

Knowledge based building facade reconstruction from laser point clouds and images

Knowledge based building facade reconstruction from laser point clouds and images

Shi Pu

Publications on Geodesy 75

NCG Nederlandse Commissie voor Geodesie Netherlands Geodetic Commission

Delft, March 2010

Knowledge based building facade reconstruction from laser point clouds and images

Shi Pu

Publications on Geodesy 75

ISBN: 978 90 6132 319 3

ISSN 0165 1706

Published by: NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission,
Delft, the Netherlands

Printed by: Optima Grafische Communicatie, Optima Graphic Communication, Rotterdam,
the Netherlands

Cover illustration: Shi Pu

NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission

P.O. Box 5030, 2600 GA Delft, the Netherlands

T: +31 (0)15 278 28 19

F: +31 (0)15 278 17 75

E: info@ncg.knaw.nl

W: www.ncg.knaw.nl

The NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission is part of
the Royal Netherlands Academy of Arts and Sciences (KNAW).

Abstract

Various applications demand realistic 3D city models. For urban planning, analyzing in a 3D virtual reality world is much more efficient than imaging the 2D information on maps. For public security, accurate 3D building models are indispensable to make strategies during emergency situations. Navigation systems and virtual tourism also benefit from realistic city models.

Manual creation of city models is undoubtedly a rather time consuming and expensive procedure. On one hand, images are for long the only data source for geometric modelling, while recovering of 3D geometries is not straightforward from 2D images. On the other hand, there are enormous amounts of objects (for example buildings) to be reconstructed, and their structures and shapes show a great variety. There is a lack of automated approaches to understand the building structures captured by data. The rapid development of cities even adds to the cost of manual city model updating. In recent years, laser scanning has been proven a successful technology for reverse engineering. The terrestrial laser point clouds are especially useful for documenting building facades. With the considerable high point density and the explicit 3D coordinates of terrestrial laser point clouds, it is possible to recover both large structures and fine details on building facades. The latest developments of mobile laser scanning technology also make it more cost-effective to take large-scale laser scanning over urban areas.

This PhD research aims at reconstructing photorealistic building facade models from terrestrial laser point clouds and close range images, with a largely automatic process. A knowledge base about building facade structures is established first, where several important building features (wall, door, protrusion, etc.) are defined and described with their geometric properties and spatial relationships. Then constraints for feature extraction are derived from the knowledge base. After a laser point cloud is segmented into planar segments by surface a growing segmentation algorithm, each segment is compared with the feature constraints to determine the most likely feature type for each segment. The feature extraction method works fine for all facade features except for windows, because there are usually insufficient laser points reflected from window glass. Instead, windows are reconstructed from the holes on the wall features. Then outline polygons or B-spline surfaces are fit to all feature segments, and the parts without laser points are hypothesized

according to knowledge. A complete polyhedron model is combined from both fitted and hypothesized outlines.

Since laser data contains no colour information, the building models reconstructed from only laser data contain only geometric information such as vertices and edges. To obtain photorealistic results, textures must be mapped from images to the geometric models. The fusing of laser points and image requires accurate alignment between laser space and image space, which is accomplished after a semi-automated process. Because of the limitations of modelling methods, the geometry model reconstructed from laser points may contain many errors which would cause poor texturing effect. Therefore, significant line features extracted from images are compared with the initial model's edges, and necessary refinements are made to correct the model errors, or at least make the model edges consistent with the image lines. Finally, in the texturing stage, the texture of each model face is selected automatically from multiple images to ensure the optimal visibility. Texture errors caused by occlusions in front of a wall are also removed by analyzing the locations of the wall, the occlusions and the camera position.

Experiments with three data sets show that building reconstruction are considerably accelerated by the presented methods. Our approach is more than 10 times faster than the traditional approach when reconstructing the same buildings, and the models by our approach contain more fine details such as doors and windows. The reconstruction of wall facades and roofs are fully automatic, while some manual interactions (48 percent of the total reconstruction time) are still required for editing the fine details. It should also be faster to make global statistics (number of floors, number of entrances, etc.) and modifications (deriving models with a lower level of detail, applying pre-defined textures, etc.) later on to our models, since different model parts have been associated with the semantic labels. While the reconstruction efficiency is improved by our approach, the visualization effects of our models are also comparable to the models by the traditional approach. The future work will focus on improving the knowledge base and developing a fully automated camera parameter estimation procedure. The completeness and adaptability of the knowledge base will be especially important for the further automation of our reconstruction approach.

Samenvatting

Realistische 3D stadsmodellen zijn noodzakelijk voor verschillende maatschappelijke toepassingen. Voor stedelijke ontwikkeling is het analyseren van een 3D virtual reality omgeving vele malen meer efficiënt dan het interpreteren van 2D kaartinformatie. Voor toepassingen in de publieke veiligheid, zijn nauwkeurige 3D gebouwmodellen onmisbaar om de juiste strategie te bepalen voor noodsituaties. Ook navigatiesystemen en virtuele toeristische activiteiten hebben profijt van realistische stadsmodellen.

De handmatige vervaardiging van stadsmodellen is zonder twijfel een arbeidsintensief en kostbare aangelegenheid. Ten eerste zijn lange tijd fotos de enige databron geweest voor geometrische modellering, terwijl het bepalen van 3D geometrische informatie niet eenvoudig is aan de hand van 2D fotos. Ten tweede zijn er grote hoeveelheden objecten (bijvoorbeeld gebouwen) die gereconstrueerd moeten worden, die daarbij ook nog een grote variëteit vertonen qua structuur en vorm. Er is geen automatische methode om uit structuur van gebouwen zoals die in de data wordt vastgelegd, te begrijpen. De snelle ontwikkeling van steden maakt het nog kostbaarder om stadsmodellen handmatig bij te houden. In de afgelopen jaren heeft laser scanning bewezen een succesvolle techniek te zijn om objecten te reconstrueren. Puntwolken vervaardigd uit terrestrische laserscanners zijn vooral geschikt om gebouwgevels vast te leggen. Gebruikmakend van de hoge punt-dichtheid en de 3D coördinaten van de laserpunten, is het mogelijk om zowel de grove structuur als de fijne details van gebouwgevels te herkennen. Recente ontwikkelingen zoals mobiele laserscanning maken het ook mogelijk om grootschalig data in te winnen in stedelijke gebieden.

Dit PhD onderzoek richt zich op het reconstrueren van fotorealistische gebouwgevels uit terrestrische laserscanner puntwolken en close range fotos, op een zo automatisch mogelijke manier. Allereerst wordt een kennisbank aangelegd, waarin verschillende belangrijke gebouwkenmerken (muren, deuren, erkers, etc.) worden gedefinieerd en beschreven aan de hand van geometrische kenmerken en onderlinge ruimtelijke samenhang. Vervolgens worden voorwaarden aan de kenmerkextractie afgeleid uit deze kennisbank. Nadat de laser puntenwolk is gesegmenteerd in vlakke segmenten, wordt elk segment vergeleken met de voorwaarden uit de kennisbank om te bepalen wat het meest waarschijnlijke gebouwkenmerk is voor dat

segment. De kenmerkextractie werkt voor de meeste gebouwkenmerken, maar niet om ramen te herkennen. Dat komt omdat er normaal gesproken te weinig laser punten reflecteren op raamoppervlaktes. Daarom moeten ramen herkend aan het feit dat er zich gaten in de muren bevinden. Vervolgens wordt om elk segment een rand of B-spline berekend. Delen waar geen laser data aanwezig is worden opgevuld aan de hand van aannames uit de gebouwkennis. Deze delen worden samen met de gereconstrueerde objectenranden gecombineerd tot een compleet polyhedronmodel.

Omdat laser data geen kleurinformatie bevat, bestaan de uit laser data gereconstrueerde gebouwmodellen alleen uit simpele draadmodellen. Om fotorealistische modellen te verkrijgen, wordt textuur op het draadmodel geprojecteerd aan de hand van fotos. Voor het samenvoegen van laserpunten en fotos is het van belang dat beide coördinaatsystemen nauwkeurig ten opzichte van elkaar bekend zijn. Dit wordt op een semi-automatische manier bewerkstelligd. Vanwege de beperkingen van de reconstructie methoden, kan het geometrisch model verkregen uit laser puntwolken veel fouten bevatten die vooral zichtbaar worden tijdens de textuurprojectie. Daarom worden duidelijke lijnkenmerken uit de fotos vergeleken met het initiële geometrisch model. Noodzakelijke ingrepen worden verricht om verbetering aan te brengen in het model, of om in elk geval de zijkanten van het model samen te laten vallen met de lijnkenmerken. Tot slot, wordt de textuur van elk gebouwvlak geselecteerd door meerdere fotos te vergelijken. De foto met het beste zicht wordt geselecteerd. Fouten in de textuurprojectie veroorzaakt door objecten die het zicht op een muur blokkeren worden verwijderd door het analyseren van de positie van de muur, de blokerende objecten en de camerapositie.

Experimenten met drie data sets hebben uitgewezen dat reconstructie van gebouwen aanzienlijk wordt versneld door het gebruik van de hier voorgestelde methoden. Onze benadering is meer dan tien keer zo snel als de traditionele aanpak bij het reconstrueren van dezelfde gebouwen, en de modellen van onze aanpak bevatten meer verfijnde details zoals deuren en ramen.

De reconstructie van voorgevels en daken gaat volautomatisch, waarbij enkele handmatige interacties (48 percent van de totale tijd voor reconstructie) nog nodig zijn om de meer verfijnde details te editen. Omdat we verschillende model-delen associëren met semantische labels verwachten we dat het eenvoudiger is om globale statistieken uit te rekenen, zoals aantallen deuren en ingangen, en om aanpassingen te maken (modellen berekenen met een lager detailniveau, het aanbrengen van voorgeprogrammeerde texturen, etc.). De efficiëntie van de reconstructie wordt door ons model verbeterd terwijl de visualisatie effecten van onze modellen te vergelijken is met die van de traditionele aanpak. In de toekomst zal ons werk focussen op het ontwikkelen van een volledig geautomatiseerde camera parameter inschattings-procedure én het verbeteren van de knowledge base. We denken daarbij vooral aan het uitbreiden van de knowledge base en aan het flexibiliseren voor toepassing van onze geautomatiseerde reconstructie aanpak in verschillende situaties.

Acknowledgement

The journey of the PhD study is a unique experience in my life. It is my first time to be so focused on one topic, and meanwhile my first time to achieve so much from one topic. Since childhood I am curious about every subject of science, and have always been eager to challenge fresh domains. After the last four years of PhD study, I have realized that the joy of exploration comes not only from the varieties of challenged topics. Years of concentration on the fantastic laser point cloud, as well as the mingled inspiration, frustration, excitement and anger, renders extraordinary sense of achievement in the end.

During the PhD study I have been supported by many people. Here, I would like to acknowledge their help. First of all, I would like to thank Prof. George Vosselman, my promoter and supervisor. I thank him for giving me the chance to pursue a PhD and for his constant guidance during the PhD study. Furthermore, his rigorous and efficient style has deeply influenced me within and beyond research.

I would like to thank Sander Oude Elberink and Yixiang Tian, my PhD mates. We worked on the similar topics about building reconstruction, started our PhD studies from more or less the same time, and shared the same office for a long period. This PhD journey would not have been so fruitful without their suggestions, and would not have been so joyful without their accompanying. Sander, thank you for your friendliness all these years. Yixiang, thank you for keeping me updated with the other side of the world so that I can still fit in.

I would like to thank my colleagues in the department of Earth Observation Science, for the professional atmosphere they have created here. Some people I would like to mention in particular. Teresa Brefeld, our vigorous secretary, thanks for your long-term assistances and patience. Markus Gerke, Martin Rutzinger and Stefan Heuvel, thanks for your constructive advices and inspirations. Wan Bakx, thanks for your scrupulous contribution to the Nederlands Samenvatting.

I thank my Chinese friends who have shared the life far away from home in the Netherlands. I am grateful for all the happy moments we have spent together. Special thanks to my fellows: Guo Cheng, Kang Zhizhong, Liu Yu, Song Chunlin, Wang Han, Zhao Lingxiao and Zhou Weihua.

I would also like to thank my parents, whose love and support were always there when I needed, even from thousands of miles away.

Lastly and most importantly, Bai Lei, my beloved wife. You have accompanied me along every step of my PhD journey, and encouraged me with your steadfast trust. You joined me, brought us the most beautiful baby in the world, and endowed new meaning to my life. Thank you.

