, EF, FA will be included in the segment consecutively. Finally, the vertices are collected from the segments of edges, so vertices $ABCDEF$ will be in the same segment.

Since the empty spaces sometimes can be concave polygons while 2D Delaunay triangulation generates convex triangulation area, it is neccessary to constrain the new triangles. In this thesis, incentre is calculated for each triangle. Assume the triangle ABC has three vertices $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, and three edges $a = BC$, $b = AC$, $c = AB$.

$$a = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}$$

$$b = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}$$

$$c = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The incenter points $C(x_c, y_c)$ can be calculated as:

$$x_c = \frac{ax_1 + bx_2 + cx_3}{a + b + c}$$

$$y_c = \frac{ay_1 + by_2 + cy_3}{a + b + c}$$

As shown in Figure 3.14, polygon $ABCDEF$ is concave. Newly generated triangles are $\triangle ABF$, $\triangle BFC$, $\triangle FCE$, $\triangle CDE$, $\triangle BCD$ with the incenter points C_1, C_2, C_3, C_4, C_5 respectively. Since C_5 is outside the concave polygon $ABCDEF$, so $\triangle BCD$ will be removed.

4 IMPLEMENTATION AND EXPERIMENTS WITH REAL-WORLD DATASETS

This chapter explains the implementation details and some experiments on test data sets. This chapter is organised based on the workflow described in Section 3.1, so each section explains one particular step in the workflow. First Section 4.1 introduces some tools and libraries used in the project. Section 4.2 describes the data used for this project, and gives more detailed description on the test data sets. Section 4.3 and 4.4 shows some results on normal estimation and texture information enrichment. Section 4.5 and 4.6 describe detailed implementation in global fitting and local fitting respectively. Section 4.7 shows the implementation and experiments on segments split and removing spikes. Finally Section 4.8 explains the implementation and experiments of mesh simplification.

4.1 TOOLS AND LIBRARIES

There are some tools and libraries used for this project. For the development, object-oriented programming language C++ is used. The most important library used is CGAL, it provides various geometric algorithms in the form of a C++ library [The CGAL Project, 2017]. Some important packages of CGAL used for this project are listed in the following:

- Polygon Mesh Processing [Loriot et al., 2017] : Normal estimation.
- Point Set Shape Detection [Oesau et al., 2017] : Plane detection and plane regularization.
- 2D Triangulation [Yvinec, 2017] : Mesh simplification.

CImg [Tschumperlé, 2012] is a lightweight, open-source C++ toolkit for image processing. In this project, it is used for texture information enrichment. Besides, there are some software used for this project. FME is used for converting 3D data format. Cloud Compare [Girardeau-Montaut, 2017] implements plane detection so it is used for some preliminary results. Meshlab [Cignoni et al., 2008] is mainly for visualization of the mesh.

4.2 DATA

As briefly mentioned in Section 1.1, the data was collected by CycloMedia Technology, Inc. The data was collected by different means. The building facade, ground etc is collected by cameras mounted on cars and the roof parts are from airborne images. As explained in Section 1.1, by photogrammetric method, overlapping images can be matched and the coordinates of each pixel in the image can be calculated. By resampling and triangulation, the

multi view stereo mesh can be generated. Because the data is from different source, the quality is not the same in one data set. For these data sets, the roof parts have much lower quality.

The data sets are 3D city models of the city of Amsterdam. The data is originally in COLLADA format (usually with .dae file extension), which is for interactive 3D applications. In order to operate more easily, the data is converted into OBJ file (with .obj file extension).

OBJ file represents 3D geometry with the following components:

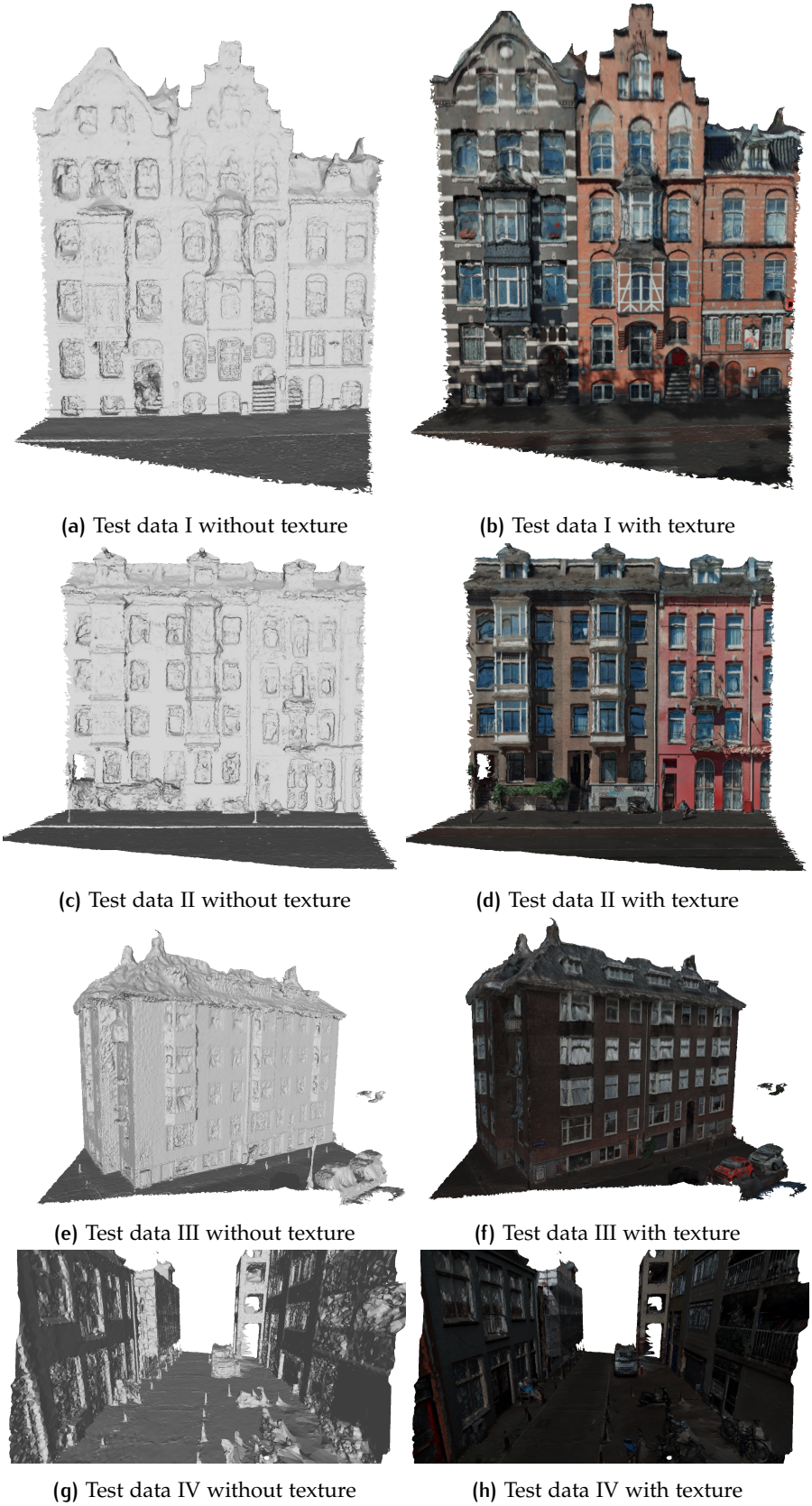
- geometry of vertex: $v\ x\ y\ z$
- vertex normals: $vn\ x\ y\ z$ (normal is not necessary normalized)
- UV coordinates: $vt\ u\ v$
- face, depending on what information is available, face can have different formats, for example: $f\ \ v_1/vt_1\ \ v_2/vt_2\ \ v_3/vt_3$ or $\ v_1/vt_1/vn_1\ \ v_2/vt_2/vn_2\ \ v_3/vt_3/vn_3$ (v, vt and vn represent the indices of vertex, UV coordinates and normal respectively).

Apart from OBJ file, each OBJ file is linked to several texture files which are in JPEG data format. MTL material file (.mtl) is the link between OBJ and texture files.

The data are clipped into different files and each one only covers a small part of Amsterdam. In order to test if the method works well in different kinds of situations, there are four test date sets in these experiments and they are shown in Figure 4.1. Different data sets have different features:

- Test data I consists of two gable roof buildings and one flat roof building, it also contains part of the ground. It contains 52879 vertices and 104842 faces.
- Test data II has also several buildings, so it is similar to test data I, but the buildings have different shapes. There are some noises in this dataset, for example, some vegetations are attached to the facade. The size of the data is a little bigger than test data I with 54962 vertices and 109082 faces.
- Test data III has one single building but compared with test data I and II, it also contains more than one corner. Besides, it also has some other objects such as the cars on the ground. There are 91898 vertices and 182114 faces in test data III.
- Test data IV is a more complex scene. It has multiple buildings and the difference is that the buildings are on the two sides of the road. Besides, there are some extra objects on the road such as cars, bench etc. The size is a little larger than test data III with 100466 vertices and 199348 faces.

The whole workflow will operate on each test data set in later sections. For each test data set, some implementation details will be explained and the results and analyses will be given after important steps.



(a) Test data I without texture

(b) Test data I with texture

(c) Test data II without texture

(d) Test data II with texture

(e) Test data III without texture

(f) Test data III with texture

(g) Test data IV without texture

(h) Test data IV with texture

Figure 4.1: Test data

Face index	Vertex indices/UV indices	Texture file
1	1757/1757 1758/1758 1759/1759	fx-i-0.jpg
2	5024/5024 5025/5025 1758/1758	fx-i-0.jpg
...	...	...
5746	44245/45513 44246/45514 44247/45515	fx-i-1.jpg

Table 4.1: Relationship between faces and texture files

Vertex index	UV coordinates	Texture file	Image coordinates	RGB
2045	(0.258845 0.976459)	fx-i-0.jpg	(207,24)	(197,125,28)
2045	(0.391349 0.095430)	fx-i-0.jpg	(313,905)	(200,145,30)
2045	(0.612067 0.534668)	fx-i-1.jpg	(490,465)	(180,130,35)

Table 4.2: Relationship between vertex and RGB value

4.3 NORMAL ESTIMATION

The OBJ file is loaded and if there is no normal information contained, the normal will be first estimated. It is implemented by CGAL Polygon Mesh Processing, computing normals package. As explained in Section 3.2, the normal is estimated for each vertex, as the average of its incident face normals.

4.4 TEXTURE INFORMATION ENRICHMENT

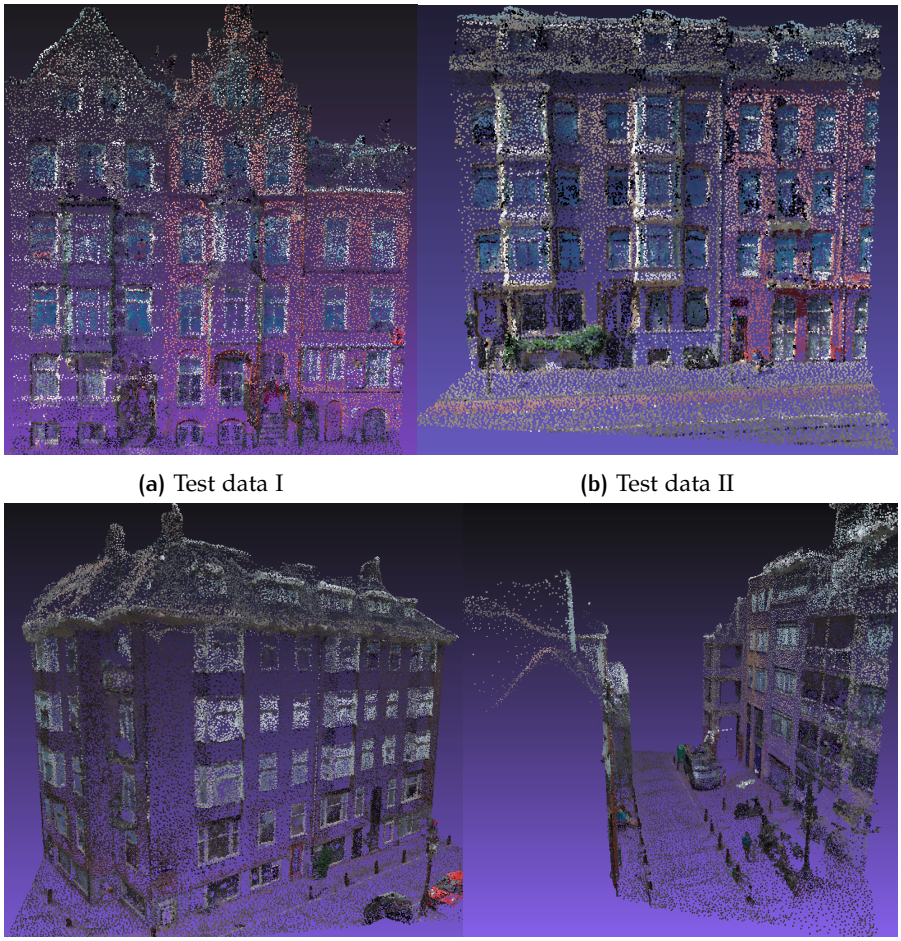
In the original data set, vertices do not contain texture information. In order to attach texture information to each vertex, the link between vertex and the pixel should be found. In OBJ file, different faces may link to different images, it is neccessary to store the relationship between face and image. The relationships is described in Table 4.1. According to this relationship, each vertex will be related to a pixel of a certain image using its UV coordiantes. The Equation 3.1 shows the relationship between image coordinates and UV coordinates. Each vertex can be related to several pixels from different images, so the average RGB value is calculated and assigned to this vertex, this relationship is shown in Table 4.2.