Contents

1	Introduction	1
1.1	State-of-the-art of terrestrial laser scanning	3
1.2	Related works	6
1.2.1	Overview	6
1.2.2	Frueh et al. 2005	8
1.2.3	Cornelis et al. 2008	8
1.2.4	Ripperda 2008	9
1.2.5	Becker 2009	10
1.3	Method overview	11
1.4	Structure of the thesis	12
2	Knowledge engineering and reasoning	15
2.1	Knowledge engineering	16
2.1.1	Assembling the knowledge	16
2.1.2	Decide on a vocabulary	19
2.1.3	Encode general knowledge	20
2.1.4	The hierarchical composition	22
2.2	Reasoning with the knowledge	22
2.2.1	The proof tree	22
2.3	Managing uncertainty	26
2.3.1	Describing the uncertainty	27
2.3.2	Making expected decisions	29
2.4	Concluding remarks	31
3	Feature extraction	33
3.1	Preprocessing	33
3.1.1	Spatial indexing	33
3.1.2	Extracting points of interest	34
3.2	Extraction of geometric features	36
3.2.1	Flat surfaces	36
3.2.2	Curved surfaces	38
3.3	Extraction of semantic features	39
3.3.1	Solid features extraction	40
3.3.2	Hole-based window extraction	42
3.4	Discussion	43

4	Geometric reconstruction	45
4.1	Polygon fitting	45
4.1.1	Least squares fitting	45
4.1.2	Convex polygon and concave polygon fitting	46
4.1.3	Minimum bounding rectangle fitting	48
4.2	B-spline surface fitting	48
4.2.1	The B-spline curve and surface	49
4.2.2	B-spline surface approximation	50
4.3	Hypotheses for parts without laser data	51
4.4	Results and Discussion	51
4.4.1	Flat surfaces	51
4.4.2	Curved surfaces	53
5	Model refinement with imagery	57
5.1	Method overview	58
5.2	Registration	58
5.2.1	Perspective Conversion	59
5.2.2	Spatial Resection	62
5.2.3	Relative Orientation	62
5.3	The model refinement	64
5.3.1	Extraction of Significant Lines from Images	65
5.3.2	Matching Model Edges with Image Lines	65
5.3.3	Refinement Strategy	67
5.4	Test cases	67
5.4.1	The restaurant house	67
5.4.2	The town hall	68
5.4.3	The wall with high windows	69
5.4.4	Summary	71
5.5	Conclusions and outlook	72
6	Texture mapping	73
6.1	Selecting texture images	74
6.1.1	Optimal image selection	75
6.1.2	Occlusion removal	75
6.2	Calculating texture coordinates	77
6.3	Results and discussion	78
6.3.1	The three joined houses	78
6.3.2	The house with a balcony	79
6.3.3	The curved walls	80
6.3.4	Discussion	80
7	Method evaluation	83
7.1	The reconstruction approaches	83
7.1.1	Our approach	84
7.1.2	The traditional approach	85
7.2	The Vlaardingen case	86
7.3	The Enschede case	94

7.4	The Esslingen case	95
7.5	Conclusions	96
8	Conclusions and recommendations	103
8.1	Conclusions	103
8.2	Recommendations	104
	Bibliography	106
	List of publications	111
	List of Figures	113
	List of Tables	115

Introduction

Realistic 3D city models are required for many purposes such as urban planning and safety analysis. Originally the access of citizens to urban development plans is generally limited to 2D design plans, which may be difficult to interpret. The availability of 3D models of the current urban environment, as well as new urban objects and their alternatives, would increase this involvement remarkably. 3D city models can play an important role in security analysis as well. A realistic environment is essential for making good security analysis and training, particularly where physical security (security of infrastructure) and social security (livability, security feeling) are concerned. Navigation systems and virtual tourism also benefit from realistic city models. For example, the readability of navigation programs can be greatly increased if the building facade models are present along with the roads.

Due to the huge number of urban objects in a city and variety of shapes, manual reconstruction of an urban area, especially the buildings, is a rather time-consuming and expensive procedure. First of all, manual measurement of data is time consuming and difficult to expedite. For example, obtaining 3D distance from images relies on manual selection and orientation of the images, which involves a considerable amount of user effort. Secondly, manual creation of building models is also a slow procedure. A common manual reconstruction starts with generating 2D building outlines from 2D GIS data, then elevating 2D outlines with a certain height to a 3D rough model, then adding 3D structures by observing and measuring in images and modelling with 3D modelling packages such as 3DStudio Max or Sketchup. Finally, the 3D model is textured by manually selecting certain parts from images. The speed of these steps is greatly dependent on the building complexity and the operator's proficiency. Finally, manual updating of city models includes manual change detection and remodelling, which is again time-consuming. The rapid development of cities also adds to the cost of manually updating city models.

Therefore, a number of attempts have been dedicated in recent years to automate the reconstruction of building. At this moment, there are two types of sensor technology which can be applied to document building geometry:

-
- **Optical data** (such as image and video) have served as important data sources for a wide variety of applications for decades.
 - **Laser scanning** is a new sampling technology which is often used in reverse engineering for constructing digital and three dimensional models.

Although image acquisition and laser scanning are becoming more accurate and automated, the automated processing of the resulting datasets is at a very early research stage (Brenner, 2005). For example, laser point clouds are mainly used to generate a Triangulated Irregular Network (TIN), although the data storage and visualization of TINs are very heavy. On the application side, the demands for virtual 3D city models has becoming more urgent. One thing that is especially needed by a building's virtual reality model is the details about its facade. Recent research (Brenner, 2005; Rabbani et al., 2007) show that terrestrial laser points and close range images are valuable pieces of data for detailed 3D model reconstruction. On one hand, the details on building facades are accurately recorded by terrestrial laser scanners and digital cameras (see Figure 1.1). On the other hand, buildings are man-made objects and have regular shapes. Thus, with intelligent generalization of building shapes, efficient integration of multiple data sources and adaptive interpretation of sensor data, automated reconstruction of building facade models should be feasible.

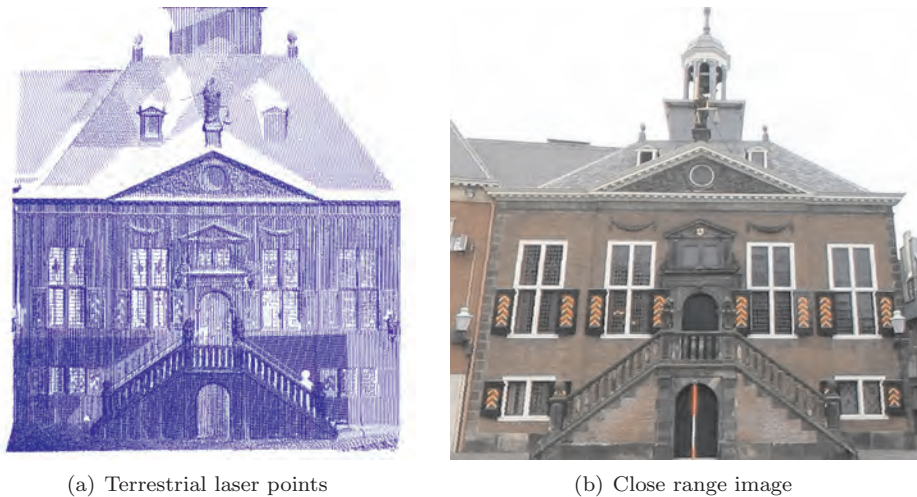


Figure 1.1: Documenting facade details by terrestrial laser scanning and close range imagery

In this research, we aim at developing a framework which automatically reconstructs photorealistic building facade models by fusing terrestrial laser scanning data and close range images. This chapter provides the background knowledge of this research and then sketches out our research methodologies. A state-of-the-art overview of the terrestrial laser scanning technology as well as its application for building reconstruction is provided in Section 1.1. Then a literature survey over

the existing approaches is given in Section 1.2. Our proposed methodology and the structure of the thesis are then outlined in the last section.

1.1 State-of-the-art of terrestrial laser scanning

As a renovative survey technology, the terrestrial laser scanning (TLS) systems have been rapidly developed in recent years. There have been more laser scanners available on the market, and their scanning abilities (angle resolution, scanning range, speed, etc.) are also very well developed (Lemmens, 2001, 2004, 2007, 2009). This section provides an overview of the latest TLS technologies.

Located near the ground surface, the TLS systems are able to deflect mass laser beams over a scene and receive the reflectance. Besides the coordinate information, the reflectance strengths of laser beams (sometimes referred as the intensity value) are also collected as an additional information. Three categories of TLS systems can be distinguished by the principle of the distance measurement:

- In the **pulse systems**, the complete flight distance of a laser pulse is calculated by multiplying the time-of-flight and the light speed. The distance between the laser scanner and any reflecting surface is half of the complete flight distance. Together with the deflecting angle of the laser beam (relative to the laser emitter), the 3D coordinate of the reflecting position can be calculated. The pulse systems are advanced in term of scanning range, on the order of kilometers, while are less accurate than the other two systems. For example, the laser range of the Riegl LMS-Z620 (see Figure 1.2(a)) is from 2m to 2000m, and the range accuracy at 50m distance is 10mm.
- The **phase-shift systems** differ from the pulse systems that the flight distance is measured from the change in pulse phase rather than the time-of-flight. Their ranges are shorter than the pulse systems, while the accuracies are higher. For example, the Faro Photon 120 (see Figure 1.2(b)) can only reach a maximum of 120 meter, and the range accuracy is 2 mm at 50m distance. The phase-shift systems also have a very high measurement rate (points per seconds emitted), e.g. Leica HDS6100 emits 500,000 points per seconds (Lemmens, 2009).
- An **optical triangulation** scanner requires an additional camera (with known relative position and orientation) looking for the laser beams' reflecting positions. For each laser reflectance, a triangulation is formed by the camera, the laser emitter, and the reflecting position. The distance between the camera and the laser emitter and the angle of the laser emitter corner are both known. The angle of the camera corner can be derived from the laser dots' position inside the cameras field of view. The shape and size of the triangle can be fully determined from the known information (one side and two corners), and hence the reflecting position can be determined. The

optical triangulation scanners have a limited range of some meters, but their accuracy is very high: on the order of tens of micrometers.

The phase-shift systems are well suited for high precision and detailed measurement of very closeby scenes, such as industrial objects (e.g. cars), heritage sites and crime scenes, while the pulse systems are well suited for 3D reconstruction of scenes farther away from the scanner, for example to create 3D models of plants, or even entire cities (Lemmens, 2009). The optical triangulation systems are more specialized for reverse engineering of small objects such as industrial products and artworks. It is quite common for most mainstream TLS scanners to mount a digital cameras so that images can be captured concurrently. With accurate calibration between the scanner and the camera, each laser point can be associated to a RGB value from the optical data.



(a) Riegl LMS_Z620
(pulse system) (Riegl,
2009)



(b) Faro Photon 120
(phase-shift system)
(Faro, 2009)

Figure 1.2: Two mainstream terrestrial laser scanners

Nearly all TLS systems work with a static scanning style, such that each single scan can only be taken from a fixed location. In order to cover a complete scene, operators have to move the scanner to the next scan position and make another scanning. The number of necessary single scans is dependent on the scene complexity. For example, complete scanning of a church might need several scan positions, while a block building might only need one scan position. Such a “stop-go” scanning style is very slow, not only because of the transportation between scan positions, but mainly due to the tedious registration steps required to transform points from different single scans to a unique coordinate system. The post-acquisition softwares usually provide the registration functionalities which are based on the Iterative Closest Points (ICP) method, but the tie pairs for initial transformation estima-

tion still need be manually selected, or a group of reflectance targets need to be set up before the single scans. There have been several methods reported in recent years towards fully automated registration (Rabbani et al., 2006; Brenner et al., 2008).

The latest mobile laser scanning (MLS) systems make significant upgrade to the traditional TLS systems (Optech, 2009; 3DLaserMapping, 2009). In MLS systems, devices like laser scanners, Inertial Measurement Unit (IMU), GPS receivers and cameras (see Figure 1.3) are mounted on the top of vehicles, and the laser scanning and image acquisition takes place during driving. The simultaneous collection of IMU and GPS data enables calculation of the laser scanner's position and pose along time steps with very small intervals, which is similar to the airborne laser scanning (ALS). Since any laser point at a local time step with a local coordinate can be transformed to a global coordinate system according to the scanner's position and pose at that time step, the automated registration of all laser points is achieved. There have been a number of MLS systems available on market at this time, for example Lynx (Optech), Streetmapper 360 (3D Laser Mapping and IGI mbH) and VMX-250 (Riegl).

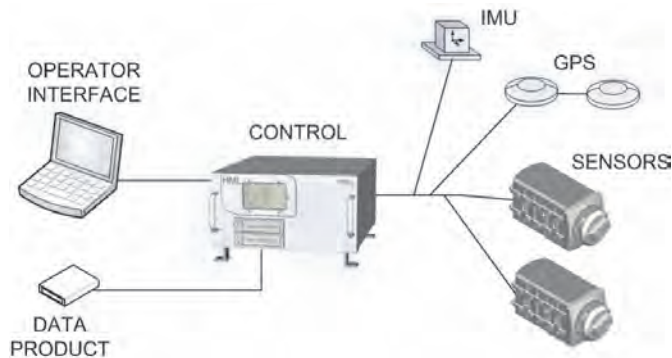


Figure 1.3: A standard design structure of MLS system (Lynx Optech (2009))

The density of the MLS point cloud is more than sufficient for purposes like building reconstruction, deformation monitoring and road inventory, but still not as high as the point clouds by static TLS systems. A list of densities of MLS data acquired by Optech Lynx under different conditions is provided in Table 1.1. Because the similarities to ALS, the errors of MLS also come from same sources: system calibration, range errors, GPS trajectory and GPS/IMU sensor calibration. The range errors of current MLS laser systems are at millimeter level. This is much lower than ALS, because the MLS ranges are much shorter so that angular errors show less influence. In contrast, while ALS has perfect reception of GPS signals, the GPS outages caused by high vegetation and buildings are the dominant error source in MLS sensor orientation (Lindenberger, 2009). The absolute accuracies of current MLS systems are at centimeter level and depend very much on the GPS condition (Optech, 2009; 3DLaserMapping, 2009). When the GPS signal is lost,

the vehicle has to be stopped to regain signal and reset the IMU. Such a drawback has yet to be improved by the manufactures.

Vehicle speed	30 km/h	50 km/h	80 km/h
Point density on road			
- along center line	1519	916	570
- 2m from center line	930	557	350
- 5m from center line	222	143	83
Point density on wall in 10m height above road			
- 5m away from center line	270	162	100
- 10m away from center line	222	134	83
- 20m away from center line	132	79	49

Table 1.1: Density (pts/m²) of MLS data acquired by Optech Lynx (Lindenberger, 2009).

1.2 Related works

A number of related publications and softwares have been found from a survey on building reconstruction topic. This section gives an overview of the existing approaches, and then emphasizes the works which are particularly related to building facade reconstruction.

1.2.1 Overview

The existing building reconstruction approaches can be categorized with respect to the input data, the reconstruction principle, and the reconstructed model type.

Commonly used input data for building reconstruction include airborne images (Suveg and Vosselman, 2004; Taillandier and Deriche, 2004), close-range images (Dick et al., 2001; Zisserman et al., 2001; Schindler and Bauer, 2003), airborne laser scanning data (Maas and Vosselman, 1999; Haala and Brenner, 1999; Oude Elberink and Vosselman, 2009), terrestrial laser scanning data (Frueh et al., 2005; Pu and Vosselman, 2009; Becker, 2009) and video (Pollefeys et al., 2008; Tian et al., 2009). **Airborne** data usually covers large urban areas. Structures on roofs and grounds can be clearly recorded with airborne systems. In contrast, **close-range** systems or terrestrial systems record information from building facades, and are more suitable for applications demanding facade details. Modern photography makes the recording of urban environments rather fast and cheap, and the **images** have the advantage that they are easily understandable by humans. However, the automation and reliability of image based reconstruction approaches are still surprisingly low, even after many years of research. The difficulties in automated image interpretation mainly comes from the recovering of 3D geometry from 2D images. Image is a two dimensional projection of a three dimensional world, and the depths of image pixels are not available. The 3D coordinates can

only be recovered by matching and intersecting corresponding 2D geometries from multiple images. The accuracies of these steps are very dependent on the image quality and perspective (or occlusion) condition. In a word, the imagery based reconstruction approach can be seen as a process of “3D to 2D” (image acquisition) plus “2D back to 3D” (image intersection), where a number of difficult problems exist throughout and can hardly be solved. In **laser scanning**, the scene is documented in terms of laser points with explicit 3D coordinates, and the 3D geometry features can be directly extracted from the 3D point clouds. The complex dimension transformation is naturally avoided. In addition, the wide field of view of laser scanners as well as the high resolution makes it possible to reconstruct not only the large scale structures but also small details. It seems that the range technology overcomes optical methods in many aspects. However, the costs of laser scanning, both airborne and terrestrial, are still too expensive to be widely applied. Even in a few years if the costs were to decrease, cameras will still be mounted on range platforms because colour information is inevitable. The **video** based approaches are motivated by the fact that consequent video frames can be used as multi-perspective image sequences for 3D intersection, and that there is sufficient redundancy for more robust feature extraction and modelling. Beside the data sources listed above, **map data** is sometimes integrated during the early stages of building reconstruction to assist the data selection or provide prior knowledge such as the building footprints.

The geometry of a reconstructed model can be described with spatial enumeration, boundary representation, and constructive solid geometry (CSG). The **spatial enumeration** models such as meshes (Frueh et al., 2005) can be directly produced from sensor data. In the **boundary representation** models (Pu and Vosselman, 2009; Becker, 2009; Tian et al., 2009), the outlines of planar features are extracted, so the model sizes are much smaller than the spatial enumerations models. The data reduction of boundary representation also leads to much faster visualization. With **CSG**, a building model is composed from some fixed primitives by Boolean operators (union, difference, and intersection). The CSG models are always watertight, and require even fewer parameters than boundary representation. However, it is hard to represent complex objects with CSG, when they can be hardly decomposed into simple shapes. Besides, the heavy conversion of a CSG model to the boundary representation is always necessary for visualization purposes. Only a few methods (Haala and Brenner, 1999; Suveg and Vosselman, 2004) fit CSG building models.

Two major reconstruction principles can be distinguished: the data-driven method (Frueh et al., 2005; Cornelis et al., 2008; Becker and Haala, 2007) and the model-driven method (Schindler and Bauer, 2003; Taillandier and Deriche, 2004; Ripperda, 2008). In **data-driven** methods, the geometries are directly extracted from the sensor data and combined as the final model. In **model-driven** methods, a number of primitive templates are predefined, and then clues from the sensors are reasoned to match the most likely templates. In general, the data-driven methods should be preferred when a complex scene is to be reconstructed, because the predefined templates of model-driven methods might not be sufficient for complete

interpretation. In contrast, the data-driven methods are quite vulnerable to data quality, while the low quality can be well compensated by the model-driven methods. Airborne technologies document the ground information with a top-down view, and the captured buildings are mainly large-scale roof structures. Since several obvious roof patterns (flat, gable, hip, etc.) exist, the model-driven principle is often adopted in airborne data based reconstruction approaches. In contrast, the building facades contains much smaller structures, and the appearances are often influenced by human factors. It is quite difficult to set up a number of templates for the structures on building facades, hence a data-driven principle should be more reliable. Combination between the data-driven and model-driven approaches can be beneficial in terms of coping with data with various qualities. For example, Becker (2009) searches the geometry features extracted from TLS point clouds for regularities and hierarchical relationships, which serve as the verification rules for previous reconstruction and synthesis rules for the areas without sensor data.

Besides progress in the academic society, a number of building reconstruction packages (Cyclone, Phidias, CC-modeler, etc.) are also available on the commercial market. Considerable manual effort is still required by the current packages. Some examples of the manual operations are: fit a plane from a group of points, selecting tie points for image matching, and intersecting of two faces to determine an edge.

1.2.2 Frueh et al. 2005

Frueh et al. (2005) represent an early attempt which generated facade meshes by integrating terrestrial laser scanning data and digital images. The mobile data acquisition platform is made up of two laser scanners and a digital camera. The global location and pose of each scan are computed by adapting the laser points to nearby airborne data or DSM with the particle-filtering-based Monte-Carlo Localization algorithm. Such an “algorithmic-solution” to the mobile localization and pose estimation problem can be distinguished with the latest MLS systems such as Lynx and Streetmapper, where GPS and IMU signals are used in addition. Concerning the data processing, the laser point clouds are first used to generate depth images, and then foreground (occlusion objects) and background (building facades) are distinguished by histogram analysis. TIN mesh models are generated from background points, and textured with selections from photos. The texture holes caused by occlusions are filled in with textures from similar areas. This approach achieves very high automation and realistic results, but no simple geometry shape (such as a polygon) is reconstructed. The large amount of triangles (every street contains millions of triangle and texture pieces) results in slow visualization.

1.2.3 Cornelis et al. 2008

The method of Cornelis et al. (2008) attempted to create visually realistic 3D representations for car navigation. The survey platform consists of a pair of stereo calibrated video cameras and the GPS/INS units, which are also mounted on a

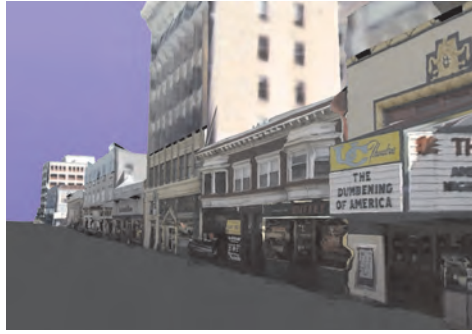


Figure 1.4: Reconstructed facade models by Frueh et al. (2005)

vehicle. The real-time video processing can be seen as an alternating process between scene reconstruction and car detection. The dense matching between stereo video frames is significantly accelerated by introducing a simple hypothesis that building facades are all vertical and the ground is a ruled surface. Detection of cars is based on the standard vote strategy for distinguishing background/foreground in images, with voting space greatly reduced by another hypothesis that cars are parked between building facades and grounds. The automation level, processing speed and visualization effect of this approach are quite impressive (see Figure 1.5). However, it needs to be pointed out that the building outlines are not recovered in this method as the method focuses only on visualization.



Figure 1.5: Reconstructed 3D city models after augmented by virtual car models (Cornelis et al., 2008)

1.2.4 Ripperda 2008

The work of Ripperda (2008) aimed at interpreting building facades with a description grammar. Facade elements (such as wall, door, and window) and abstract elements (such as repetition, symmetry and array) are written as terminals, and then the building facades are described by hierarchical composition of these terminals (see Figure 1.6). The reconstruction of a building facade can be seen as a

stochastic process of interpreting sensor data with the grammar. The probability distribution of the terminals is searched under supervision of the reversible jump Markov Chain Monte Carlo (rjMCMC) method. Regular shaped buildings with flat facades can be well represented with the grammar defined in Ripperda (2008), and the rjMCMC method is particularly appropriate for a zone of buildings with similar shapes. However, the stochastic analysis for complex facades, for example with small sized protrusions, might not lead to many clues with the current knowledge base.

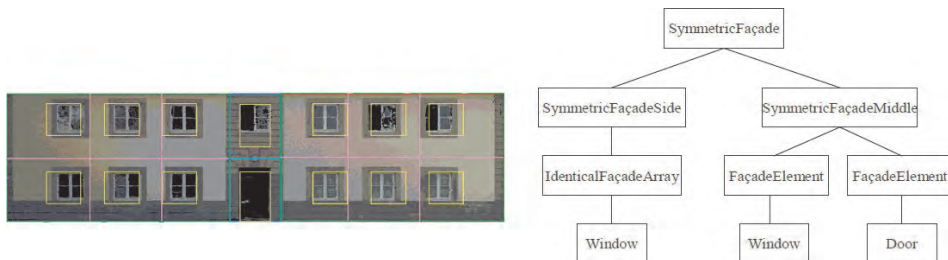


Figure 1.6: Interpretation of a building facade with the formal grammar defined in Ripperda (2008)

1.2.5 Becker 2009

Becker (2009) first reconstructs polyhedron models from terrestrial laser points and images, and then uses the clues extracted from the models to synthesize areas without laser data. To register laser point clouds and digital photos, intensity images generated from laser point clouds are used as a bridge. This turns the problem into an image-image registration problem, which can be solved by the SIFT (scale invariant feature transform) algorithm (Lowe, 2004). Assuming that no laser points are available from windows, laser points on window edges can be determined, and horizontal and vertical window border features are further extracted. Edges extracted by a Sobel filter from digital photos are used to refine the windows edges. The resulting model contains window frames with even window crossbars, which looks quite realistic. After the reconstruction step, a grammar which is similar to the one in Ripperda (2008) is used to interpret the reconstructed model. The discovered patterns can be propagated to the upper regions of the same facades to compensate the low point density, or even propagated to other buildings to create similar styled building models. Figure 1.7 illustrates the result of this approach. The textures are not present so far, although they can be mapped from images or pre-defined textures in any future work.

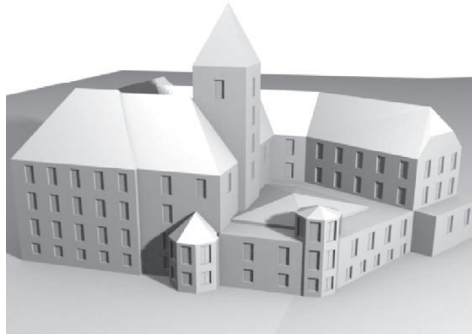


Figure 1.7: A building facade model reconstructed by Becker (2009)

1.3 Method overview

From the literature review discussed in the previous section, we realize that there are indeed a number of techniques that have been devised for automated building reconstruction, while only a few concern facade details. The lack of facade reconstruction approaches can be largely explained by the low efficient acquisition of ground based systems, such that most of the attentions is put into the post acquisition steps such as calibration and registration. However, the recent developments in laser scanners and cameras have made the documentation of street level scenes much easier and faster. Beside the MLS systems which were introduced earlier, integration of GPS/IMU also benefits the acquisition of close range images. For example, all Dutch cities have been photographed with panoramic images which are captured by a GPS/IMU mounted optical system (see Chapter 5 for more information about panoramic images). In summary, the acquisition and post acquisition are no longer difficulties for both terrestrial laser data and close-range images. Therefore our work focuses on the extracting and recovering of 3D geometries.

Reconstruction of buildings starts with recognition of structures from the raw data. Usually, the recognized structures are geometric features such as points, lines, or planes. In our approach, a higher level feature: the semantic feature, is defined, because we believe that understanding the meaning of each segment will benefit the facade reconstruction in several aspects. Firstly, a point cloud often contains points which do not belong to the building facade (for example, people walking by), or points of objects on the building which we do not want to include in the final model (for example, flower pots behind windows). These points can be filtered out if we know that they do not belong to the building. Secondly, laser scanning can hardly scan a building completely, as there are always occluded areas which cannot be reached by the laser beams. Knowledge can be used to fill these gaps by understanding the relationships between the features surrounding the gaps, and guarantee a water-tight model. Thirdly, a semantically labeled building model

enables a better visualization, by sharing the same texture in all polygons with the same feature label, or assigning a pre-defined texture to a certain semantic type. Finally, a semantic labeled building model can be associated with more attributes (name, owner, floor, etc) according to the semantic types, which helps integration with object based geo-databases.

Since laser data contains no colour information, the building models reconstructed from only laser data are simply wireframe models. To obtain photorealistic results, textures must be mapped from images to the wireframe models. The texture mapping can be seen as a procedure of transforming from model space to image space. If we desire automated texture mapping, the relative orientation of the texture image (or the camera) against the model coordinate systems must be determined first by spatial resection. Texture mapping has high demands for consistency between model and texture image, and even a small inconsistency can lead to rather poor visualization. After aligning the image space with the model space, the model edges and significant image lines are compared and adjusted in order to remove the possible inconsistencies.

The overall process of our reconstruction method is illustrated in Figure 1.8. In general, a first building facade structure is established with the planar features extracted from laser data, then image features are introduced in order to refine the model details. In the preprocessing stage, the exterior orientations of the images are calculated using a series of semi-automated operations. Then the planar features are extracted from terrestrial laser points and modelled as an initial polyhedron model. Because of the limitations of modelling algorithm, the initial model is still not accurate enough for texturing purposes, therefore significant line features extracted from images are compared with the initial models edges, and necessary refinements are made according to the image lines. Finally, during the texturing stage, textures of different model faces are selected automatically from multiple images to ensure the optimal visibility. Texture errors caused by occlusions in front of a wall are also removed by analyzing the locations of the wall, the occlusions and the camera position.

1.4 Structure of the thesis

The structures of the thesis naturally follows the reconstruction process as shown in Figure 1.8. Chapter 2 first introduces the theoretical background of knowledge, inference and the probabilistic based reasoning of sensor data. Based on the theory, Chapter 3 extracts planar features from terrestrial laser point clouds, and further determines the semantic meaning of each feature. In Chapter 4, features extracted in Chapter 3 are fit to some appropriate shapes such as polygons and B-spline surfaces. Hypotheses for the parts without sensor data are also made according to the knowledge base. Integration of imagery is elaborated in Chapter 5 and 6. In Chapter 5, images are first aligned to models after spatial resection, then the image lines are matched with model edges which are generated from laser data in

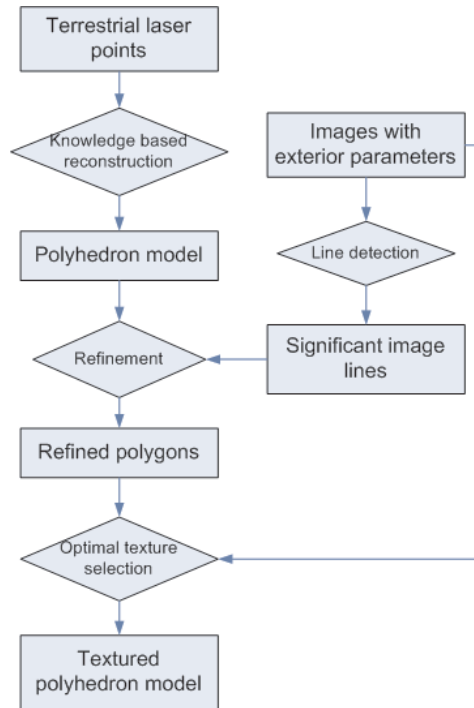


Figure 1.8: Building reconstruction pipeline

order to improve the geometry and remove inconsistencies between laser space and image space. With the camera parameters determined in Chapter 5, images are semi-automatically mapped to the models as textures as demonstrated in Chapter 6. The applicability and limitations of our methods are examined with several data sets in Chapter 7. Finally, the conclusions of this work and the recommendations for future work are stated in Chapter 8.