After this step, each vertex is enriched with an RGB value, then the vertices with color can be visualized by generating a new mtl file, defining different colors, and OBJ file can be linked to this mtl file so that each vertex can be colored. The texture enriched vertices are shown in Figure 4.2. It is clear to see that vertices are correctly enriched with texture information.

4.5 GLOBAL FITTING

4.5.1 Main plane fitting

After normal estimation and texture enrichment, the first step is global fitting. In this step, the whole test data will be the input and parameters should be strict in order to get accurate main planes such as facade, ground, roof etc. As mentioned in Section 3.4, RANSAC parameters setting is an important factor for the performance of the method. Parameters should be set properly to adjust to the test data set, and they should be set differently



(a) Test data I **(b)** Test data II

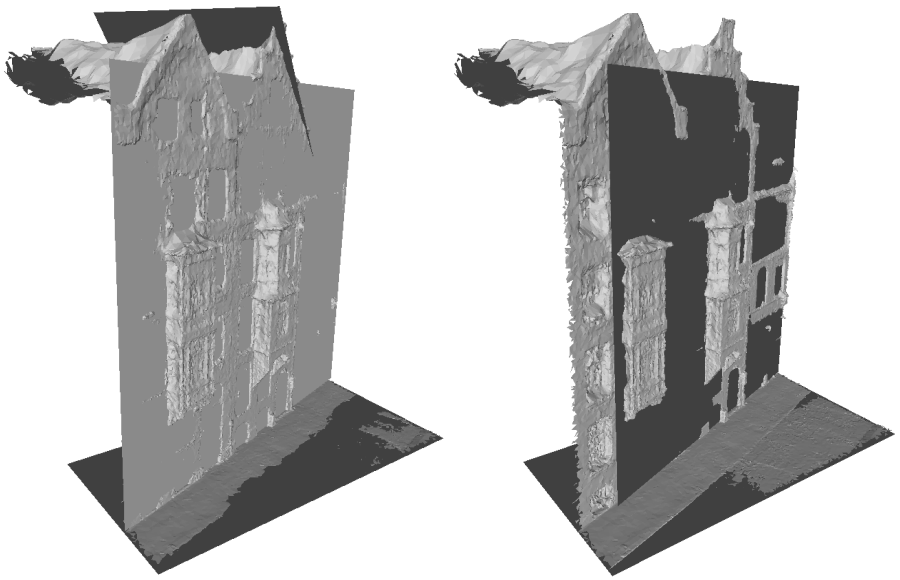
(c) Test data III **(d)** Test data IV

Figure 4.2: Textured vertices

for global fitting and local fitting. In order to get a better understanding of the influcence of some important parameters for global fitting, test data I is used as an example to test different settings of parameters. And since RANSAC has some randomness because it randomly chooses the hypothetical inliers (see Section 3.4), the tests on parameters are performed several times to reduce the influence of randomness on the conclusions.

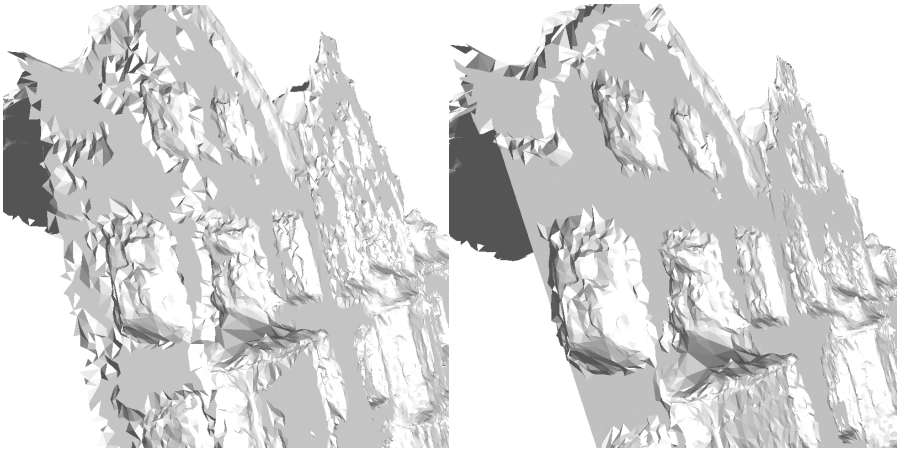
First, different probabilities are tested. Two extreme examples are chosen, one is when $p = 0.05$ and the other is when $p = 0.95$. As shown in Figure 4.7, when $p = 0.95$ the facade plane is not extended to cover the whole facade points, the facade plane is detected and it already satisfies the requirement for minimum number of points for estimating the plane, so the iteration stops. Compared with the situation when $p = 0.05$, the facade plane is better fitted and includes all the facade points, moreover, an extra roof plane is detected while it is often ignored when $p = 0.95$. So in global fitting, probability p should be set as a very low number.

Min points (definition see Section 3.4) is also an important parameter. It determines the size of the planes detected in this stage. Figure 4.4 shows the results when $n = 200$ and $n = 1000$. 14 planes are detected when $n = 200$, these planes include facade, roof, ground and some windows. When $n = 1000$, only 3 planes are detected. The aim of global fitting is only to find main planes, so that the windows and other details should be left for



(a) $p = 0.05$ **(b)** $p = 0.95$

Figure 4.3: Different probabilities p



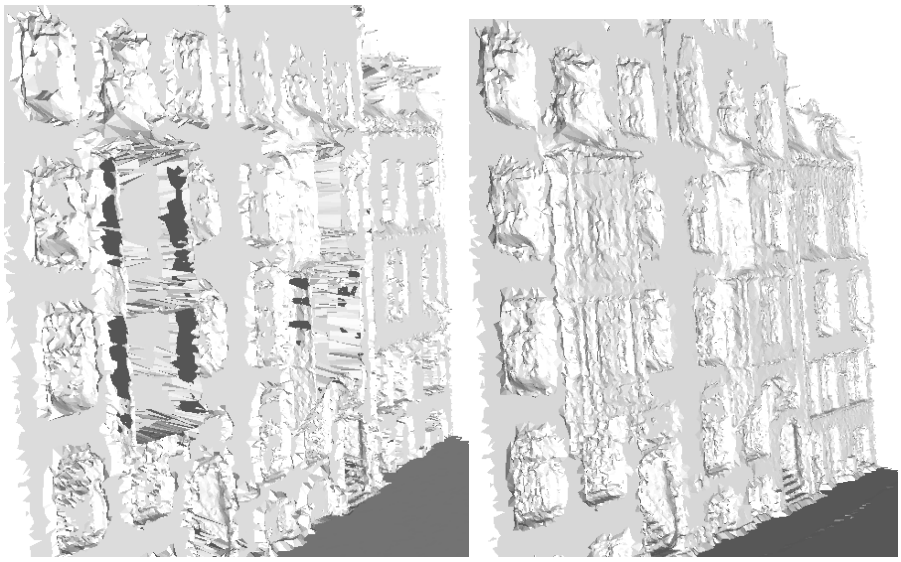
(a) $n = 200$ **(b)** $n = 1000$

Figure 4.4: Test on parameter min points n

local fitting, thus n should be reasonably large. Besides, if n is small, instead of fitting a complete facade plane, all the facade points will be divided into several parts and be fitted to different planes since a plane does not require many points. This will probably lead to spikes problem (in Figure 4.4).

Epsilon ϵ determines how close the point to the plane should be. Large ϵ will lead to spike problem while small ϵ will make less points fit to a model but fit more accurate model. Figure 4.5 shows when $\epsilon = 0.5$, many points that are actually far away from the facade plane are snapped to the facade, for example many points on the window are moved to the facade. This causes severe problem to the model, so in order to avoid this, ϵ must be a small number.

Besides above mentioned parameters, there are two parameters cluster epsilon E and normal deviation σ. Since in global fitting, it only focuses on the main planes, some details can be ignored, thus E should be relatively



(a) $\epsilon = 0.5$ **(b)** $\epsilon = 0.03$

Figure 4.5: Test on parameter ϵ

large. And for points on the main surfaces such as facade, the quality of the normal is relatively high, so the normal deviation should be small. To conclude the parameters setting, in general the parameters used for global fitting can be set as:

- probability $p = 0.03$
- min points $n = 1000$
- epsilon $\epsilon = 0.05$
- cluster epsilon $E = 1$
- normal threshold $\sigma = 0.98$

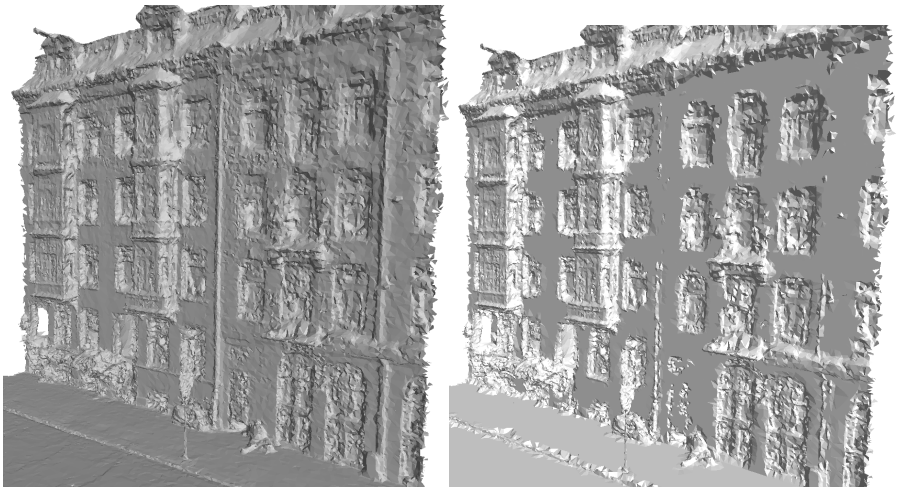
However, depends on the quality of the results, the parameters can change a little to adjust to the test data better.

4.5.2 Snapping

As explained in Section 3.6, there are two types of snapping in global fitting. First, after detecting planes, inlier points are snapped to their corresponding planes. Second, points that are close to the intersection lines of the main planes, will be snapped to the intersections between planes so that the corners can be sharpened.

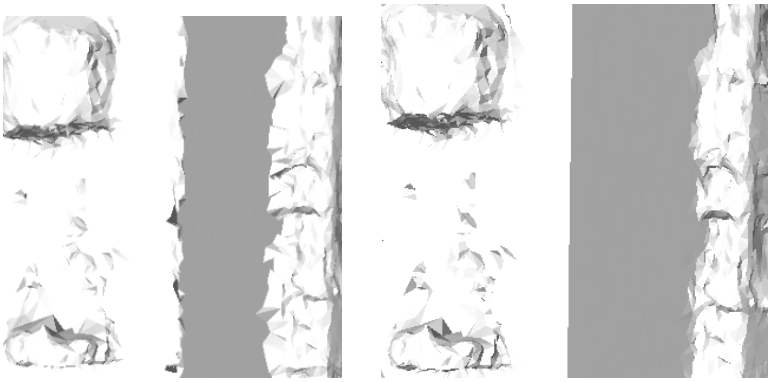
Figure 4.6 shows the result of test dataset II before and after snapping points to planes. It can be seen that after snapping, the facade and the ground are straightened.

Because the normal of the edge point is usually not regular, which means it is not in the direction of one of the two main planes, and the direction is usually in between. Thus these points will not be snapped to the main planes, leading to ragged edge problem. Figure 4.5 (a) shows the ragged edge problem, and after snapping points to the intersection lines, the quality of the edge is much improved.