Knowledge engineering and reasoning

Despite the progress in sensor development, autonomous recognition of the building structures remains the most challenging task among the whole reconstruction process. In reality buildings come in a wide amount of types. Despite their varieties in function, shape, colour and material, humans seem to have no difficulty understanding building structures within their field of vision, and can even speculate the areas which are not directly visible. We believe such an intelligent ability depends on three prerequisites:

- The **depth perception** ability of human eyes to perceive the world in depth;
- The **knowledge** about a building's appearances, which can be acquired and reinforced through education and experience;
- The **logic** to combine knowledge and to reason with knowledge.

The aim of autonomous recognition is to establish a computer agent, who can finish the recognition task of feature recognition with minimal manual assistance, and achieve comparable precision as manual recognition. The performance of a recognition agent also relies on its observation, knowledge and logic. Close range images and terrestrial laser points cloud are the commonly used sensor data. Image based building reconstruction has been researched for years. From multiple 2D images captured from different positions, 3D coordinates of the image features (lines for example) can be calculated. Although acquisition of images is cheap and easy, the complexities of image understanding still make it difficult to automate the reconstruction using only images. Laser altimetry has been used more and more in recent years for automated building reconstruction. This can be explained by the explicit and accurate 3D information provided by laser point clouds. Researches (Vosselman, 2002; Frueh et al., 2004; Brenner, 2005) suggest that the laser data and images are complementary to each other, and efficient integration of the two

data types will lead to a more accurate and reliable extraction of three dimensional features.

Although plenty of information about the reality is provided by imagery and laser altimetry, a computer agent can not yield satisfactory results because it lacks the knowledge and logics to deal with the sensor data wisely. This chapter explains the procedure of training a knowledge-based agent which attempts to reproduce the performance of human recognition. In Section 2.1, the human knowledge of building recognition is collected, discussed and translated into machine-readable rules. Section 2.2 specifies how the agent should act with the rules and make recognition decisions. Because of the laziness and ignorance during knowledge engineering, the agent does not know how reliable the knowledge base and the decisions are (Russell and Norvig, 2003). In Section 2.3, we explain how the agent makes more reliable decisions by reasoning with uncertainty factors. Finally, some concluding remarks are given in Section 2.4.

2.1 Knowledge engineering

In this section we will develop an machine-readable knowledge base whose problem domain is specialized in the spatial distribution of building structures. The process starts with collecting of general human knowledge about building structures. Then the important domain-level concepts are translated into logic-level terms. Finally, the detailed rules describing relations between concepts are organized and listed. If not specifically stated, the term “building feature” in the rest of this thesis refers only to an exterior structure. Reconstruction of interior structures is not covered by this thesis.

2.1.1 Assembling the knowledge

What do we know about the buildings? Where can we acquire the knowledge about building structures? The term “building” has different meanings in different fields. In architecture, construction, engineering and real estate development, buildings refers to “the man-made structures used or intended for supporting or sheltering any use or continuous occupancy” (Wikipedia (2009)). As the products of human construction, a number of rules can be traced in different construction procedures. A straightforward method of acquiring building knowledge is to consult architects and civil engineers or even analyzing design blueprints. However, this knowledge is applied in different level of construction, and contains too much irrelevant information from what is actually concerned: the spatial distribution of building structures. The procedure of understanding, filtering and translation can be rather time consuming before the knowledge is directly applicable for feature recognition. However, an alternative assembling method is to summarize knowledge from observations and experiences. Buildings are very common objects in the world, and everyone has rich experiences about building appearances. The

knowledge of ordinary people should be sufficient to explain most observations, and be referenced to establish the knowledge base for an autonomous recognition agent.

As the products of human construction, the composition of buildings by various components is explicitly visible. These components, or **features**, serve as important context nodes during both construction and reconstruction process. We can either decompose a building according to component functionality, such as: *wall, roof, door, window, balcony, dormer, awning, chimney, water pipe* and *decoration*, or according to the spatial partitions, such as: *floor 1 (room 1.1, room 1.2, room 1.3...)*, *floor 2 (room 2.1, room 2.2, room 2.3)*, and so on. The components categorized with functions usually appear in different geometries, which are easily distinguishable from optical or range data. The partitioning categorization is beneficial for registration related purposes, such as cadastration, but the partitions are not always straightforward and may even bring ambiguity in complex cases.

It is not necessary and somewhat redundant to assemble all functional components into the knowledge base. Thus, the selection of features follows three considerations:

1. Only the features which are within the interests of the reconstruction are considered. For example, awnings and water pipes are not included in the knowledge base because they are not often desired in the reconstructed model.
2. The features which can be hardly captured in the sensor data are not considered. For example, the chimneys are seldom scanned from terrestrial laser scanner; some decorations are too small to be recorded clearly.
3. The features can be defined in the level of primitive or plane. At this moment, only planar features are considered because plane is an important unit in both human vision and knowledge, and planes can be easily extracted from laser point clouds with existing methods.

Each building feature has a number of **attributes**, which are similar within the same feature type. Strong clues can be extracted from the combination of feature attributes to suggest particular feature types. Some common attributes are considered below:

size The size is probably the most distinguishable attribute. It is most likely that walls occupies the largest space on a building, and usually windows are not longer than the height of a floor. The term size can refer to a feature's length, width, height, area, or volume.

position Some features can be expected at certain relative positions inside a scene. For example, ground is usually the lowest part, and roofs are seldom low. Positions of smaller structures (such as windows) are more random.

orientation Features' orientations are also predictable. For example, ground surfaces are somehow horizontal; walls are usually vertical as they are the supporting body of buildings; roofs are never vertical. It is very common that at least one wall of a building faces the street side.

shape As man-made objects, building features have regular and common shapes. In most cases, walls and roofs are flat, windows and doors are rectangular, etc.

colour Colour is probably the most important information for humans to perceive the world. The colour pattern of objects can be used to identify a large number of features. However, colour does not necessarily differ between features, and may bring ambiguity. For example, it is difficult to distinguish a door and its surrounding wall when they have the same colour, and shadows can bring significant colour contrast on the same feature. Furthermore, the colour information of laser points are "mapped" from optical data. Occluding objects in front of building facade will lead to the wrong colour mapping if the laser data and optical data are not acquired from the same location.

material Building features are usually made with particular materials, such as brick, glass, wood, and tile. Identification of material is difficult even for human being if only visual information is available.

Besides a feature's own attributes, some **spatial relations** between features can be predicted. These spatial relations are:

intersect Intersection is a basic spatial relation indicating the neighbouring relationship of two features. For example, walls always intersect with the ground; neighbouring walls often intersects perpendicularly; dormers must intersect a roof plane.

angle Angles between building features are often parallel or perpendicular. For example, protrusions on a wall usually have a face parallel with the wall; dormers on a roof often face the same direction with the wall facade supporting the roof.

inside If a feature is part of another feature, it is usually inside the boundary of the other feature. For example, windows of a wall must be inside the wall's boundary.

to certain direction of The locations of some features can be expected to certain direction of another features. For example, a roof can be expected above a wall; the facade face of a protrusion should be nearer to the street side than the supporting wall.

2.1.2 Decide on a vocabulary

It is clear from the assembled knowledge that a feature's functional type (or semantic type in other words) can be inferred from its attributes and the spatial relations with other features. The feature types, attributes, and spatial relations are the most important concepts to the knowledge base. Now these concepts are to be translated from general knowledge into logical terms.

The selected feature types are shown in Figure 2.1. Note that although ground is not a building feature, it is included in the feature type list because grounds are very helpful to locate the walls.

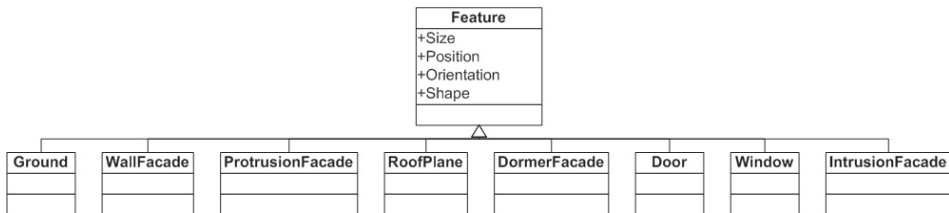


Figure 2.1: Features types

Firstly, the statement of a feature type is represented with a binary function:

$$IsType(arg1, arg2) = TRUE$$

where $arg1$ is a geometry feature; $arg2$ is a constant indicating $arg1$'s feature type.

For example, the assertion $IsType(F1, Wall) = True$ states that the type of $F1$ is $Wall$. An alternative way to indicate ontological distinction would be $Type(F1) = Wall$, but by doing so an individual feature can only have one type.

Then, the feature attributes and spatial relations are also represented with functions, which take (the index of) a particular feature as their sole argument. Sometimes, a feature's own attributes does not give a useful clue if the relative situation is unknown. Therefore, some relative attributes and their enquiry functions are introduced. Determination of relative information requires sorting of particular attributes throughout a group of features. For example, a feature is considered *Large* if its area is ranked among the first 30 percent within a sorted list of area values in descending order. The attribute functions are:

- $Area()$, $Width()$, $VerticalLength()$, $IsLarge()$; $IsWide()$, $IsTall()$
- $Center()$, $Height()$, $IsHigh()$, $IsLow()$;
- $NormalVector()$, $IsVertical()$, $IsHorizontal()$;
- $IsRectangle()$;

The spatial relation functions take two or three arguments (the third argument indicates a reference feature). They are:

- $Intersect(arg1, arg2)$;
- $Angle(arg1, arg2)$, $IsParallel(arg1, arg2)$, $IsPerpendicular(arg1, arg2)$;
- $Inside(arg1, arg2)$, $ProjectionInside(arg1, arg2)$;
- $LeftOf(arg1, arg2, arg3)$, $RightOf(arg1, arg2, arg3)$, $Nearer(arg1, arg2, arg3)$, $Above(arg1, arg2)$, $Below(arg1, arg2)$;

2.1.3 Encode general knowledge

Below the initial rules of the knowledge base are specified in terms of first-order logic (Russell and Norvig, 2003). Note that only flat planes are considered at the moment because of their simplicity. The free form features will be specially handled in Section 3.2.2 and Section 4.2.2.

1. A ground feature is a low, large, and horizontal plane:

$$\begin{aligned} & \forall f \quad IsType(f, Ground) \\ \implies & IsLow(f) \wedge IsLarge(f) \wedge IsHorizontal(f) \end{aligned} \quad (2.1)$$

2. A wall facade feature is a large and vertical plane which intersects ground:

$$\begin{aligned} & \forall f1 \quad IsType(f1, WallFacade) \\ \implies & \exists f2 \quad IsLarge(f1) \wedge IsVertical(f1) \wedge Intersect(f1, f2) \\ & \wedge IsType(f2, Ground) \end{aligned} \quad (2.2)$$

3. A wall facade feature is not inside of any other feature.

$$\begin{aligned} & \forall f1 \quad IsType(f1, WallFacade) \\ \implies & \exists f2 \quad \neg Inside(f1, f2) \end{aligned} \quad (2.3)$$

4. A roof plane feature is a non-vertical plane, and it intersects a lower wall facade or a lower dormer facade:

$$\begin{aligned} & \forall f1 \quad IsType(f1, RoofPlane) \\ \implies & \exists f2, f3 \quad \neg IsVertical(f1) \wedge \\ & ((Intersect(f1, f2) \wedge IsType(f2, WallFacade) \wedge Above(f1, f2)) \vee \\ & (Intersect(f1, f3) \wedge IsType(f3, DormerFacade) \wedge Above(f1, f3))) \end{aligned} \quad (2.4)$$

5. A door feature is a low, small, and vertical plane, and it is inside a wall facade:

$$\begin{aligned} & \forall f1 \quad \text{IsType}(f1, \text{Door}) & (2.5) \\ \implies & \exists f2 \quad \neg \text{IsHigh}(f1) \wedge \neg \text{IsLarge}(f1) \wedge \text{IsVertical}(f1) \\ & \wedge \text{Inside}(f1, f2) \wedge \text{IsType}(f2, \text{WallFacade}) \end{aligned}$$

6. A window feature is a small hole inside a wall facade.

7. A protrusion facade feature is a plane whose projection onto a wall facade is inside this wall facade, it does not intersect any ground, and is nearer to the street side than this wall facade:

$$\begin{aligned} & \forall f1 \quad \text{IsType}(f1, \text{ProtrusionFacade}) & (2.6) \\ \implies & \exists f2, f3 \quad \text{IsType}(f2, \text{WallFacade}) \wedge \text{ProjectionInside}(f1, f2) \\ & \wedge \text{IsType}(f3, \text{Ground}) \wedge \neg \text{Intersect}(f1, f3) \wedge \text{Nearer}(f1, f2, f3) \end{aligned}$$

8. A protrusion facade should intersect another protrusion facade and a wall facade to form a two-facade protrusion; or together with another two protrusion facades to form a triple-facade protrusion:

$$\begin{aligned} & \exists f1, f2, f3, f4 \quad \text{IsType}(f1, \text{ProtrusionFacade}) & (2.7) \\ & = \text{IsType}(f2, \text{ProtrusionFacade}) = \text{IsType}(f3, \text{ProtrusionFacade}) \\ & = \text{IsType}(f4, \text{WallFacade}) = \text{True} \\ \implies & (\text{Intersect}(f1, f2) \wedge \text{Intersect}(f1, f4) \wedge \text{Intersect}(f2, f4)) \vee \\ & (\text{Intersect}(f4, f1) \wedge \text{Intersect}(f1, f2) \wedge \text{Intersect}(f2, f3) \wedge \text{Intersect}(f3, f4)) \end{aligned}$$

9. A intrusion facade feature is a plane whose projection onto a wall facade is inside this wall facade, and it is further to the street side than this wall facade:

$$\begin{aligned} & \forall f1 \quad \text{IsType}(f1, \text{IntrusionFacade}) & (2.8) \\ \implies & \exists f2, f3 \quad \text{IsType}(f2, \text{WallFacade}) \wedge \text{ProjectionInside}(f1, f2) \\ & \wedge \text{IsType}(f3, \text{Ground}) \wedge \text{Nearer}(f2, f1, f3) \end{aligned}$$

10. A dormer facade feature is a vertical plane which is above a roof plane and intersects this roof plane:

$$\begin{aligned} & \forall f1 \quad \text{IsType}(f1, \text{DormerFacade}) & (2.9) \\ \implies & \exists f2 \quad \text{IsVertical}(f1) \wedge \text{IsType}(f2, \text{WallFacade}) \wedge \text{Above}(f1, f2) \\ & \wedge \text{Intersect}(f1, f2) \end{aligned}$$

2.1.4 The hierarchical composition

The hierarchical composition of building by features is illustrated as a UML model in Figure 2.2. Beside the feature types discussed in the previous subsections, some *body* types (see the nodes with italic name) are introduced to help understanding the relations between features, while these body types are not instantiable by any sensor data. A large number of buildings can be interpreted using the semantic network model (Sowa, 1992) in Figure 2.2, and modification and extension can be easily achieved by adjusting the layout or inserting new nodes. Figure 2.3(a) shows a simple house and its interpretation with hierarchical features. The connection lines in Figure 2.3(b) indicate the relation of “PartOf” between the connected features.

2.2 Reasoning with the knowledge

2.2.1 The proof tree

A number of features have been defined and described in the previous section. If our definitions are flawless and the building structures are simple enough, the implication between a term and its descriptions should be mutual, or in other words, the description clauses can be used directly as the statement clauses to uniquely determine a feature type. The statement clauses are categories of two groups: the **observation clause** which describes the data and the **fact clause** which is inferred from observation clauses. The path from observations to goals can be organized as a proof tree (Pereira and Shieber, 2002), which is a specialized Bayesian network with all variables at the same level independent of each other. The observation clauses serve as the leaves of the proof tree, and the fact clauses serve as the branches. For example, suppose there is only a ground and a wall facade in the scene. The proof tree for interpreting this scene is shown below in Figure 2.4.

There are two major methods to reason with the proof tree: the forward chaining and backward chaining.

Forward chaining

A number of observation clauses can be written for the available data. By searching for the patterns within the existing clauses, new clauses may be inferred. Combination of these new clauses with the existing clauses may form more patterns, and trigger more inferences consequently. This procedure is repeated until no new patterns can be discovered. Because the data determines which clauses are selected and used, the forward chaining is also called data-driven.

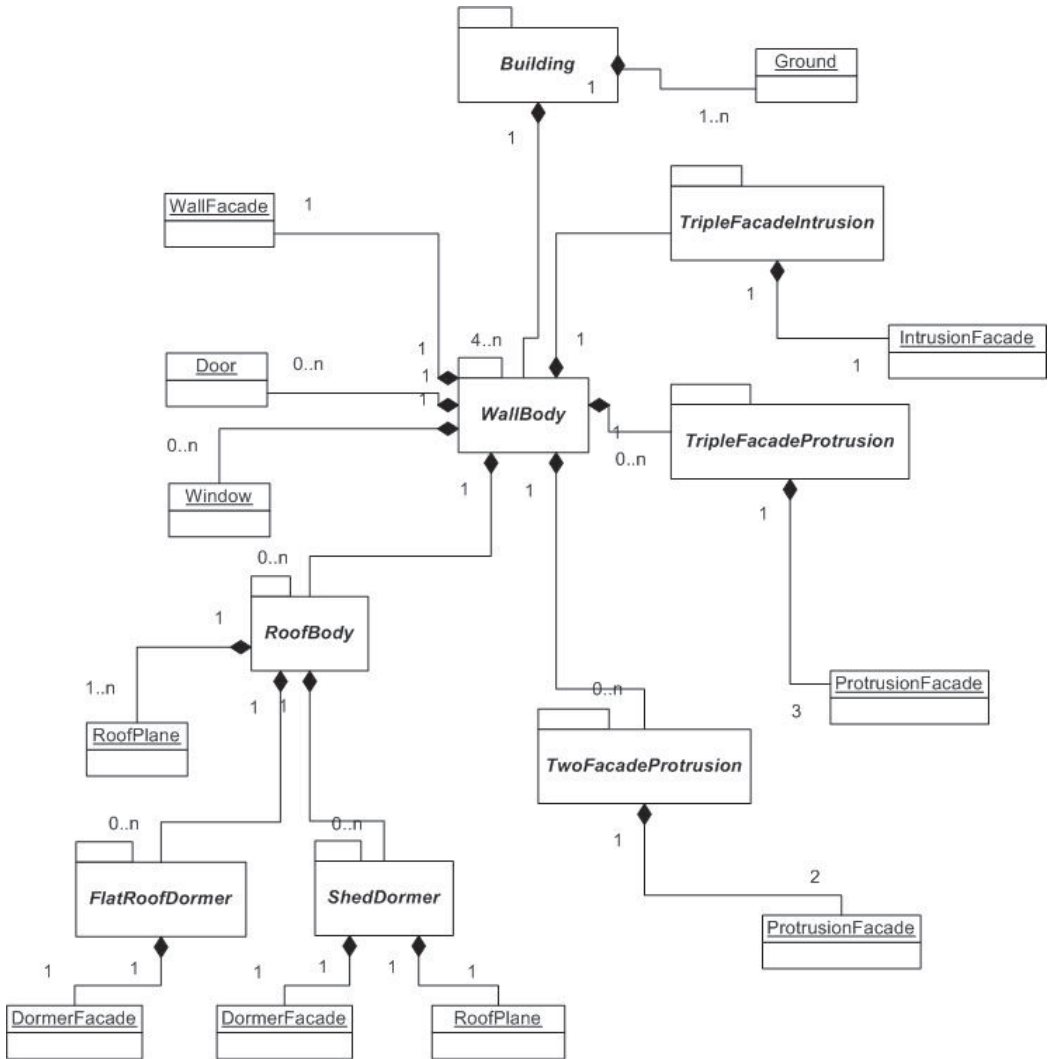


Figure 2.2: Composition of building features in a semantic network.

The forward chaining of the proof tree in Figure 2.4 is given below:

1. Create an empty clause pool.
2. All possible observation clauses are written for $f1$ and $f2$: $IsLarge(f1)$, $IsVertical(f1)$, $\neg IsHorizontal(f1)$, $\neg IsLow(f1)$, $IsLarge(f2)$, $\neg IsVertical(f2)$, $IsHorizontal(f2)$, $IsLow(f2)$, $Intersect(f1, f2)$. Add them to the clause pool.
3. Search the current clause pool, and a pattern $IsLarge(f2) \wedge IsLow(f2) \wedge$



Figure 2.3: Interpretation of a simple house with hierarchical feature composition.

$IsHorizontal(f2)$ is found.

4. A new clause: $IsType(f2, Ground)$ is inferred from the pattern. Add this clause to the clause pool.
5. Search the current clause pool, and a new pattern: $IsLarge(f1) \wedge IsVertical(f1) \wedge IsType(f2, Ground) \wedge Intersect(f1, f2)$ is found.
6. A new clause: $IsType(f1, WallFacade)$ is inferred from the pattern. Add this clause to the clause pool.
7. Search the current clause pool. If no new pattern is found, the iteration ends.

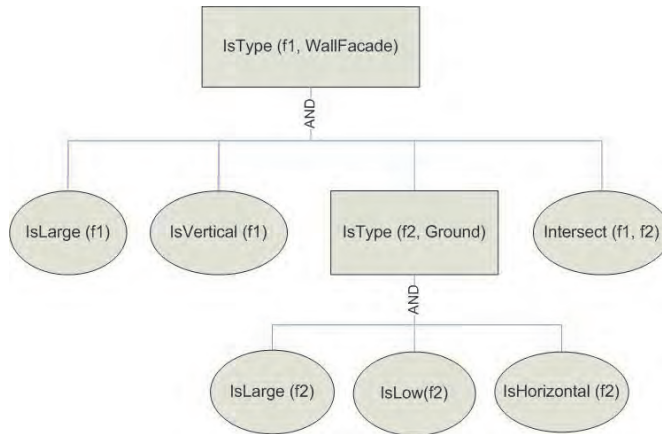


Figure 2.4: A simple proof tree.

Backward chaining

The backward chaining starts with a particular goal, and tries to prove this goal by searching the supporting clauses. If some supporting clauses are not yet proved, the inference engine will set these clauses as new goals, and tries to prove them first. This procedure iterates until all goals are proved.

The backward chaining of the proof tree in Figure 2.4 is given below:

1. Set up a goal: $IsType(f1, WallFacade)$. Push the goal to an empty goal stack.
2. According to the knowledge base, the supporting clauses to achieve this goal are: $IsLarge(f1)$, $IsVertical(f1)$, $IsType(f2, Ground)$, and $Intersect(f1, f2)$.
3. The first, second and last clause are proved directly from observation, while $IsType(f2, Ground)$ requires further proof.
4. Set up a new goal: $IsType(f2, Ground)$. Push it to the goal stack.
5. According to the knowledge base, the supporting clauses to achieve this goal are: $IsLarge(f2)$, $IsLow(f2)$ and $IsHorizontal(f2)$.
6. All the clauses can be proved directly from observation, so $IsType(f2, Ground)$ is proved.
7. The next goal in the stack: $IsType(f1, WallFacade)$ is proved.
8. There are no more goals in the stack. The iteration ends.

Forward chaining is more efficient when there are a lot of things to prove and there are a small set of clauses concerned. Besides, it is very easy to incorporate human assistances and additional data sources. But forward chaining is very vulnerable to the quality of observation, and may generate a lot of irrelevant inferences. Backward chaining is better when a single goal needs to be proved from a complex set of clauses, but may produce incomplete solutions or fall into repeated loops. Building features' complexity are different, and they are documented with different level of completeness in the sensor data. It is rigid to adopt the same inference mechanism and apply it to different feature types. Wall facades and grounds are simple objects, and they are very well recorded in any sensor data. In our feature recognition algorithms described in Chapter 3, these two feature types are recognized using forward chaining. Doors, roof planes, dormer facades, protrusion facades and intrusion facades are smaller structures with more complex patterns than wall facades and grounds, so inferencing of these types are proceeded in the backward order. Windows are extracted with a special data-driven algorithm because of the different nature. This algorithm is elaborated in Chapter 3.

2.3 Managing uncertainty

In the knowledge base and the inference engine described in previous sections, we assume that we know the whole truth about building structures, and all the required facts are accessible. Unfortunately, this is seldom possible for both human recognition and autonomous recognition. Understanding of building structures is a complex problem, and there are always cases beyond our knowledge. For example, we describe a “wall facade” as a large and vertical plane which intersects ground, but satisfaction of $IsLarge(f1) = IsVertical(f1) = Intersect(f1, f2) = IsType(f2, Ground) = True$ does not guarantee $f1$ to be a wall facade. $f1$ may also be a side of a truck, or a large advertisement board. According to Russell and Norvig (2003), the possible failure of knowledge based feature recognition is caused by three reasons:

- Laziness: It is too much work to list the complete set of clauses to ensure an exceptionless rule. For example, one may argue that walls can be distinguished from the advertisement boards by satisfying another clause: $Intersect(f1, f3) \wedge IsType(f3, WallFacade)$. However, too much specialization cannot fully solve the problem either, but makes the knowledge base too heavy to handle.
- Theoretical ignorance: We cannot foresee all the possible situations.
- Practical ignorance: Even if the rules cover all situations, we might not be able to collect all required information. For example, the fact “ $f2$ intersects another wall facade $f3$ ” may not be recorded in the sensor data because of the acquisition angle.

So the agent we have developed so far has an uncertain knowledge base and inference engine, and it cannot be fully sure about whether a feature is a wall facade, or simply believe a feature is large or small. However, the agent should at least provide some most likely solutions, if we teach it how to act with uncertain observations and facts, or let it learn from its own experiences. This section discusses how the agent's knowledge base and inference engine is adapted to manage uncertainty. In general, the rules in the knowledge base are associated with a number of uncertainty factors, and the inference engine is enhanced with the ability of reasoning towards expectations.

2.3.1 Describing the uncertainty

Uncertain situations are very common in human society. When making a decision, we know which pieces of available evidences are more reliable, i.e. that should be considered more, and which ones are less reliable that should be less involved or not involved at all. The reliability of evidences can be separated into two sorts. Firstly, a piece of evidence (or clause) itself may not be 100 percent certain. Only clean water is drinkable, but how clean it should be? Secondly, one type of evidence is reliable under certain circumstances but might be less reliable under other circumstances. Vehicles coming from right hand should go first at equal crossings, but this traffic rule should not be followed when traffic lights are present. We describe the first kind of reliability with the **truth factor**: T , which indicates the absolute reliability of a clause. The value range of T is from 0 to 1: a clause i is true when $T_i = 1$ and false when $T_i = 0$. The second kind of reliability is described with the **weight**: w , which indicates the relative reliability or priority of a clause. The value range of w is also from 0 to 1: $w_i = 1$ indicates clause i should be fully trusted over other clauses and $w_i = 0$ indicates clause i should not be trusted at all.

Suppose a clause I is supported by a number of clauses i . The truth factor of I is derived from i by:

$$T_I = \sum_{i=0}^n w_i T_i \quad (2.10)$$

And weights of the supporting clauses satisfy:

$$\sum_{i=0}^n w_i = 1 \quad (2.11)$$

The truth factor value of an observation clause is determined as such: if this observation clause concerns the absolute attributes or spatial relations (for example $IsVertical()$ and $Intersect()$), the value will be either 0 or 1; if the relative attributes are concerned, the value will be distributed between 0 and 1 with defined distribution functions. For example, the truth factor of $IsLarge()$ is $T_i = 1 - (r_i - 1)/(n - 1)$, where r_i is the ranking of this attribute among all

features in descending order¹; the truth factor of $IsLow()$ is $T_i = 1 - (Height(f_i) - Height_Min) / (Height_Max - Height_Min)$. The optimal valuing of weights is dependent on data quality and the building style. In general, the relative attributes should be given higher priority when the data quality is bad, and a good values set for one building type should also work well within similar building types.