(a) Test dataset II before snapping **(b)** Test dataset II after snapping

Figure 4.6: Global fitting result of test dataset II



(a) Test datset III before snapping **(b)** Test datset III after snapping

Figure 4.7: Comparison between before and after snapping points to intersetion lines

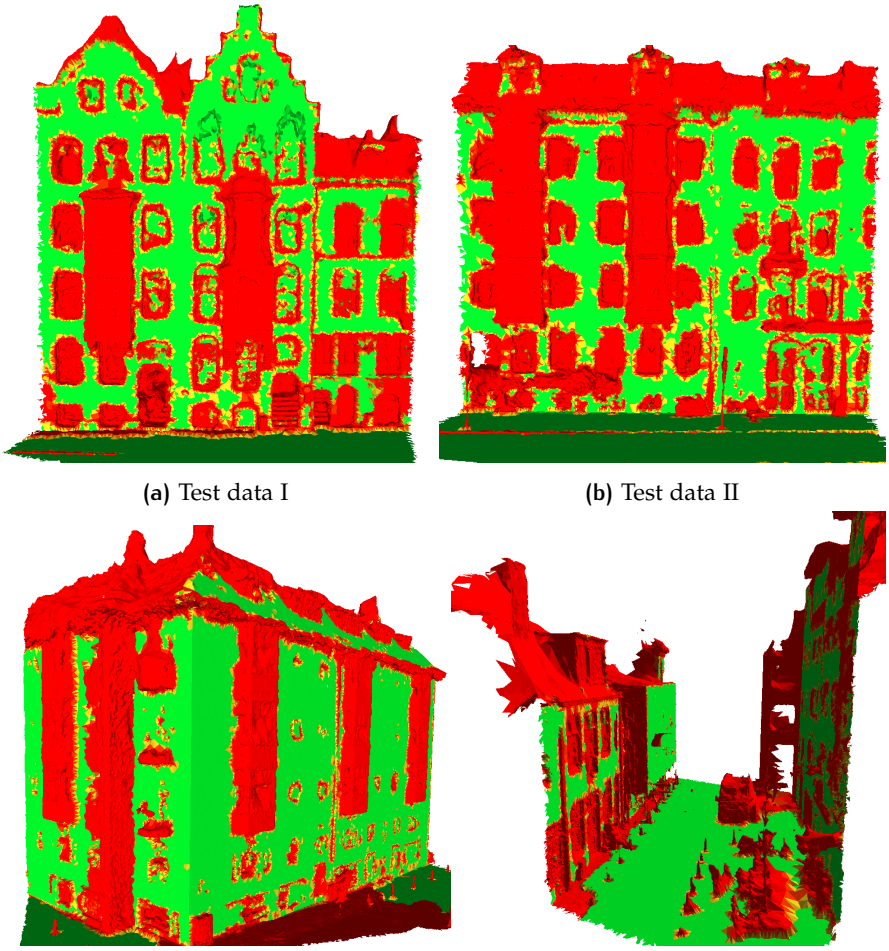
After global fitting, points on main surfaces are snapped. In order to check which areas have been straightened, the triangle faces can be classified into 3 classes based on the number of points snapped in the triangle:

1. red: no vertex is snapped.
2. orange: one vertex is snapped.
3. yellow: two vertices are snapped.
4. green: all three vertices are snapped.

Figure 4.8 shows the classification results of all four test datasets. The green areas are flat while the red areas remain the same as the original data. It can be seen that after global fitting, most parts of the facade are straightened and the windows, balconies are not processed yet.

4.5.3 Mesh segmentation

As shown in Figure 4.8, there are still many points unsnapped. In order to fit small planes on these points, it is neccessary to get segments of these



(a) Test data I **(b)** Test data II

(c) Test data III **(d)** Test data IV

Figure 4.8: Classification of faces after global fitting

points, and input these segmetns to RANSAC. It is because in local fitting, the parameters of RANSAC are relaxed and RANSAC does not consider if the points are actually on the same plane in real world, as long as the requirements are satisfied, the plane can be modelled.

The mesh segmentation is based on region growing in Section 3.7. In order to grow regions efficiently, the topology information needs to be used. A triangle mesh contains many faces and each face has 3 vertices, thus these 3 vertices are connected to each other. From the face information, it is possible to make a structure to store neighboring points for each vertex.

After that, for each vertex, compare the similarity between its adjacent points and the seed point of the segment, if they are similar, they will be put into the same segment. There are two criteria for similarity in this project:

- normal deviation: compare the normal of the seed point and the normal of the adjacent vertices, if the deviation of the normals is not high, they belong to the same segment.

- RGB mahattan distance: if the deviation of the normals is high, compare the RGB mahattan distance between the seed point and adjacent vertex, if the distance is short enough, they belong to the same segment.

(a) Test data I region growing without texture information

(b) Test data I region growing with texture inforamtion

Figure 4.9: Comparison between with and without texture information in region growing

The quality of the data is low, thus the results of the segmentation sometimes can be unsatisfying. Oversegmentation is the main problem in this stage, because the mesh is bumpy so the deviations of the normals are large. However, with the help of texture information, oversegmentation can be reduced. Figure 4.9 shows the results between with and without texture information when region growing. Colors are reused for segments, so two different segments might have the same color. It can be seen that in some parts, oversegmentation problem is reduced, but in general, the mesh is still oversegmented.

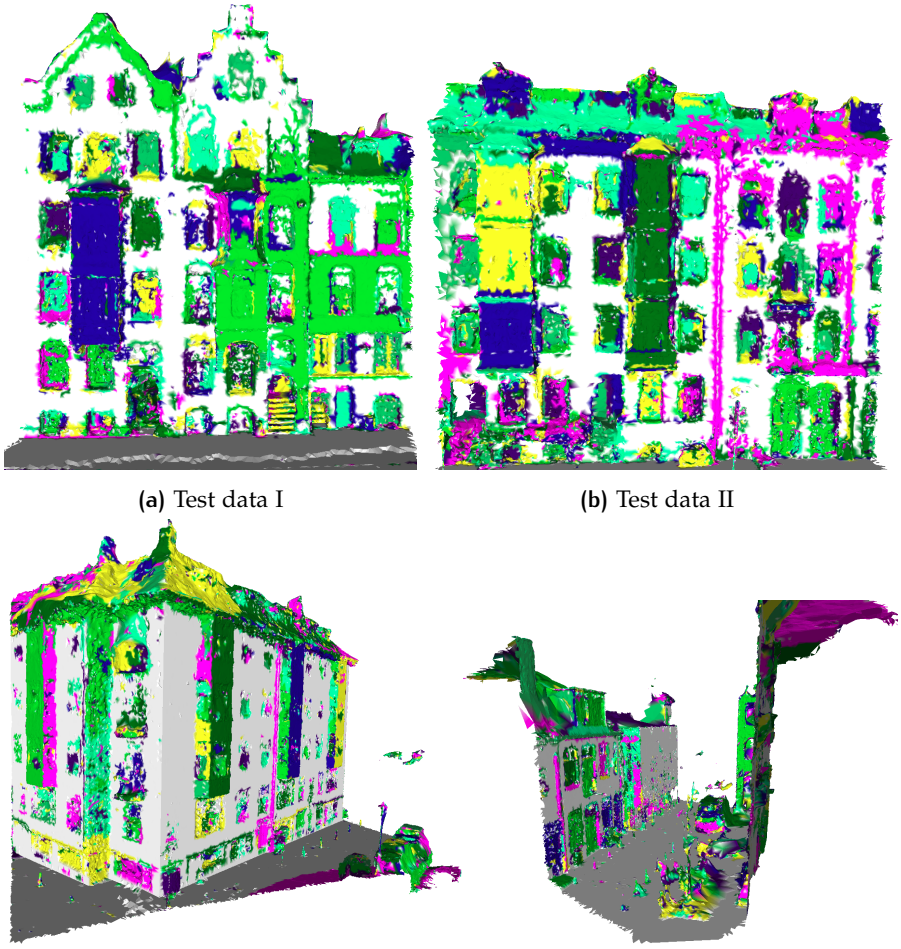
Figure 4.10 shows the segmentation results of all 4 test datasets. White parts in the figure are already straightened in global fitting and the vertices are colored based on segments. It can be seen that due to the quality of the data, oversegmentation cannot be avoid.

4.6 LOCAL FITTING

4.6.1 Small plane fitting

Each segment from mesh segmentation will be input individually to RANSAC to detect smaller planes. Considering the influence of each parameters of RANSAC explained in Section 4.5.1, the paramters should be changed to adjust to local fitting. Generally speaking, the parameters should be relaxed, thus it allows more planes to be detected. The parameters used for local fitting is listed below, and for different datasets there might be some minor adjustments:

- probability $p = 0.05$
- min points $n = 80$
- epsilon $\epsilon = 0.03$
- cluster epsilon $E = 0.5$
- normal threshold $\sigma = 0.9$



(a) Test data I **(b)** Test data II

(c) Test data III **(d)** Test data IV

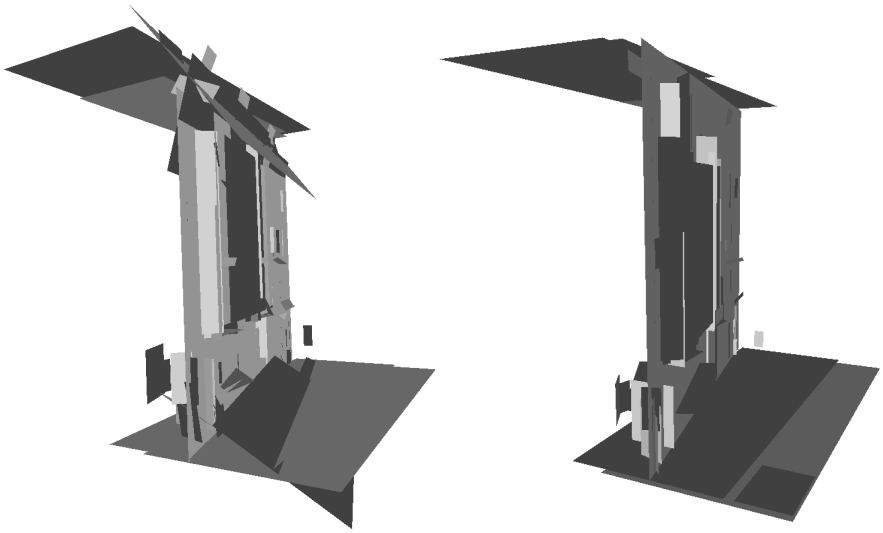
Figure 4.10: Mesh segmentation

In this step, much more planes are detected, and since the parameters for detecting planes are relaxed, some of them might be invalid. Thus there should be more constraints on these planes. As explained in Section 3.5, detected planes in local fitting should be orthogonal or parallel to main planes detected in global fitting.

Figure 4.11 shows the results of small plane fitting with and without plane constraint. In subfigure (a), there are some planes detected which clearly does not exsit in reality for example the oblique plane on the ground. After constrain the planes by the rules, sub figure (b) shows that some invalid small planes are removed thus unreasonable fitting can be controlled by this means.

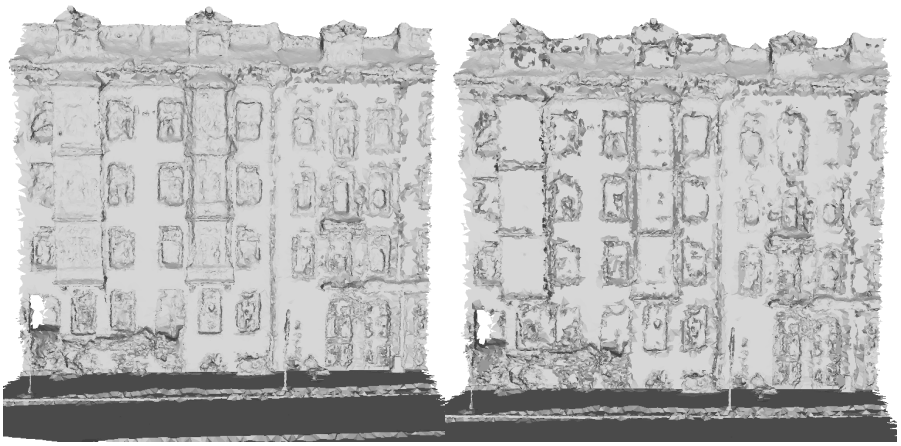
4.6.2 Snapping

After detecting small planes, similar to global fitting, following is snapping . However, different from global fitting, in local fitting there is only one kind of snapping operation, that is snapping inlier points to the planes. Because small planes are less regular and their intersection lines usually do not fit to the intersection lines in reality.



(a) Test data II unconstrained planes **(b)** Test data II constrained planes

Figure 4.11: Comparison between unconstrained and constrained planes in local fitting

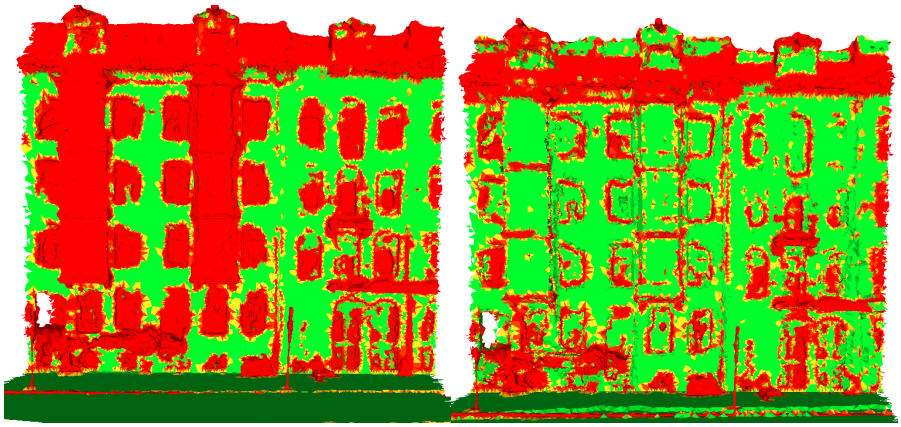


(a) Global fitting **(b)** Local fitting

Figure 4.12: Comparison between result of global fitting and local fitting, test dataset II

Figure 4.12 compares the result of global fitting and local fitting of test dataset II. Compared with the results from global fitting, windows and some other small parts of the model are straightened after local fitting. However, in some parts where the original data is in bad quality, there will be many spikes and the mesh will become less smooth than the original mesh.