Figure 2.5: A simple building with four features.

Figure 2.5 shows a simple building with four features. Suppose we wish to recognize grounds using forward chaining, where three clauses are concerned according to Equation 2.1. The truth factors of these observation clauses are given in Table 2.1. If the three clauses are equally treated, their weights are all set to 0.33. Applying Equation 2.10 derives the truth factors of $IsType(f_i, Ground)$ (see Table 2.1). If the **likelihood threshold** L is set to 90 percent, then only $f1$ is believed to be a ground.

	f1	f2	f3	f4
$T_A: IsLarge(f_i)$	0.75	0.5	1	0
$T_B: IsHorizontal(f_i)$	1	0	0	0
$T_C: IsLow(f_i)$	1	0.73	0	0.21
$T: IsType(f_i, Ground)$ $= 0.33T_A + 0.33T_B + 0.33T_C$	0.92	0.42	0.33	0.07

Table 2.1: Recognize grounds in Figure 2.5 with truth factors

Now the inference goes on to search for the pattern for a wall facade. There are four clauses influencing the decision of a wall facade as indicated Equation 2.2. This time we wish to rely more on the vertical clause, and the results are given in

¹Implementation of this function is slightly different. See Chapter 3 for details.

Table 2.2. It is obvious that the likelihood threshold will decide whether f_2 (wall of a small garage) is kept as a wall facade or not.

	f2	f3	f4
$T_A: IsLarge(f_i)$	0.5	1	0
$T_B: IsVertical(f_i)$	1	1	1
$T_C: Intersect(f_i, f_1)$	1	1	0
$T_D: IsType(f_1, Ground)$	0.92	0.92	0.92
$T: IsType(f_i, WallFacade)$ $= 0.2T_A + 0.4T_B + 0.2T_C + 0.2T_D$	0.89	0.98	0.58

Table 2.2: Recognizing wall facades in Figure 2.5 with truth factors.

The truth factor and weight provide a quantifiable method to describe uncertainty, and make it possible to build simulation models to handle uncertain problems. The truth factor of a fact clause can also be seen as the probability of this clause. The problems in reality can be very complicated, and we are aware that the recognition of building features is far more complex to be exactly represented with this model. But it is vital that an agent is able to judge the values of facts according to different situations, instead of rashly believing or not believing at all. Furthermore, more advanced simulation model can be developed in the future from deeper knowledge mining or statistical learning from observations.

2.3.2 Making expected decisions

With the introduction of uncertainty factors, it is possible for a feature to have multiple semantic meanings, and a group of features to have multiple interpretations. Theoretically, a feature may be of any type, just with higher or lower truth factors. It is undoubted that we would like the agent to recognize each feature with high accuracy, but this does not mean the implication with a higher truth factor should always be preferred. This is because minimizing local uncertainties might bring the consequence of increasing the global uncertainty. Let's take Figure 2.6(a) for example. The truth factor of the implication $IsType(f_1, WallFacade)$ is 0.9, and the truth factor $IsType(f_1, ProtrusionFacade)$ is 0.53 (see Equation 2.6-2.7, weights are equalled). Although f_1 is "more likely" to be a wall facade, the asserting of f_1 to be a wall facade would make f_2 's not likely to be any feature type (truth factors are all too low). Alternatively, if f_1 is asserted to be a protrusion facade, then f_2 are possibly to be also a protrusion facade (with truth factor 0.53), which makes a large difference to the global interpretation.

In order to measure a group of features' overall reliability, we introduce another concept called the reliability score, which can be calculated with:

$$S = \sum_{i=0}^n Area(i)t_i \quad (2.12)$$

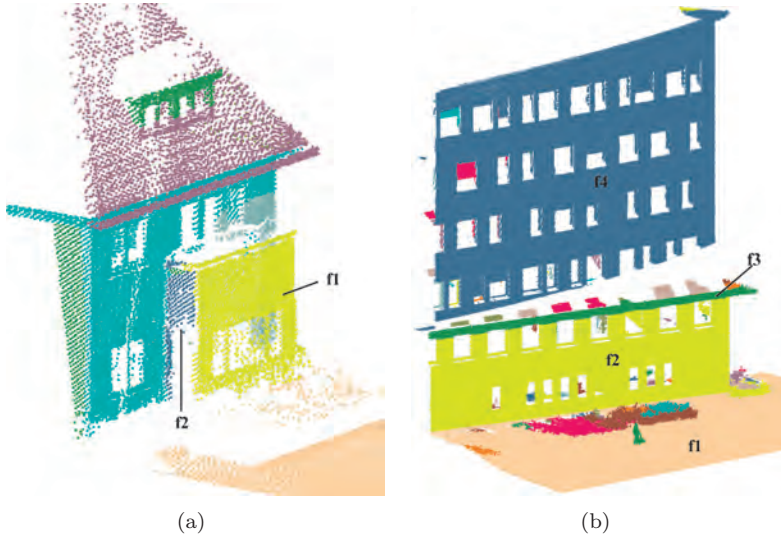


Figure 2.6: Two buildings with multiple possible interpretations.

$$t_i = \begin{cases} T_i & \text{if } T_i \geq L, \\ 0 & \text{if } T_i < L. \end{cases} \quad (2.13)$$

where L is the likelihood threshold.

The procedure of recognizing features in Figure 2.6(b) is given as follows:

1. The likelihood threshold is set to 60%. The weight of $IsType(f, WallFacade)$ is set to (20%, 40%, 20%, 20%). The weights of all other clauses are set equally.
2. Search for ground. Only $f1$ is above the likelihood threshold to be a ground, and the probability is 95%.
3. Search for wall facades, and we get $f2(60\%)$ and $f4(96\%)$.
4. Search for the other features, and we get: $f3$ has 98% probability to be a roof facade; $f4$ has 98% probability to be an intrusion facade (inside $f2$);
5. Calculate the reliability scores of possible solutions (see Table 2.3). It is obvious that the scoring of solution 2 is higher than solution 1.

Among the possible interpretations, normally the one with the maximum reliability score is chosen as the best solution. Setting a stricter likelihood threshold can reduce the number of possible interpretations and therefore the computation cost is lighter. But some features may not be recognized at all if the likelihood threshold is too strict. Usually the different interpretations lead to the similar geometry

outlining, but may result in differences in the final reconstructed model when the model parts are rendered with pre-defined textures which are related with feature types. A human operator can also choose an alternative solution depending on specific applications.

	f1	f2	f3	f4
Area (m^2)	116	94	15	302
Solution 1	Ground(0.95)	WallFacade(0.96)	RoofFacade(0.98)	WallFacade(0.6)
Solution 2	Ground(0.95)	WallFacade(0.96)	RoofFacade(0.98)	IntrusionFacade(0.98)

Table 2.3: Value list to calculate reliability scores.

2.4 Concluding remarks

In this chapter, we have transferred human knowledge to the computer agent, so that it can achieve more intelligent recognitions by simulating the human behaviours. We will demonstrate the effectiveness of this agent by extracting semantic features from terrestrial laser points in the next chapter. As mentioned earlier, the current classification of feature types may not be sufficient to interpret complex buildings, and the optimal weights vary between different building styles. Unsupervised modification of the knowledge base by learning from sensor data should be researched in the future.

Feature extraction

This chapter focuses on the application of the theory developed in the previous chapter. A flowchart of the feature extraction process is provided in Figure 3.1. The process starts with a spatial index of the raw laser data, so that the selection of working data set is efficient. Then a local terrestrial laser point cloud is segmented into either flat or curved surface patches. Finally, geometry extracted from each segment is reasoned with the proof trees to decide a most likely feature type. Section 3.1.1 to 3.3.3 elaborates these steps sequentially.

3.1 Preprocessing

Processing of laser point clouds is a challenging task because of the huge amount of data. All mainstream stationary laser scanners are able to provide point clouds with density up to thousands points/ m^2 in street level scanning, so there can be hundreds of thousands of laser points from a single wall facade. In practice, multiple buildings within a single street scene are usually scanned together. Instead of processing the whole point cloud directly, it is wise to first extract points belonging to specific buildings, and then apply the reconstruction methods to these local data sets. In our method, a complete point cloud is firstly partitioned to indexed grids, then the two dimensional boundary polygon of a specific building is derived from ground plan, and finally the points of this building are extracted from the grids which overlap the boundary polygon.

3.1.1 Spatial indexing

A raw laser scanning data set is usually provided as rows of points in the format of: coordinate(x,y,z), colour(r,g,b) and reflectance value. In our indexing method, the point sets are partitioned by the 2D coordinates, because most buildings are bounded by a 2D footprint. The method first determines the minimum and maximum x and y values of the whole data set. By specifying a grid size (for example 20 meters), the number of grids (N) in x and y directions (M) can be calculated.

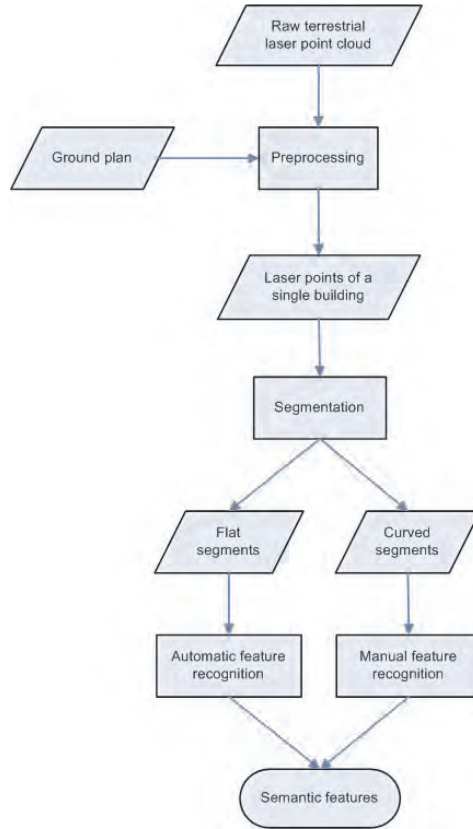


Figure 3.1: The process of feature extraction

Then the index number of each point in both dimensions are calculated, and the points are reallocated in the disk storage by the consequent index numbers (first x then y). Finally, the grids without any points are labelled so that later a spatial query will avoid these grids.

Figure 3.2 illustrates the indexing of a laser point cloud with 18 million points. In total there are 21 by 17 grids partitioned, with a grid size of 15 by 15 meter. The overall partitioning time is linear with the number of points in the data set, as the algorithm complexity is $O(2n)$. In the case of Figure 3.2 it took a dozen of minutes to complete the spatial partitioning.

3.1.2 Extracting points of interest

As the knowledge base describes the building features within a single building, it is desired to extract the laser points by the buildings they belong to. In most cases, buildings are supported by vertical walls and have fixed 2D outlines. These

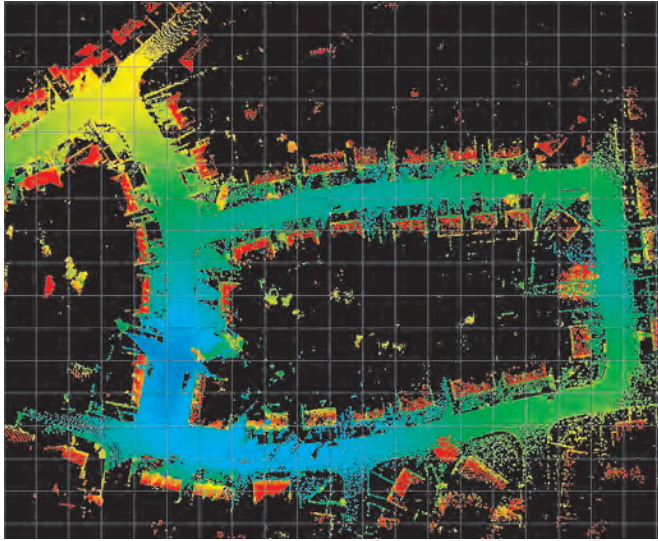


Figure 3.2: Partitioning of a laser point cloud.

outlines can be retrieved directly from 2D ground plan (Figure 3.3(a)), or by horizontal slicing of the laser points at an appropriate height if the 2D ground plan is not available (Figure 3.3(b)). There are a few buildings which are larger in the upper part than the lower part. The laser points of these building should be manually located and extracted.



Figure 3.3: Building's 2D outlines.

Once one or more outline edges are selected, they are firstly grouped to a polygon, which is then expanded for two meters to form the ROI (region of interest) polygon for the points selection. The expansion is necessary because the 2D outlines

indicate the walls' location, while balconies and roofs often extrude outside the walls. Another reason for this expansion is that the ground plans may not be accurate. Once the ROI polygon is determined, its range in x and y directions are both calculated to estimate which grids may contain the interested laser points. Then each point within these grids is tested with a "point in polygon" algorithm to decide whether it should be extracted.

An alternative approach for point clouds management is to use the spatial databases. For example, Oracle Spatial 10 supports a number of geometry types (point, line, curve, polygon, etc.), and can build R-tree indexing over the stored geometries. Experiments show that processing of the data set in Figure 3.2 is not faster by Oracle Spatial than our partitioning method. But it is expected that Oracle Spatial should be faster when manipulating larger data sets, and should be more efficient when the ROI has an arbitrary shape. Besides, managing point clouds with spatial databases should be beneficial to the collaboration efficiency. When multiple tasks are involved in a city modeling project, a spatial database would be convenient to distribute the data to various subtasks, and maintain the results from the subtasks.

3.2 Extraction of geometric features

A semantic feature consists of surface patches which are either flat or curved. Before identifying the semantic meaning of a surface patch, we first need to extract these surface patches from the random points, or in other words, group points according to their coplanarity. This process is often referred as the **segmentation** of point clouds. A number of segmentation algorithms for point cloud are available, and we adopted the surface growing algorithm in Vosselman et al. (2004) to extract flat surfaces and the smooth surface algorithm in Rabbani et al. (2006) to extract curved surfaces.