The same as the classification rules in global fitting in Section 4.5.2, triangles can be classified into 4 classes. Figure 4.13 shows the classification result. As shown in the figure, compared with the classification result after global fitting, most points are snapped to one plane after local fitting. However there are still many points remain unprocessed. These points mainly are on the edges.



(a) After global fitting **(b)** After local fitting

Figure 4.13: Comparison of the classification after global fitting and local fitting

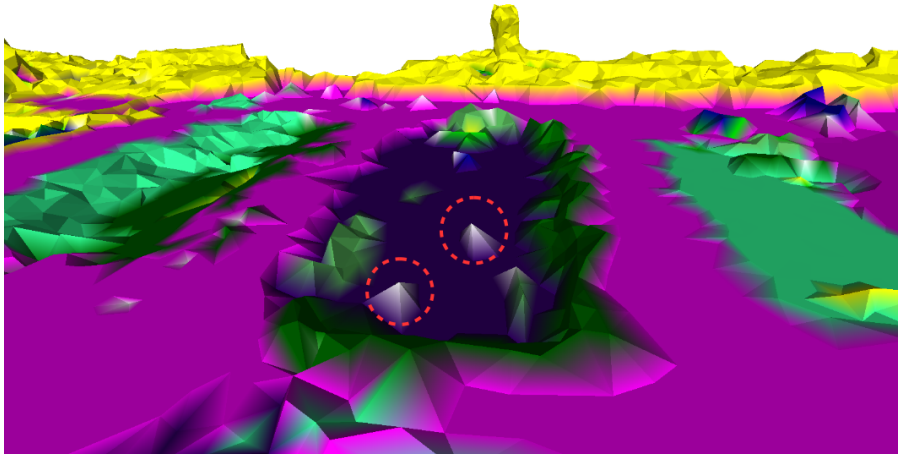


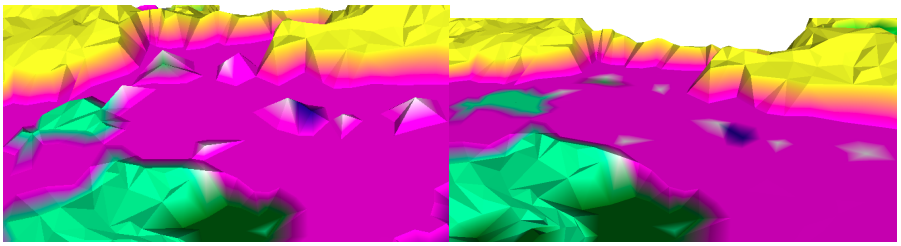
Figure 4.14: Two separate points in one segment are snapped to the same plane

4.7 SEGMENTS SPLIT AND REMOVING SPIKES

In local fitting, most points are snapped to some plane. And based on the plane they are snapped to, all the snapped points can be labelled with plane ID and can be classified into different segments according to these IDs while all the unsnapped points can be classified into the same segment. As explained in Section 3.8, the segments often have several disconnected components. In order to define and remove spikes, it is neccessary to split these disconnected components into different segments.

As shown in Figure 4.14, one color represents one segment. It is clear to see that some disconneted points are snapped to the same plane thus they are in the same segment. For exmaple two white points circled by red dashed lines, clearly they are snapped to the same plane but they are totally separate. Thus they should be split into two segments so that they can be defined as spikes and be removed.

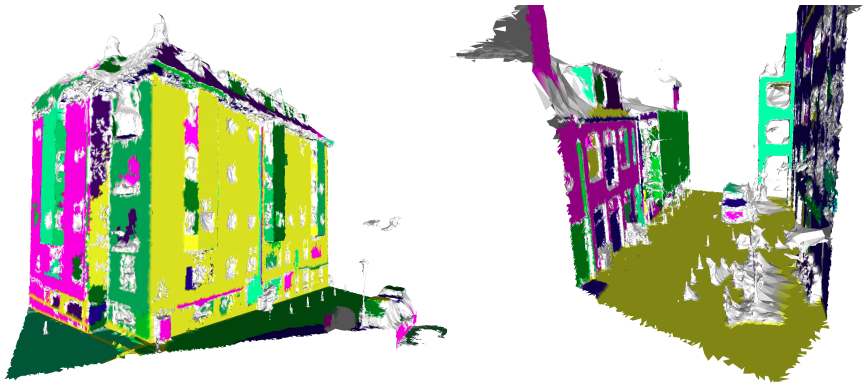
A segment is defined as a spike if it meets the requirements described in Section 3.8. The idea is to remove all the small segments which are surrounded by large segments. It is noticeable that there are many spikes after local fitting, in order to get more smooth surfaces, it is neccessary



(a) Spikes on facade **(b)** After removing spikes

Figure 4.15: Comparison between before and after removing spikes

(a) Test data I **(b)** Test data II



(c) Test data III **(d)** Test data IV

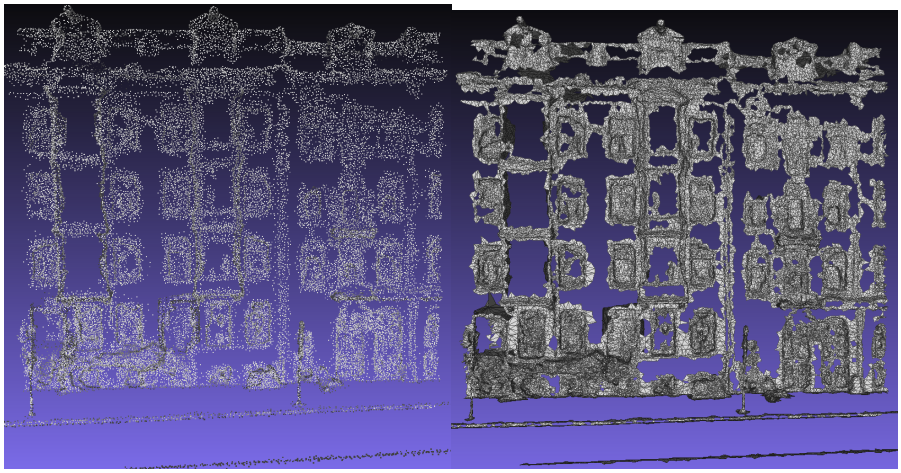
Figure 4.16: Remove spikes and recolor the mesh

to remove them. If a segment is defined as a spike, all the points of the segments will be snapped. Figure 4.15 shows a small part of the facade of test data set II. The colors of the two subfigures are the same, and the purple part is the facade. In subfigure (a), there are several spikes on the facade and in subfigure (b), they are detected and snapped to the facade plane, thus the facade becomes more smooth.

After removing spikes, all the vertices are recolored based on the planes they are snapped to. Figure 4.16 shows the colored final results of 4 test datasets. Colored parts of the mesh are already straightened and the white parts remain unprocessed.

4.8 MESH SIMPLIFICATION

According to the rules of removing verices in Section 3.9, some vertices will be removed. Figure 4.17 (a) shows the result of removing unneccessary vertices. It is clearly to see that many points on the planes are gone, and it is still possible to see the shape of the building including windows and balconies, because important vertices describing the features of the buildings are kept. In Section 3.9, it is also mentioned that some faces can be kept, Figure 4.17 (b) shows the kept faces. There are many empty spaces in these straighten areas, if these areas are triangulated, the mesh will be complete and there will be less faces and vertices so that the data storage will be reduced.



(a) After removing unnecessary points **(b)** Kept faces

Figure 4.17: Mesh simplification

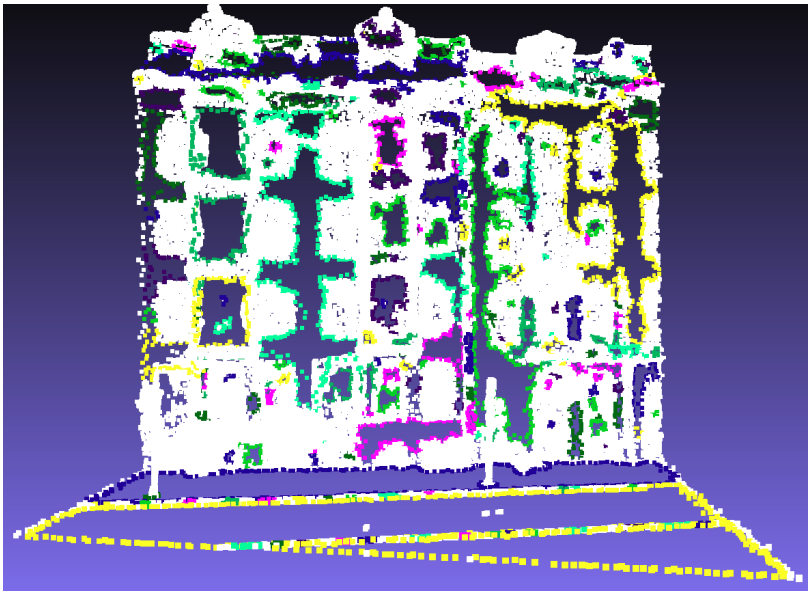
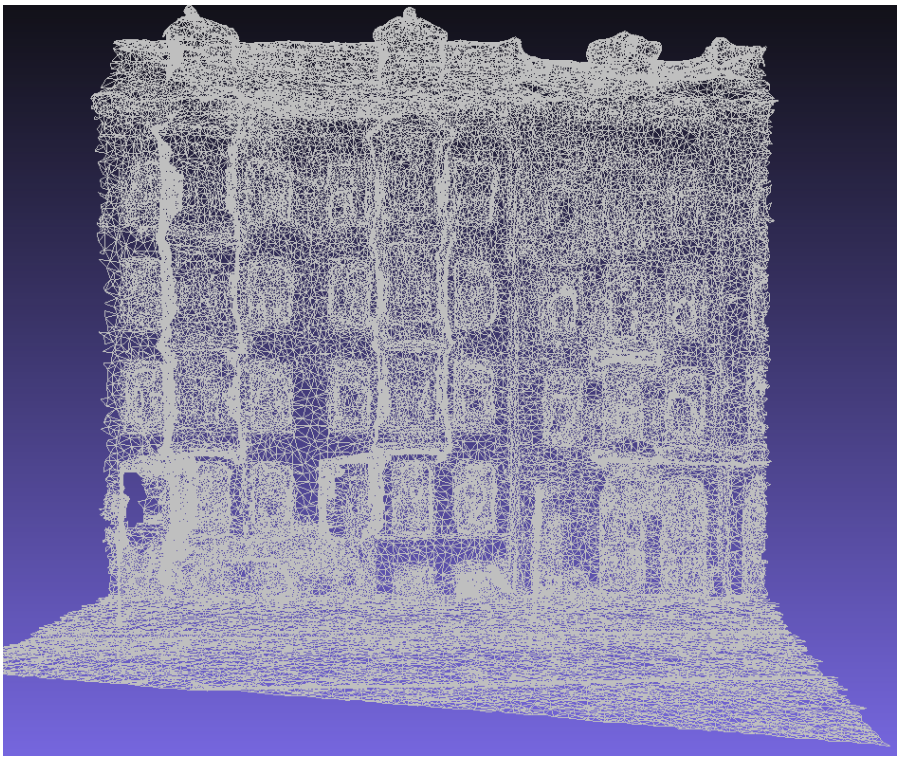
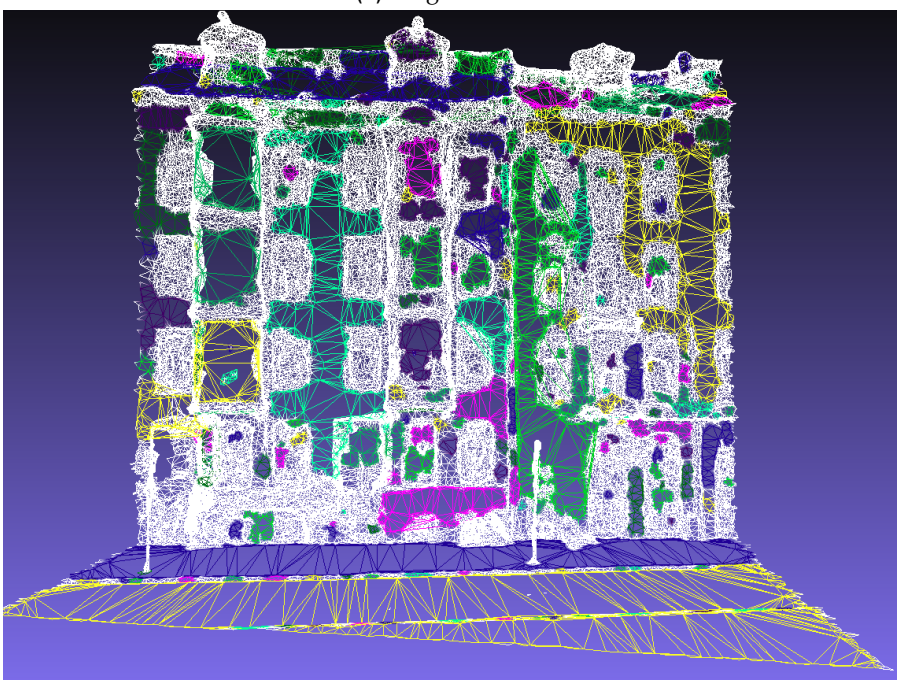


Figure 4.18: Segmented points on the edges of the empty spaces

Figure 4.18 shows the result of segmented points. Points belong to the same segment are colored the same (colors are reused). It can be seen that



(a) Original mesh



(b) Simplified mesh

Figure 4.19: Comparison between original mesh and simplified mesh

the empty spaces are surrounded by points in the same color. These segments of points will be triangulated separately and newly generated triangles will be constrained by the method proposed in Section 3.9. After triangulation, the empty spaces will be filled with new triangles. Figure 4.19 shows the comparison between original mesh and simplified mesh, it

is clearly to see that in planar areas, original small dense triangles are replaced by large sparse triangles. The original mesh has 54962 vertices and 109082 faces while the simplified mesh has 44624 vertices and 88452 faces, since simplified mesh has less vertices and triangles, the data storage can be reduced.

5 ANALYSIS AND COMPARISON

In this chapter, the results will be analysed in details. Section 5.1 will show the good and bad aspects of the results. In some situations, the mesh can be well straightened however in other situations, the method might not work very well. Section 5.2 compares the results from this thesis and Jonsson [2016]. These two thesis have the same aim but use different method and data. In this section, the results of these two methods will be compared.

5.1 RESULT ANALYSIS

Some results are already shown in Chapter 4. The datasets used in this thesis are real world data, so they contain much noise.

It can be seen that global fitting has reliable detected planes. Because these main surfaces contain many points so that the noise has little influence on the detected planes. So in general, main surfaces like facade are well straightened (shown in Figure 5.1).

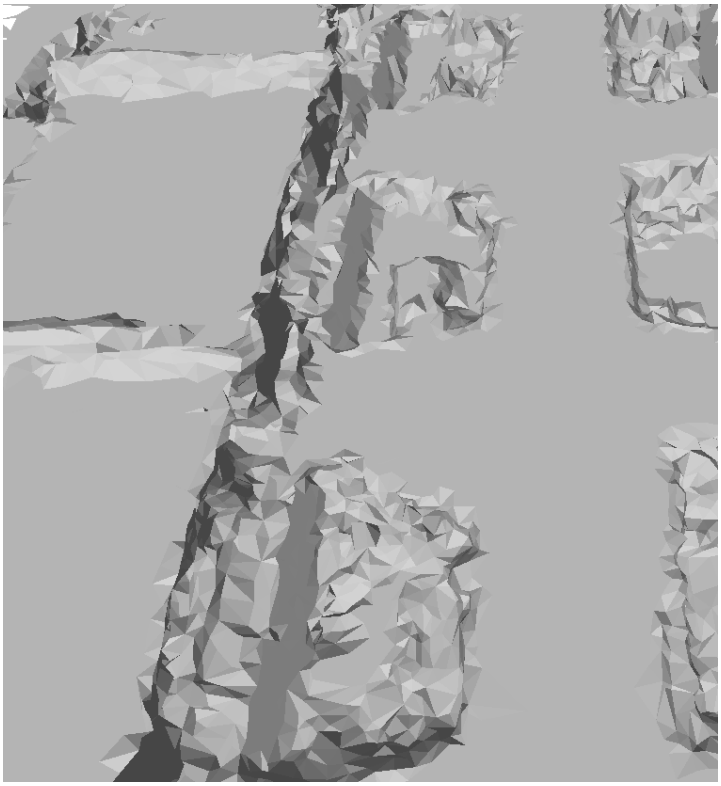


Figure 5.1: Straightened facade

However, the detailed parts like windows etc are much more problematic. In some situations, the small parts can be well straightened, for example in

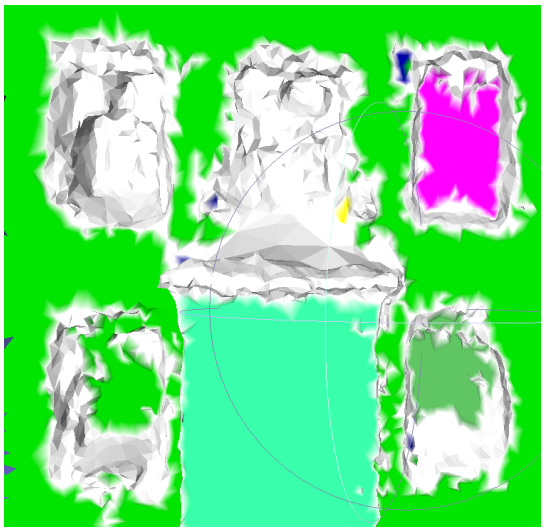
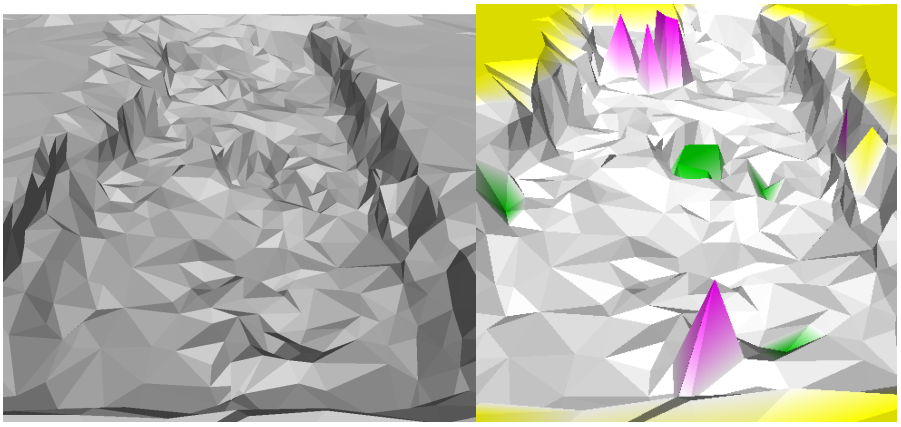


Figure 5.2: Well and badly straightened windows



(a) Original mesh **(b)** Spikes problem

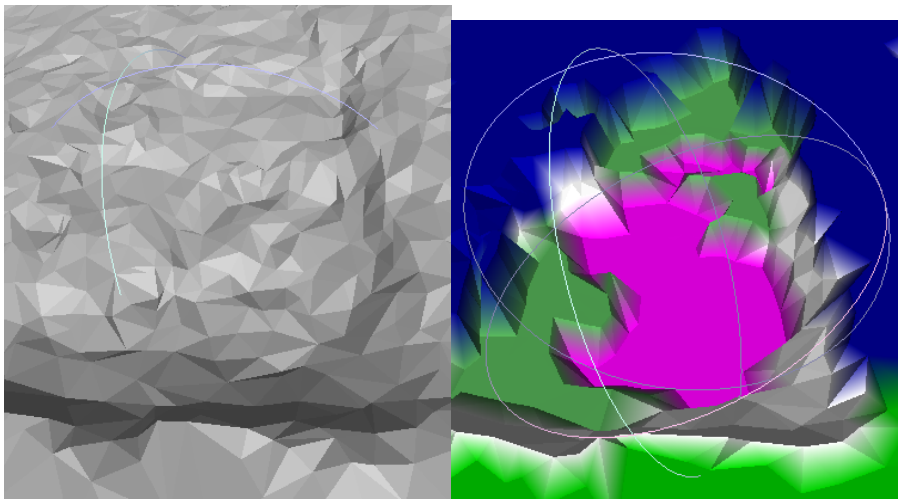
Figure 5.3: Unsolved spike problem caused by unsnapped neighbors

Figure 5.2, the inside of the window in purple are well straightened. But in more situations, these parts cannot be straightened. According to the observations, the inner parts of the windows are often bumpy, which means the normals are not consistent and the deviations of the points are large thus it is difficult to detect planes automatically in these parts. As shown in Figure 5.2, some windows are partly straightened (green part) or not straightened at all (white part).

5.1.1 Spike problem

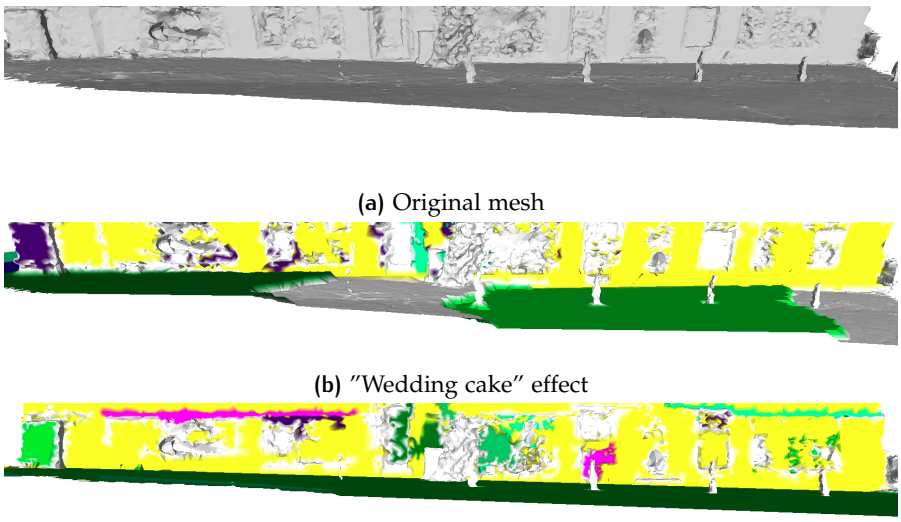
Some common problems will happen in straightening process. The first one is spike problem. Although this method tries to remove as many spikes as possible, it is still an unavoidable problem. For example in Figure 5.3, because the neighboring points of the spikes (in purple and green) are not straightened, the spikes cannot be snapped to any planes according to the method, thus in this situation, the spikes cannot be removed.

Apart from the situations where the neighbors of the spikes are not snapped to any planes, Figure 5.4 shows another situation where the neighbors of the



(a) Original mesh **(b)** Spikes problem

Figure 5.4: Unsolved spike problem caused by mixed neighbors



(a) Original mesh

(b) "Wedding cake" effect

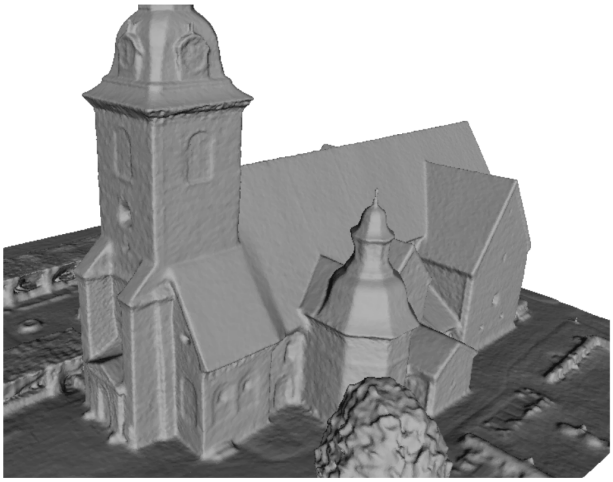
(c) Threshold $\theta = 1°$

Figure 5.5: "Wedding cake" effect caused by inclined surface

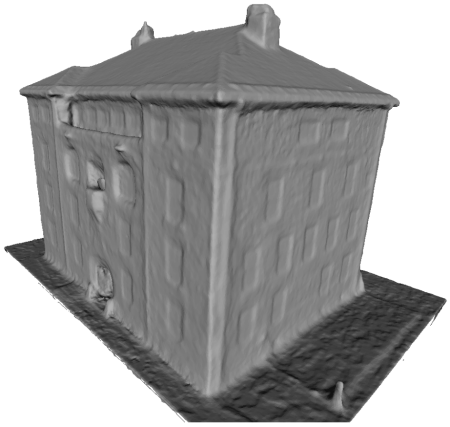
segment are snapped to different planes or unsnapped so that it does not meet the requirement of defining a spike. In this situation, the spikes are not defined as spikes so that they will not be removed.