3.2.1 Flat surfaces

We include a short explanation of the surface growing algorithm here because of its strong relevance to our feature recognition method. The algorithm consists of the following steps:

1. Determine a seed surface. A seed surface consists of a group of nearby points that fit well to a flat plane. The algorithm selects an arbitrary unclassified point and tests if a minimum number of nearby points can be fitted to a plane. If this is the case, these points constitute the seed surface. Otherwise, another arbitrary point is tested.
2. Grow the seed surfaces. The growing can be based on one or more the following criteria:

- **Proximity of points.** Only points within certain distance to a seed surface, or the neighboring points (in kd-tree) of a seed surface, can be added to this seed surface.
- **Globally planar.** For this criterion a plane equation is determined by fitting a plane through all surface points in this seed surface. Points can only be added if the perpendicular distance to the plane is below some threshold.

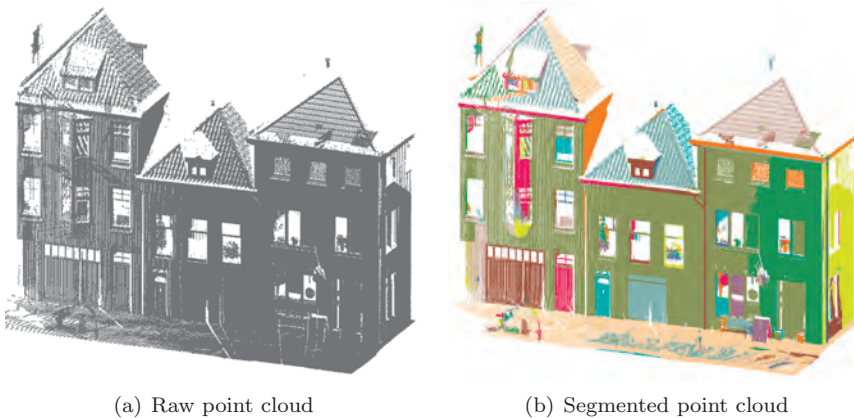


Figure 3.4: Segmentation result of a building facade

Figure 3.4(a) shows the original laser scanned point cloud of a building facade, and Figure 3.4(b) shows the segmentation results of this point cloud. In the ideal situation, each segment should represent one of the semantic features categorized in Figure 2.1. In practice this can be hardly achieved because:

- A laser point cloud always contains irrelevant data, such as the objects behind windows, cars, and benches besides buildings. The segmentation results normally contain these irrelevant segments too. The irrelevant segments can be filtered out in the semantic feature extraction step.
- Ideal segmentation parameters are difficult to determine. Even the best possible segmentation parameters will result into some over-segmentation (one feature segmented to several segments), under-segmentation (several features segmented to one segment), or miss detection (feature is not segmented).
- The ideal segmentation parameters are dependent on the point density. There are fewer laser points reflected from more distant parts and tilted parts. Segments of these parts may be missing because the point density is too low.

Therefore we are seeking the optimal segmentation parameter values, such that each semantic feature is represented with as little number of segments as possible. Over-segmentation is preferred over under-segmentation when large and tiny

structures coexist in the same data set. This is because over-segmented parts can be combined later in the modelling stage when they share certain properties, but under-segmented parts are difficult to split again. For our experiments on several data sets we determined a set of optimal parameter values for point clouds with at least 100 points per square meter (10 centimeter point spacing).

From the segmentation result in Figure 3.4 it is clear that most wall facades, doors, roof planes and protrusion facades are roughly distinguished. But the wall facade and one of the roof plane are segmented into two pieces, and some protrusion facades and dormer facades are segmented into three or more pieces. Segmentation of a window is relatively good if curtains are covering the windows, otherwise none or a few segments are generated on the window frames.

As introduced in Chapter 1, the laser points can be coloured by mapping the pixels of digital images which are taken simultaneously with the laser scanning. An attempt was made to improve the segmentation by using colour information of laser points (Sapkota, 2008). The colour assisted segmentation is not guaranteed to be better than the ones using only coordinates. The colour can be incorrectly mapped to laser points of occluding and occluded objects when the laser scanner and the camera were located at different positions. Even if the colour mapping is correct, the ambiguous colour caused by illumination conditions (such as shadow) can result in many extra segments which actually belong to one plane. In Figure 3.5 left, the coloured laser point cloud contains pixels of two cars parking in front of the building; therefore the segmentation result contains unnecessary extra segments in the positions of the cars as shown in Figure 3.5 right.

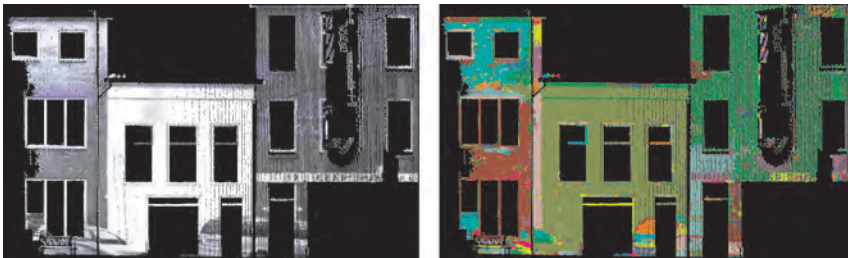


Figure 3.5: Segmentation using colour information (left: coloured point cloud; right: segmentation result).

3.2.2 Curved surfaces

A number of building features appear as smooth curved surfaces, which can be distinguished using an effective smooth surface segmentation algorithm as presented in Rabbani et al. (2006). Although the surface growing method can also segment curved surfaces, the smooth surface method works better with cylindroid surfaces which are often seen on modern buildings. The stages of the smooth sur-

face method are similar to the surface growing method, including the steps of seed region fitting and region growing. The most significant difference is that normals are used in the smooth surface method to filter out any noisy points. Specifically, after a flat seed surface is determined, the residues of the seed plane fitting are further examined in order to figure out whether they arise from noise or from a cylindroid surface.

Figure 3.6 shows the segmentation of some curved building facades using the smooth surface method. Although the result is satisfactory, the processing speed with the smooth surface method is much slower than the surface growing method, because the normal calculation and the cylindroid surface fitting are both computationally expensive. In practice it is preferred to segment a laser point cloud for flat geometry features with the surface growing method first, considering flat surfaces are usually the dominant building shapes. If some curved surfaces are present, they will be segmented to multiple small flat pieces and should be easily detected. Then the point cloud region with curved features should be locally segmented using the smooth surface method, and later on reconstructed with the outlining method for freeform geometries (see Section 4.2).



Figure 3.6: Segmentation with the smooth surface method.

3.3 Extraction of semantic features

From the geometry features extracted after the segmentation, the next level feature: semantic features, can be further extracted. The principle of knowledge based feature recognition is given in the previous chapter. In order to establish

a proof tree for reasoning, the relevant attributes and spatial relations of laser segments should be determined first. This information can not be directly calculated because a laser segment is a group of points. Instead, each segment is approximated with its convex hull, and the relevant information of the convex hull is calculated for reasoning. Figure 3.7 illustrates this approximation strategy. The size of a segment is approximated with its convex hull polygon's area; the position is approximated by its convex hull polygon's geometry center; the orientation is approximated by the normal vector of the plane of the convex hull; the topology of two segments is determined by intersecting their convex hull polygons.

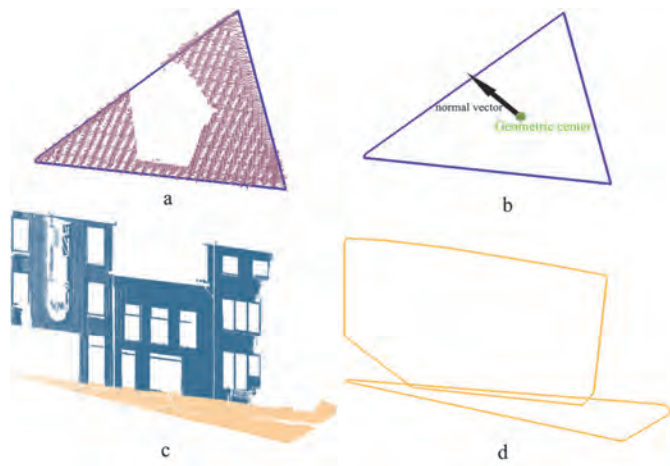


Figure 3.7: Convex hull used as approximation of laser segment: (a) a roof segment with its convex hull; (b) approximated by convex hull's properties; (c) a wall facade segment intersects a ground segment; (d) approximated by checking intersected convex hulls.

3.3.1 Solid features extraction

Eight semantic feature types are defined and described in the previous chapter. A geometry feature, or a laser segment in current context, can be reasoned with a proof tree to determine a most likely semantic type. Figure 3.8 shows the feature extraction solution of the data set in Figure 3.4 with the highest reliability score. There are six types of building features extracted:

Wall facade Although some doors are detected as walls, this does not influence the wall's contour. Actually, if some segments are roughly coplanar, attached to each other, and have the same feature type, the agent will treat them as a result of over segmentation and hence merge them as one feature.

Roof face Four segments are recognized as roof faces. Two of them are merged.

Protrusion facade A two-facade protrusion is detected.

Dormer facade There are two dormers on this building, while the left dormer facades are not recognized. This is because the solid part of this dormer facade is so small that too few laser points are available to record the geometry reliably.

Window The recognition of windows is not satisfactory. Not all windows are recognized, and even recognized feature segments give incomplete and inaccurate geometry. The reason for the failure is due to the fact that the feature extraction algorithm mainly counts on a relatively good segmentation. Window frames are usually small parts of wall, so terrestrial laser scanning retrieves only a few laser points for windows. When a window is not covered with a curtain, the laser beam will just penetrate the glass, and the pulse will only reflect far behind the window. Insufficient information leads to bad segmentation for windows, and in turn leads to bad window recognition.

Door The most left door segment is a segmentation error (result of the attempt to fit a plane to a curved surface), but is still recognized as a door. If the request for level of detail is high, this curved segment can be fit to a B-spline surface in the geometric reconstruction stage (see Section 4.2), otherwise this segment can be just manually removed.



Figure 3.8: A feature extraction result.

The clause weights of the proof tree are all set to equal in this case because the building is completely scanned and the shape is rather simple. The likelihood thresholds are set to 90 percent for grounds and wall facades and 60 percent for

other features. Such a parameter configuration maintains the optimal balance between the computation cost and recognition accuracy, and works fine for most data sets (see Chapter 7 for more test cases).

3.3.2 Hole-based window extraction

The current feature extraction method is able to recognize solid building features such as wall facades and doors, but doesn't work well for windows, because of the lack of laser points on windows or window frames. However, by observing the extracted wall facades it is obvious that the holes are caused by windows, doors and protrusions on the wall. Even if a window is covered by a curtain, this curtain should be on a different plane than the wall facade and result in a separate segment, and finally also results in a hole on the wall facade. If a TIN is generated from the laser points of wall facade segments, it can be anticipated that solid areas would result in short TIN edges, and holes would result in long TIN edges connecting the points on the holes' boundaries. Based on this hypothesis, we developed an alternative method to extract windows from wall holes.

The main steps are:

Triangulation A TIN for the wall segments is generated first.

Extracting boundary points Long TIN edges appear only at the outer boundary (wall outline) or inner boundary (holes) of a wall. Boundary points are just the end points of the long TIN edges.

Clustering Points belonging to the same hole are grouped together. The clustering algorithm is given as follows:

1. Choose a boundary point A which has no label value yet. Label A with an integer value: i .
2. Find all the long TIN edges which connect to this A. For all of these long edges, determine the other end points: B.
3. Give B the same label value: i , if B is not labeled yet.
4. Make B the new A, iterate step 2) to step 4), until no more unlabeled points can be found.
5. Choose another unlabeled boundary point A, give it a label value: $i+1$, repeat step 2) to step 5), until all boundary points are labeled.

Extracting holes In a TIN mesh, an interior triangle always has three neighbour triangles, while the triangles on the outer boundary only have one or two neighbour triangles. Therefore, we first get all the triangles that have a boundary point as a vertex, and check these triangles on the number of their neighbour triangles. If there are three, then the checked triangle is an interior triangle, and the boundary point is an interior boundary (hole) point. Otherwise the boundary point is an outer boundary (wall outline) point. Figure 3.9(a) and 3.9(b) illustrates the two kinds of boundary points.

Fitting rectangles Assuming windows are rectangular, a minimum bounding rectangle is fitted to each hole cluster, as shown in Figure 3.9(c). We are aware that some window borders are curved, and curve fitting method should be applied instead of rectangle fitting.

Removing holes caused by protrusion and door The holes in Figure 3.9(c) are compared with the recognized protrusion and door segments. If a hole doesn't have overlap with any protrusion or door segment, it is considered as a window hole. Noise holes which have irregular shapes, such as extremely long and narrow, and very small holes are also removed. Figure 3.9(d) gives the result after this step.

This method is not limited to wall facades. It can be applied to extract windows from holes on doors, protrusion facades or dormer facades, too.

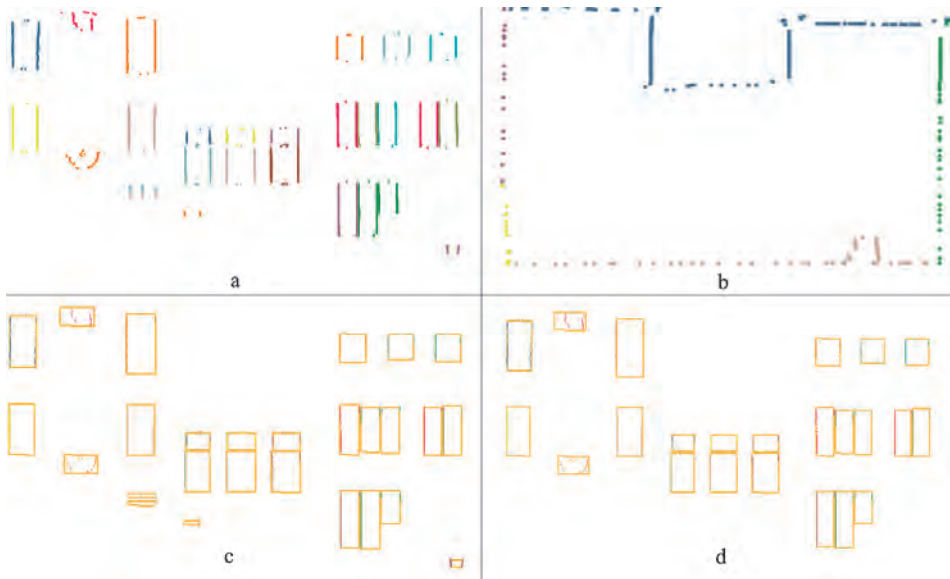


Figure 3.9: Extracting windows from holes in the wall: (a) inner hole points; (b) outer hole (boundary) points; (c) directly fitted rectangles from inner hole points; (d) after filtering out holes of extrusions and doors.