5.1.2 "Wedding cake" effect

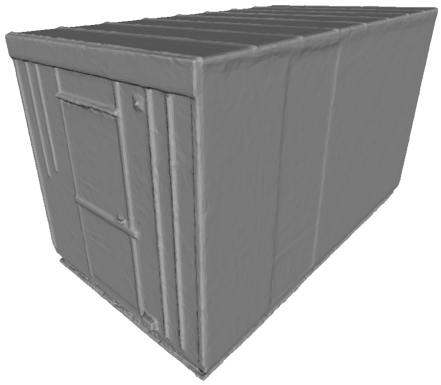
Another problem is "Wedding cake" effect. It often happens in global fitting on large surfaces especially ground surfaces. Because ground surfaces often have many points, and if the ground is not exactly horizontal, these points might be separated into several parts and fitted to different planes. Because these planes are constrained, the detected planes will be made orthogonal if they are near orthogonal. This sometimes leads to stage problem shown in Figure 5.5 (b). In order to solve this problem, the threshold of the angle θ to orthogonality should be small (shown in Figure 5.5 (c)).



(a) Vreta Church data set



(b) Vasallen data set



(c) Container data set

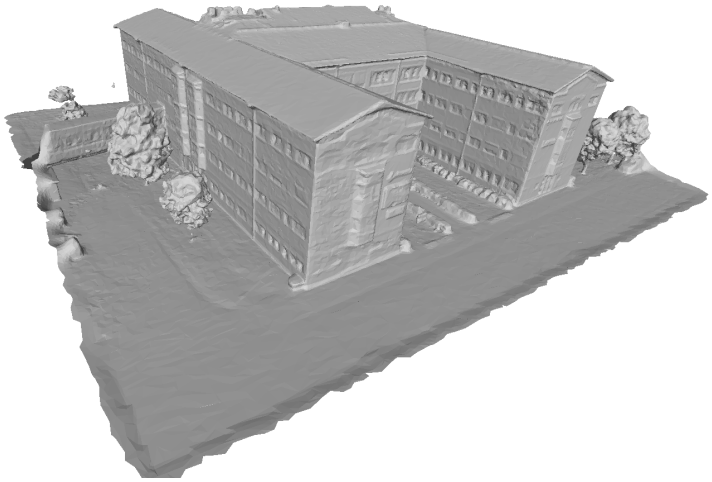
Figure 5.6: Input datasets of Jonsson [2016]

5.2 COMPARISON

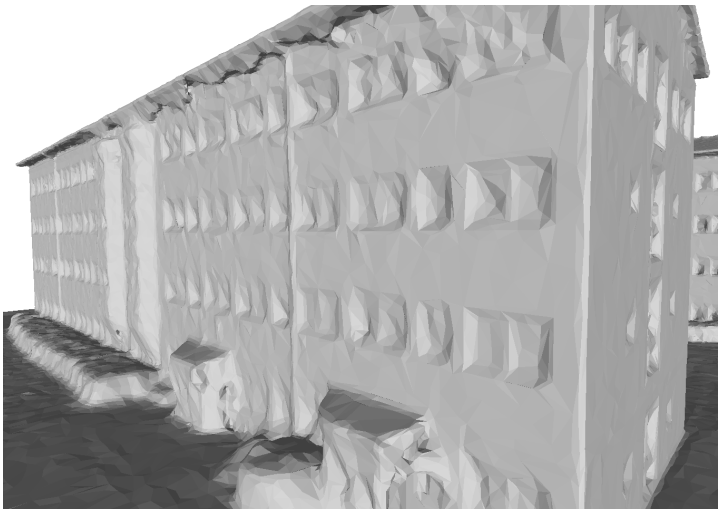
The method of this thesis is compared with the method from Jonsson [2016]. As explained in Chapter 2, Jonsson [2016] used segmentation based method. Generally speaking, the planar areas are detected based on curvature of the vertices then planes will be fitted on these planar areas. This method works very well when the data has good quality so that planar areas are easier to be detected and they are complete in general. However with low-quality data, oversegmentation is a problem so it might has negative influences on the results of straightening.

As shown in Figure 5.6, the three input datasets of Jonsson [2016] have good quality, the meshes are smooth and have little noise. However, the datasets used in this thesis have much lower quality, so it is not possible to apply the same method on these datasets.

In order to compare if the method of the project can yield similar results, besides the data collected by Cyclomedia Technology Inc, the method is also tested on other dataset from Jonsson [2016]. Figure 5.7 shows the test data used for comparison. It can be seen that this dataset has better quality and it contains much less noises. The data contains ground, a building and several trees.



(a) Overview of the data

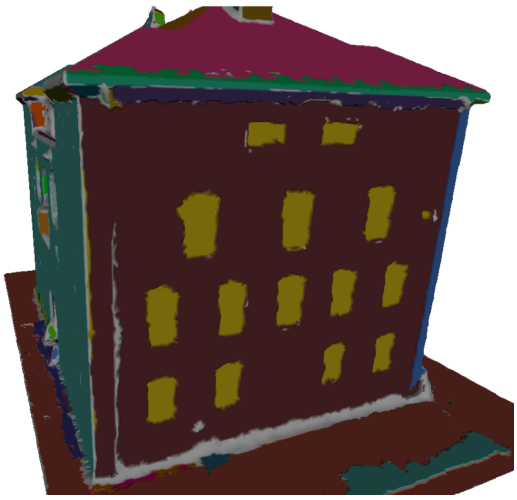


(b) Small part of the data

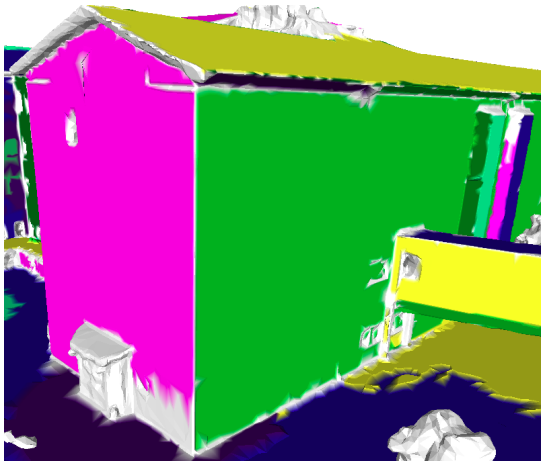
Figure 5.7: Other test dataset

Figure 5.8 compares the results of Jonsson [2016] and the method of this thesis applying on the dataset with similar quality. It seems two methods can yield similar results, but since these two results are not from exactly the same dataset, there are still some differences. A clear difference is that the method of this thesis removes the windows on the facade according to the chosen parameters because as shown in Figure 5.7, the windows in this dataset are very shallow.

Two methods can achieve similar results on main surfaces, however the method of this thesis also refines the edges. A common problem of straightening the mesh is that the edges are always ignored because the points on the edges often have irregular normals. In the method of Jonsson [2016], points on the edges have higher curvatures thus they are often not defined as planar areas. After straightening the mesh, these points will remain the same. However, this thesis provides an inspiration that edges can be refined by snapping points to the edges. Figure 5.9 compares the results between unrefined and refined edges.



(a) Result from Jonsson [2016]



(b) Result from this thesis

Figure 5.8: Comparison of the results

(a) Result from Jonsson [2016]

(b) Result from this thesis

Figure 5.9: Comparison of the results on edges

5.3 RESULT VALIDATION

Most of the result estimation is done by visualization in this thesis. By visually checking whether the planar parts of the mesh are straightened, the result can be estimated as good or bad. Chapter 4 and previous sections in **??** have shown some result estimation visually. Besides that, the output meshes are also checked with validation of the geometry, the following table shows the validation result.

Test data I	Self intersecting faces	Non manifold edges	Non manifold vertices
Original mesh	414	414	179
Straightened mesh	3542	414	179

Table 5.1: Validation result of Test data I

Test data II	Self intersecting faces	Non manifold edges	Non manifold vertices
Original mesh	778	523	190
Straightened mesh	3662	523	190

Table 5.2: Validation result of Test data II

Test data III	Self intersecting faces	Non manifold edges	Non manifold vertices
Original mesh	1477	661	257
Straightened mesh	13186	661	257

Table 5.3: Validation result of Test data III

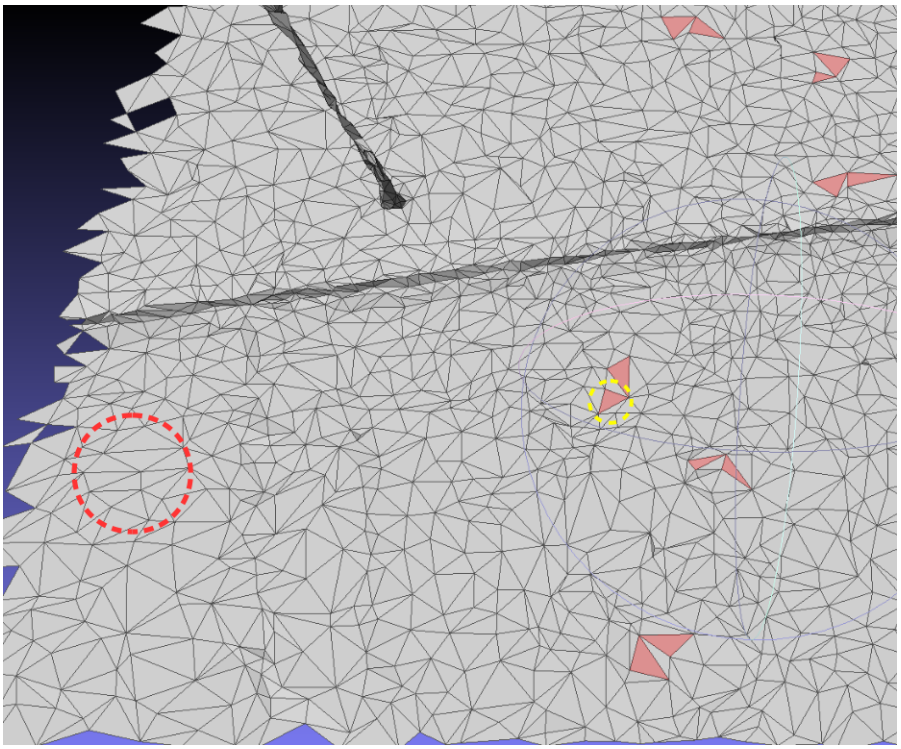
Test data IV	Self intersecting faces	Non manifold edges	Non manifold vertices
Original mesh	1738	1326	451
Straightened mesh	12692	1326	451

Table 5.4: Validation result of Test data IV

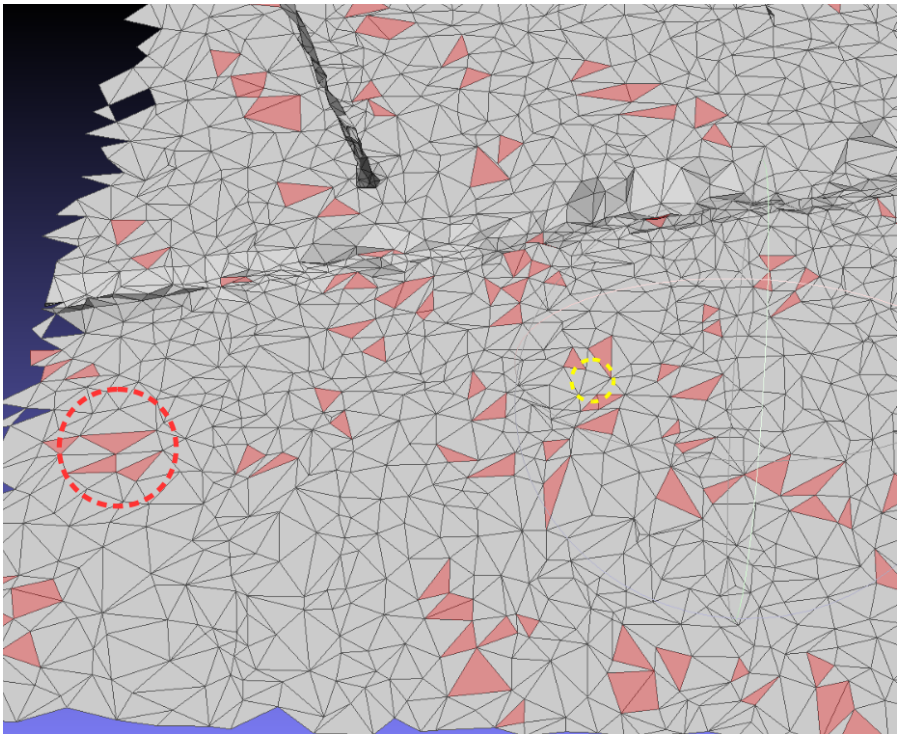
Self intersecting faces, non manifold edges and non manifold vertices are checked for validation. It can be seen from the table that the numbers of self intersecting faces significantly increase, while the number of non manifold edges and non manifold vertices do not change.

Meshlab is used for result validation. And according to the observation to the original mesh and straightened mesh, many faces actually do not intersect but they are defined as self intersecting faces (see Figure 5.10). By comparing Figure 5.10 (a) and (b), it is clearly that the shapes of some faces do not change much but they become self intersecting in the straightened mesh (in red dashed cirle) while some faces become valid instead in the straightened mesh (in yellow dashed cirle).

The reason why there become more unlikely self intersecting faces and how to validate these geometries including self intersecting faces, non manifold edges and non manifold vertices, should be investigated in future work.



(a) Self intersecting faces in orignal mesh



(b) Self intersecting faces in straightened mesh

Figure 5.10: Self intersecting faces

6 CONCLUSION AND RECOMMENDATIONS

The aim of this thesis is to design a method which can straighten multi view stereo meshes. Since the data is real world data and it contains many noises, the method has to be applicable to low-quality data. First in Section 6.1, reseach questions defined in Section 1.3 will be answered. Then there will be some discussions on the methods and the results in Section 6.2. Finally in Section 6.3 some recommendations and future works will be given.

6.1 RESEARCH QUESTIONS

In this section, the research questions posed in Section 6.1 will be answered.

- *"Can RANSAC algorithm based method yield similar or better result than existing approaches for straightening multi view stereo mesh"*.

  According to the comparison in Chapter 5, the method of this thesis yields similar result and it is capable of dealing with data with low quality. According to the experiment, plane fitting method like RANSAC is capable of detecting planes in the datasets with a lot of noises. Compared with segmentation based methods, using plane fitting method can be more efficient and robust. The method of this thesis actually combines both plane fitting and segmentation in global fitting and local fitting respectively. The reason is that in global fitting, main planes are the aims and it is not neccessary to segment them because it is easy to detect these planes directly. However, in local fitting, because the parameters of the plane fitting method are relaxed, it is neccessary to get planar segments first.

  Besides the main research question, the sub research questions can also be answered:

- *What methods are currently used? What are the advantages and disadvantages?*

  As discussed in Chapter 2, most methods resort to detecting planar areas first. Then the planes are fitted on these planar areas. These methods provide stable result because the points in these planar areas are highly spatial related and share similar features, which means the planes fitted on these planar areas are very likely to exist in reality. However, these methods require good segmentation result, so if the data have high quality, these methods will perform well, while when the data contain too much noise, it is difficult to get decent result.

- *How can some plane constraints be used for straightening meshes?*

  Plane constraints can be used to improve the quality of detected planes. Sometimes detected planes might have small errors so they are nearly but not exactly orthogonal, coplanar etc. Plane constraints can help to correct these small deviations. However, plane constraints should

be used carefully otherwise they will have negative influences on the detected planes.

- *How can geometry/topology/texture information be used?*

  Geometry is the basic information of MVS mesh. Plane fitting is based on the geometry of the points.

  Topology information is useful in mesh segmentation. Since MVS mesh already provides topology information, which means the incidents of each points are known. Thus region growing can be based on this topology information. Besides, it is also used for calculating normal for each vertex from incident faces. Moreover, in mesh simplification, the topology information is used as well. By comparing if a vertex and its incidents are snapped to the same plane, it is decidable whether a vertex should be kept or removed. Moreover, topology information is important for edge based region growing in mesh simplification, because the edge structure stores incident edges of the edge.

  Texture inforamtion is additional information in MVS mesh, this project also tests how texture information can be involved. Due to low quality of the data, oversegmentation is a common problem. According to the experiments, texture information can reduce this problem in region growing, because it is expected that same segment should have similar color, region growing will not stop easily because of the large deviations of normals.

- *Is it feasible to simplify the straightened meshes regarding data storage and attach textures to simplified meshes?*

  According to the final results, MVS mesh is straightened and many unnecessary details are removed at the same time. Thus the representation of the model is simplified which is an advantage of 3D city modeling because in 3D models, some details are not important and useful. After MVS mesh is straightened, the project also tests whether the mesh can be simplified with data storage. Since many vertices are snapped to the same planes, it is not necessary to keep all the triangles, so many triangles are removed after simplification. Less vertices and faces means the straightend meshes are simplified regarding data storage.

  Because after removing vertices, the mesh is retriangulated, the indices of the vertices and the number of triangles are changed. The texture information is highly related to triangles and indices, thus it is tricky to attach textures back to simplified meshes. In order to attach texture back, it is necessary to find the links between original triangles and new triangles and create new texture image for new UV mapping.

6.2 DISCUSSION

In this section, the choice of the methods and some shortcomings of the methods will be discussed.

As explained before, a lot of research prefers segmentation based method. They use different approaches to get planar areas based on normal, curvature etc. Then planes are fitted on these planar areas. Segmentation based

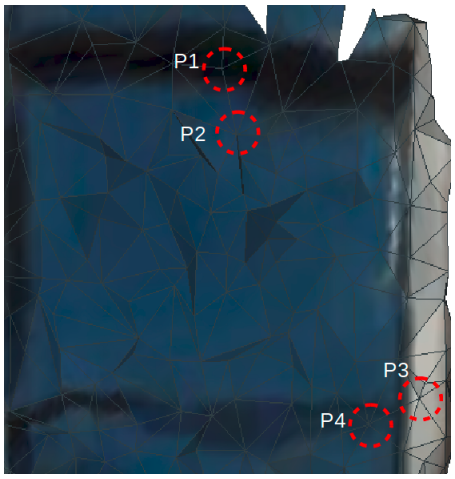


Figure 6.1: Problem in texture enrichment

methods can achieve good results when the data contains a few noises. Instead of plane fitting method, this thesis adopts plane fitting method directly in global fitting without segmentation, because it is relatively easier to detect these main planes so it is not neccessary to segment the mesh first. Besides, the data of this thesis uses has low quality, it is difficult to achieve decent segmentation results. RANSAC has strong ability to detect planes with many outliers, so it is suitable for the datasets of this project.

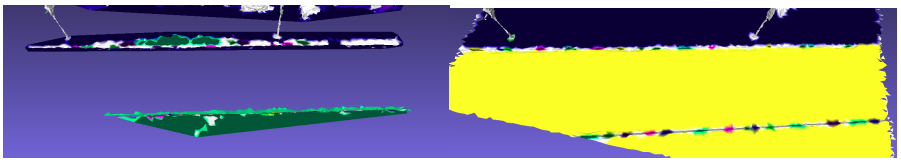
In local fitting, this thesis also adopts segmentation method, because compare with global fitting, local fitting is more tricky. RANSAC does not consider if points are spatially related, and in fact, these details to be straighten in local fitting are always separate, so they should be segmented first. In mesh segmentation, oversegmentation and undersegmentation are two main problems. Between these two problems, according to the experiment, oversegmentation is preferable, because if the mesh is oversegmented, plane fitting method will ignore many non planar areas, these parts will remain the same as the original mesh. However, if the mesh is undersegmented, plane fitting method will fit some planes which do not exist in reality. If the points are snapped to these planes, the quality of the mesh will decrease.

Texture information is enriched to each vertex in this thesis. It is used in region growing in this project. And because the data has relatively low quality, normal based region growing often leads to oversegmentation. Including texture information in segmentation can reduce this problem. However, texture information on each vertex is limited, much texture information is lost because vertices are discret but texture is continous and stored in face unit. Because of this, adjacent vertices sometimes have totally different colors. Figure 6.1 shows part of a window. P_1 and P_2, P_3 and P_4 are connected. However, because there is a distance between adjacent vertices, P_2 and P_4 will get RGB values of blue, while P_1 is black and P_3 is white. For this reason, texture information can only be a supplyment during mesh segmentation.

Regarding of mesh simplification, the theory works with some prerequisites:

- The polygons of empty spaces should be closed.
- The polygons of empty spaces should be simple, which means polygons do not self intersect.

As shown in Figure 6.2 (a), if the shape of the empty space is not a closed polygon, the newly generated triangles cannot recover the original shape of the empty space. So it is necessary to keep the boundary points in order to make the polygon closed, as shown in Figure 6.2 (b).



(a) Non-closed polygon result **(b)** Closed polygon result

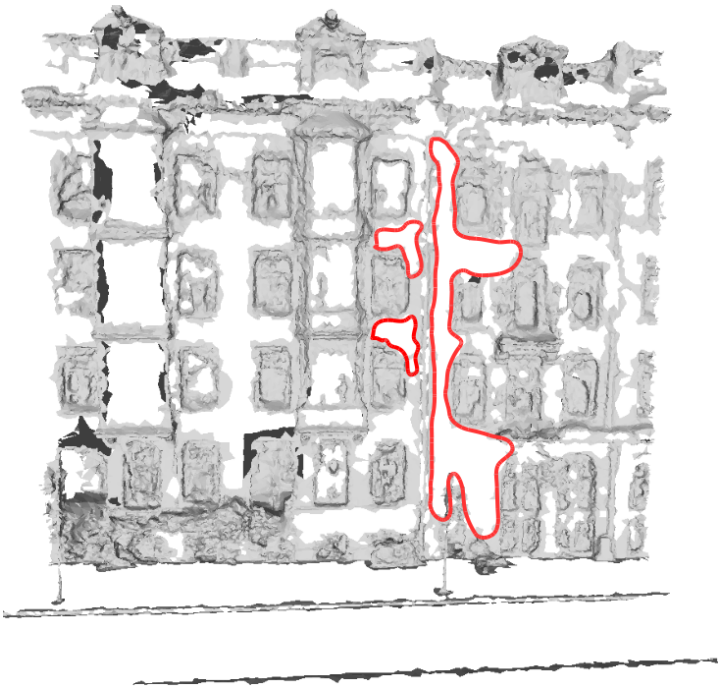
Figure 6.2: Comparison between original mesh and simplified mesh

If the polygon of empty space is not simple polygon, it is not possible to check whether the incenter of the triangle lies inside the polygon or not, thus the generated triangle area will be convex. As shown in Figure 6.3, the original shapes are concave, however, these polygons are not simple because they self-intersect, thus it is not possible to remove triangles outside the shape so that the shapes will become convex (shown in Figure 6.3 (b)). Non-simple polygon problem is caused by inappropriate input order of the vertices of the polygon. It is often caused by adding extra vertices that do not belong to the polygon because the original mesh is not manifold. In a manifold mesh, edges with only one incident face should only have two incident neighbor edges (one previous edge and one following edge). However since the mesh is not manifold, there will be some extra edges (red circle shown in Figure 6.3). If the region grows based on edge with the direction of the arrow from P_1 to P_6 in Figure 6.3 (a), after tracing the vertices from the edges, the polygon will end up with self intersection. In conclusion, in order to get decent result of mesh simplification, the above two problems need to be solved first.

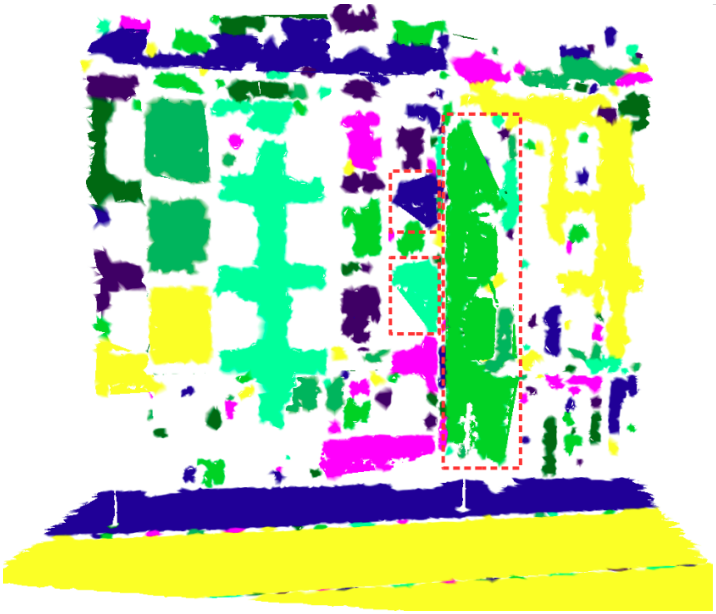
6.3 RECOMMENDATIONS AND FUTURE WORKS

There are several aspects that can be improved in this thesis. Besides, there are some related future work that this thesis can contribute to.

In this thesis, the input of global fitting is the whole dataset. The foundation of this way is that RANSAC is capable of detecting planes among all the points with noises. However, farther points have less spatial relations and RANSAC does not consider spatial relations but only relies on the parameters of detecting a plane. The consequence is that RANSAC might fit planes on points that are far away, which means these planes might not be valid in reality. There are two recommendations for solving this problem. The first one is segmentation. The reason why a lot of research resorts to segmentation before actual operations on the mesh is that segmentation is a way to find highly spatially related points, because points in a segment often have more spatial relations. However there are situations where it is not possible to get decent segmentation results. Then the second way is to clip the whole data into several parts. First, ground points can be found and based on this, separate buildings can be clipped into different datasets. Because the points on the same building have more spatial relations, thus the best way is to clip separate buildings and input them to RANSAC. By this means, the detected planes are more likely to be valid.



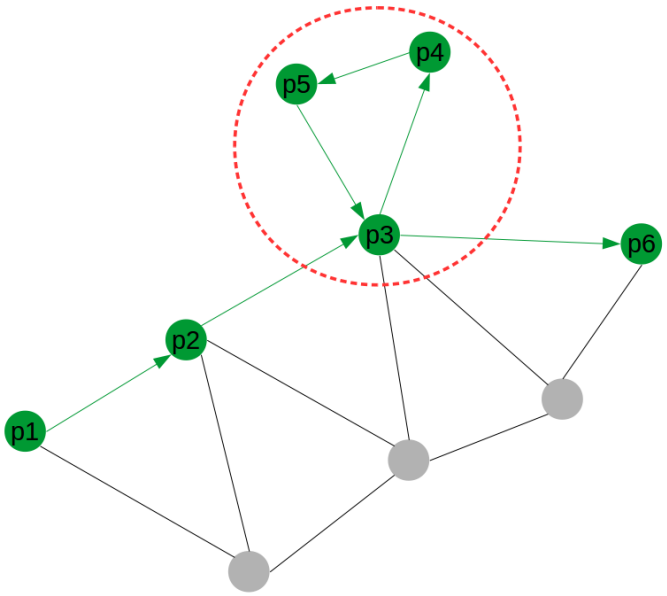
(a) Original concave shapes of the polygon



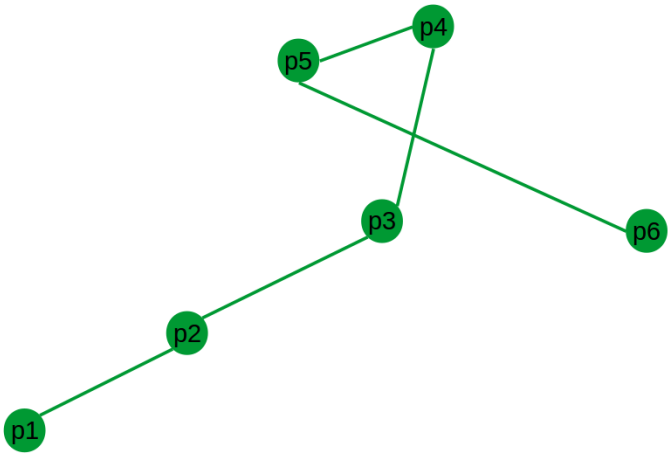
(b) Convex shapes after triangulation

Figure 6.3: Problem caused by non-simple polygon

The way of using texture information can be improved as well by using image processing method. In order to solve the texture problem stated in Section 6.2, there are two recommendations. First the texture images can be blurred using some filters. Thus adjacent vertices are less likely to be influenced by some details in the texture images. Besides, as explained before, much texture information is in faces, so the average color value of the incident faces of the vertex might be a better indicator of the texture of the point.



(a) Edge-based region growing on non-manifold mesh



(b) Self-intersection problem

Figure 6.4: Self-intersection problem caused by non-manifold mesh

Snapping points to edges is a way to refine the edges in this thesis. Edges are defined as the intersection lines of the planes. Planes are infinite so that every two planes will have one infinite intersection line. However in reality, a building consists of several polygons instead of planes, which means some planes detected in this project will not intersect in reality. So it is necessary to transfer detected planes into polygons, and use the intersection line segment of two polygons as edge. Besides, using polygon representation has advantage of defining, generating other format of city models.

Apart from recommendations on the methods used in this MSc project, there are also some related works:

Semantic enrichment is an important topic in city modeling. Verdie et al. [2015] provides an idea of labeling segments as ground, roof, facade etc based on planarity, elevation, horizontality. Besides, texture information enrichment can be used for semantic enrichment as well. For example, veg-

etation can be identified with texture and planarity. If the mesh is enriched with semantic information, detected planes can have their own attribute, which can be helpful for straightening the mesh.

In addition, if the mesh and the detected planes are enriched with semantics, it would be possible to translate MVS mesh to other format of city model such as cityGML. There are several Level Of Details (LODs) in cityGML. Considering the condition of the MVS mesh used in this project, it is better to start with LOD 0. LOD 0 only contains ground and foot prints of the buildings, and since the facade of the buildings and ground plane can be detected, by intersecting these planes, the foot prints of the buildings can be acquired thus LOD 0 can be generated. As described in Chapter 1, the roof of the MVS mesh in this project has bad quality, it is difficult to restore the shape of the roof. However, it is still possible to get elevation values for each roof according to some statistics on roof points. If the elevation of the roof is known, LOD 1 can be generated. LOD 2 is more problematic because the shape of the roof is the characteristics of LOD 2. In order to solve this problem, some other data can be combined with MVS mesh. For example AHN3 data has much more accurate point cloud on roof areas, it might be feasible to replace the roof areas of MVS mesh with AHN3 data so that the quality of the roof can be improved.

BIBLIOGRAPHY

Alharthy, A. and Bethel, J. (2004). Detailed building reconstruction from airborne laser data using a moving surface method. In *20th Congress of International Society for Photogrammetry and Remote Sensing*, pages 213–218.

Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.

Borrmann, D., Elseberg, J., Lingemann, K., and Nüchter, A. (2011). The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):1–13.

Chauve, A.-L., Labatut, P., and Pons, J.-P. (2010). Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1261–1268. IEEE.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association.

Cohen-Steiner, D. and Morvan, J.-M. (2003). Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321. ACM.

Demir, N. and Baltsavias, E. (2012). Automated modeling of 3d building roofs using image and lidar data. In *Proceedings of the XXII Congress of the International Society for Photogrammetry, Remote Sensing, Melbourne, Australia*, volume 25.

Diakité, A. A. and Zlatanova, S. (2016). First experiments with the tango tablet for indoor scanning. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 67–72.

Douros, I. and Buxton, B. F. (2002). Three-dimensional surface curvature estimation using quadric surface patches. *Scanning*.

Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

Elberink, S. O. and Vosselman, G. (2009). Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations. *Sensors*, 9(8):6101–6118.

Fan, H., Meng, L., and Jahnke, M. (2009). Generalization of 3d buildings modelled by citygml. In *Advances in GIScience*, pages 387–405. Springer.

Filin, S. and Pfeifer, N. (2006). Segmentation of airborne laser scanning data using a slope adaptive neighborhood. *ISPRS journal of Photogrammetry and Remote Sensing*, 60(2):71–80.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Furukawa, Y., Hernández, C., et al. (2015). Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148.

Gallup, D., Frahm, J.-M., and Pollefeys, M. (2010). Piecewise planar and non-planar stereo for urban scene reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1418–1425. IEEE.

Girardeau-Montaut, D. (2017). *Cloudcompare (version 2.9) [gpl software]*. [accessed 17-April-2017].

Hoffman, R. and Jain, A. K. (1987). Segmentation and classification of range images. *IEEE transactions on pattern analysis and machine intelligence*, (5):608–620.

Huang, J. and Menq, C.-H. (2001). Automatic data segmentation for geometric feature extraction from unorganized 3-d coordinate points. *IEEE Transactions on Robotics and Automation*, 17(3):268–279.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449.

Jonsson, M. (2016). Make it flat: Detection and correction of planar regions in triangle meshes.

Kada, M. and Wichmann, A. (2013). Feature-driven 3d building modeling using planar halfspaces. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, pages 37–42.

Katz, S., Leifman, G., and Tal, A. (2005). Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658.

Kim, H. S., Choi, H. K., and Lee, K. H. (2009). Feature detection of triangular meshes based on tensor voting theory. *Computer-Aided Design*, 41(1):47–58.

Klasing, K., Althoff, D., Wollherr, D., and Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3206–3211. IEEE.

Koschan, A. (2003). Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE.

Lienhardt, P. (1991). Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-aided design*, 23(1):59–82.

Loriot, S., Tournois, J., and Yaz, I. O. (2017). Polygon mesh processing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.9.1 edition.

Mäntylä, M. (1988). An introduction to solid modeling.

Martinovic, A., Knopp, J., Riemenschneider, H., and Van Gool, L. (2015). 3d all the way: Semantic segmentation of urban scenes from start to end in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4456–4465.

Murdock, K. L. (2008). *3ds Max 2009 bible*, volume 560. John Wiley & Sons.

Oesau, S., Verdie, Y., Jamin, C., and Alliez, P. (2017). Point set shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.9.1 edition.

Orthuber, E. and Avbelj, J. (2015). 3d building reconstruction from lidar point clouds by adaptive dual contouring. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):157.

Pauly, M., Gross, M., and Kobbelt, L. P. (2002). Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pages 163–170. IEEE Computer Society.

PEH, D. (2007). Ro duda, pe hart, and dg stork, pattern classification, new york: John wiley & sons, 2001, pp. xx+ 654, isbn: 0-471-05669-3. *Journal of Classification*, 24(2):305–307.

Remondino, F. and El-Hakim, S. (2006). Image-based 3d modelling: a review. *The Photogrammetric Record*, 21(115):269–291.

Roth, G. and Levine, M. D. (1993). Extracting geometric primitives. *CVGIP: Image Understanding*, 58(1):1–22.

Rouhani, M., Lafarge, F., and Alliez, P. (2017). Semantic segmentation of 3d textured meshes for urban scene analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 123:124–139.

Sampath, A. and Shan, J. (2010). Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds. *IEEE Transactions on geoscience and remote sensing*, 48(3):1554–1567.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library.

Shewchuk, J. R. (1997). Delaunay refinement mesh generation. Technical report, DTIC Document.

Shlafman, S., Tal, A., and Katz, S. (2002). Metamorphosis of polyhedral surfaces using decomposition. In *Computer graphics forum*, volume 21, pages 219–228. Wiley Online Library.

Sun, S. and Salvaggio, C. (2013). Aerial 3d building detection and modeling from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1440–1449.

Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., et al. (2007). Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *Proceedings of the ISPRS Workshop on Laser Scanning*, volume 36, pages 407–412.

Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., et al. (2008). Extended ransac algorithm for automatic detection of building roof planes from lidar data. *The photogrammetric journal of Finland*, 21(1):97–109.

Taubin, G. (1995). Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 902–907. IEEE.

The CGAL Project (2017). *CGAL User and Reference Manual*. CGAL Editorial Board, 4.9.1 edition.

Tschumperlé, D. (2012). The cimg library. In *IPOL 2012 Meeting on Image Processing Libraries*, pages 4–pp.

Tseng, Y.-H. and Hung, H.-C. (2016). Extraction of building boundary lines from airborne lidar point clouds. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 40.

Vadivel, A., Majumdar, A., and Sural, S. (2003). Performance comparison of distance metrics in content-based image retrieval applications. In *International Conference on Information Technology (CIT), Bhubaneswar, India*, pages 159–164.

Valentin, J. P., Sengupta, S., Warrell, J., Shahrokni, A., and Torr, P. H. (2013). Mesh based semantic modelling for indoor and outdoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2067–2074.

VC, H. P. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.

Verdie, Y., Lafarge, F., and Alliez, P. (2015). Lod generation for urban scenes. Technical report, Association for Computing Machinery.

Wang, M. and Tseng, Y.-H. (2010). Automatic segmentation of lidar data into coplanar point clusters using an octree-based split-and-merge algorithm. *Photogrammetric Engineering & Remote Sensing*, 76(4):407–420.

Wang, M. and Tseng, Y.-H. (2011). Incremental segmentation of lidar point clouds with an octree-structured voxel space. *The Photogrammetric Record*, 26(133):32–57.

Wichmann, A. and Kada, M. (2014). 3d building adjustment using planar half-space regularities. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):189.

Wiemann, T., Lingemann, K., and Hertzberg, J. (2016). Optimizing triangle mesh reconstructions of planar environments. *IFAC-PapersOnLine*, 49(15):218–223.

Wiemann, T., Nüchter, A., and Hertzberg, J. (2012). A toolkit for automatic generation of polygonal maps-las vegas reconstruction. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6. VDE.

Woo, H., Kang, E., Wang, S., and Lee, K. H. (2002). A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture*, 42(2):167–178.

Yvinec, M. (2017). 2D triangulation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.9.1 edition.

Zhang, C., Wang, L., and Yang, R. (2010). Semantic segmentation of urban scenes using dense depth maps. In *European Conference on Computer Vision*, pages 708–721. Springer.

Zhao, J., Stoter, J., and Ledoux, H. (2014). A framework for the automatic geometric repair of citygml models. In *Cartography from pole to pole*, pages 187–202. Springer.

Zhao, Z., Ledoux, H., and Stoter, J. (2013). Automatic repair of citygml lod2 buildings using shrink-wrapping. ISPRS.

Zlatanova, S. (2000). *3D GIS for urban development*. International Inst. for Aerospace Survey and Earth Sciences (ITC).

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.