3.4 Discussion

The procedure of extracting semantic features from a terrestrial laser point cloud is presented in this chapter. Although the method works fine with the example data set, the applicability is limited to the building shapes defined in the knowledge

base. For example, the current knowledge base describes doors as “low features inside a wall facade”, but doors can appear in high positions on buildings with open corridors (uncertain knowledge), or inside a protrusion/intrusion facade (incomplete knowledge). The first limitation can be avoided by adjusting the clause weights. The second limitation can only be solved by improving the knowledge base.

The success of feature extraction not only depends on the knowledge base, but also relies on the uniformity and completeness of the laser scanning. Nonuniform distributed laser points can be seriously over-segmented at the sparse regions such that there is no sufficient clue for merging. The building features, especially small ones, are often occluded partially or completely during terrestrial laser scanning. If a feature is completely occluded, it might be recovered later in the outline generation stage by examining the gaps between outlines. If a feature is partially occluded, the calculated geometry attributes and topology relations with other features would be unreliable. This will in turn lead to incorrect feature extractions. Our method works better with the mobile laser scanned (MLS) point clouds than the static scanned TLS data, because the MLS data are much more uniform and completed than the static TLS data (see Chapter 8 for experiments with MLS data). Furthermore, additional data sources such as ground plan or digital imagery should be beneficial to recover the missing information.

Geometric reconstruction

In our approach, geometric reconstruction can be seen as a process of shape fitting plus the knowledge based hypotheses for occluded parts. The flat features are fitted with polygons by least squares fitting, the Quick hull method, or the Hough transformation. The curved features are fitted with a B-spline surface approximation algorithm. Since complete scanning of a building is hardly possible, it is necessary to make certain hypotheses on the side and back parts of a building to compose a solid polyhedron model. Without these hypotheses, the final reconstructed model will be just a group of disconnected polygons which only looks realistic when viewed from the scan position.

4.1 Polygon fitting

The applied polygon fitting algorithms are different according to the size of the feature.

4.1.1 Least squares fitting

For the wall facades, we only generate the upper contour lines using least squares fitting. The outside points of the fitted lines are then projected onto ground level to make a closed wall outline. This procedure has been adopted because for a wall's outline (see Figure 3.9(b) and Figure 4.1), the lower boundary is usually not so easily recoverable from laser points, due to the low laser reflection on windows, occlusions of decorations, advertisement boards, and people. To avoid the noise in the lower parts of building contours, it is assumed that most parts of a wall intersect the ground. Figure 4.2 gives a flowchart of the wall outline generation process. Firstly, we only apply the least squares fitting algorithm to the wall's upper contour points, to determine the wall's upper boundary. Gaps caused by vertical boundaries are filled with vertical line segments. Then, the left and right boundary are generated by projecting the most left and right vertices of the upper outline to the ground plane. The left boundary and right boundary, or projections

of the left wall and right wall on the front view, are two vertical line segments, which accord with the hypothesis that “walls are vertical”. Finally, a line segment is generated on the ground plane which connects the left and right outlines as the bottom boundary of the wall.



Figure 4.1: Wall outline generation (Left up: upper contour points; left down: least squares fitting; right: generated wall outline).

Sometimes, small occlusions on a facade or segmentation errors cause irregular edges on the generated outline. These irregular edges should be removed. Usually, the irregular edges are a consequence of more than two nearby edges. Such short edges are removed from the outline and then result in a gap on the outline. If the left long edge (to the gap) and right long edge belong to the same line, the gap is just filled by connecting a line segment; if the two edges are parallel, a line segment which is perpendicular to the two edges is generated, and the two edges are both extended a bit to reach the perpendicular segment; in other cases, the two edges are just extended or shortened until they intersect at a point to fill the gap (see Table 4.1).

4.1.2 Convex polygon and concave polygon fitting

The shapes and orientations of the roofs and protrusions are more variable than those of walls. So it is necessary to depend more on the laser data than on generic model knowledge. We use the Quick hull algorithm as presented in Barber et al. (1996) to generate convex polygons, or use the Hough transformation to generate concave polygons. Currently convex polygons are fitted to roofs, because most roof planes that are visible from the street side are convex and the Quick hull method is fast. Concave polygons are generated for protrusions to guarantee their accurate outlines. We define the concave polygon fitting algorithm as follows:

1. Generate the TIN for all points on a segment.

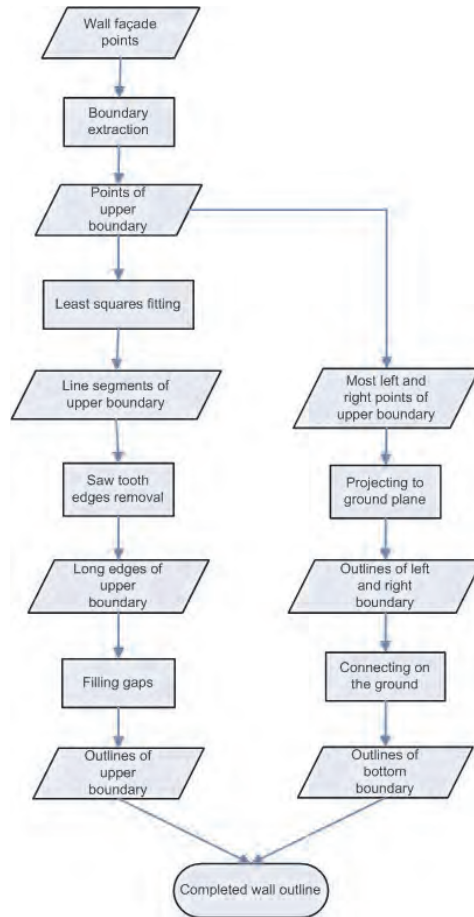


Figure 4.2: Wall outline generation process.

2. Delete long TIN edges, most of which are within the concave parts.
3. Derive the contour points. We start with the point with the largest X coordinate, which is guaranteed to be one of the contour points. We then find all the points which are connected to the starting point by the TIN edges. The second contour point is just the first point among the connected points, which is in counter-clockwise order starting from three o'clock. This search is iterated until the contour is closed.
4. The Hough transformation is applied to the contour points to extract linear features. For any gap between two line segments, a gap line segment is generated between the two line segments. This line is perpendicular to the dominant direction determined by the Hough transformation.
5. The concave polygon is combined by connecting all the line segments.



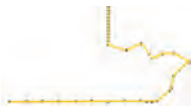
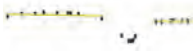


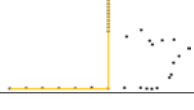
	Left and right edge on same line	Left and right edge parallel	Other cases
Line segments			
After removing irregular edges	...		
After filling gaps	...		

Table 4.1: Handling irregular edges

In an ideal situation, the directly fitted outlines can make up a water tight model. This is seldom the case because there are always parts on building facades that have not been scanned. This makes the fitted outline polygons smaller than reality. These parts are mostly located in the intersection zones between features, for example between roof and wall, between protrusion and wall, and between different faces of the same protrusion. To solve this problem, we replace a feature's outline edges which are close to another feature with the intersection line of the planes that the two features belong to.

4.1.3 Minimum bounding rectangle fitting

The minimum bounding rectangles are fitted to the window holes extracted in 3.3.2. To make the model more realistic, we also intrude the windows inside its supporting wall facade for a short distance. This distance can be calculated from actual laser points, because the window frames provide the intrusion distance of the windows. Even if there are no window frames, a few points are also available from glass, which is sufficient to give the actual offset of the window to its supporting wall. It is also possible to assign an arbitrary intrusion value (5 cm for example), which is definitely good enough for visualization (see Figure 4.3).

The same fitting method is applied to doors, too.

4.2 B-spline surface fitting

Many building features are curved and should not be fitted with polygon. The shape of a curved feature can be cylindrical, conical, spherical or even freeform. Instead of detecting the curved shape first and then apply corresponding fitting algorithm, we adopt the B-spline surface as an uniform mathematical model, and

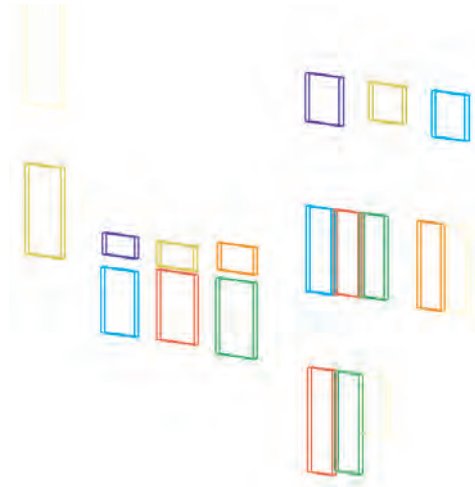


Figure 4.3: Generating window outline.

fit B-spline surfaces to all curved segments. This section first gives a simple introduction to the mathematical model of B-spline¹, then elaborates the surface approximation method which is based on Piegl and Tiller (2000).

4.2.1 The B-spline curve and surface

A B-spline curve of degree p is defined by $n+1$ control points P_0, \dots, P_n and a knot vector of $m+1$ knots:

$$U = \{u_0, u_1, \dots, u_m\}$$

where U is a nondecreasing sequence with $u_i \in [0, 1]$, and n , m and p must satisfy:

$$p \equiv m - n - 1 \quad (4.1)$$

The B-spline parametric curve function is of the form:

$$C_p(u) = \sum_{i=0}^n P_i N_{i,p}(u) \quad (4.2)$$

$N_{i,p}(u)$ is the basis functions of B-splines, defined by:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

¹The reader should refer to Piegl and Tiller (1997) for a solid background of B-spline curves and surfaces.

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad (4.3)$$

A B-spline surface is an expansion of B-spline curves in two directions, with corresponding control points, knot vectors, and univariate B-spline functions. A B-spline surface is defined by:

$$S_{k,l}(u, v) = \sum_{i=0}^n \sum_{j=0}^m N_{i,k}(u) N_{j,l}(v) P_{i,j} \quad (4.4)$$

where the k, l are the orders (degree+1) of the B-spline surface in both directions. The $N_{i,k}(u)$ are the polynomial B-spline basis functions of degree $k - 1$ in the u parameter direction, and $N_{j,l}(v)$ are the basis functions of degree $l - 1$ in the v direction (Piegl and Tiller, 1997).

4.2.2 B-spline surface approximation

There are two main types of fitting method: in **interpolation** the fitted curves or surfaces must pass through the given points, and in **approximation** the curves or surfaces do not necessarily pass through the given points, but only approximately (Piegl and Tiller, 1997). A laser point cloud often contains noise points, and surface interpolation might generate a large number of control points and knots which are unnecessary. Approximation methods are more commonly used in reverse engineering, where the reduction of data is principle and certain level of inaccuracy can be tolerated. The method presented in Piegl and Tiller (2000) is a B-spline surface approximation algorithm which is developed particularly for row points. In laser scanning, the points on building facades often appear in rows, which suits this algorithm well. Sometimes a laser point cloud is combined from multiple single scans, and the row pattern might be ambiguous. In this case, we can define a dominant direction (for example along the map outline), and slice rows of laser points as input to the B-spline surface approximation algorithm.

Suppose there are n rows of point in u direction, with various number of points in each row (v direction). The approximation method is briefly outlined as follows:

1. Approximate a B-spline curve $C_0(v)$ with an initial knot vector to the first row of points. The initial knot vector is selected so that it balances between the numerical stability and the quality of the approximating curve.
2. Pass the knot vector to the rest rows of points for separately curves approximation, producing the v -directional curves $C_i(v), i = 1, \dots, n$. If necessary, add or remove the knots after each curve approximation.
3. Making the curves compatible (same degree and knot vector, same number of control points) by knot insertion and knot refinement. The uniform degree and knot vector of the curves are the corresponding parameters of the B-spline surface in v direction.

4. Since the curves are compatible, columns of points can be selected from the control points of $C_i(v)$ with the same index. Repeat step 1-3 to the columns of points, and the degree, surface control points and the knot vector in u direction are obtained.

4.3 Hypotheses for parts without laser data

A building is seldom completely scanned. The usual case is that only facades on the street side are scanned. To make a solid polyhedron model instead of just a grouping of facade polygons, we have to make hypotheses on the occluded areas. In particular, hypotheses on the following parts have been made:

Left, right, top, bottom and back sides of a building Right now we simply perform a translation sweep of the front face backward for a certain distance to extrude a solid building model. The sides are generated automatically by sweeping. The sweeping trajectory is a straight line segment which is automatically extracted from the 2D ground plan of this building.

The occluded areas on front facades These areas are usually the side faces, top face, and bottom face of the protrusions. Our solution is to project the outlines of a protrusion to the supporting surface of this protrusion. For a wall protrusion and intrusion this supporting surface is a wall (see Figure 4.4 left and bottom) and for a dormer this supporting surface is a roof (see Figure 4.4 right). Outlines of the occluded areas are generated during the projection.

Roof When a building is tall, the laser point density can be very low on roofs. The low point density becomes sufficient for reliable outline generation. Here knowledge can be integrated to avoid unlikely roof outlines. Two kinds of roof shapes are hypothesized: parallelogram and triangle. We first find whether there is a highest horizontal edge (ridge line) from the roof's convex hull. If so then a parallelogram roof is determined, otherwise a triangle roof. Then the left and right edges are determined by connecting the ridge line or top point (for triangle shape) with the wall outline edge which supports this roof.

4.4 Results and Discussion

4.4.1 Flat surfaces

Figure 4.5 shows the reconstructed polyhedron model of the point cloud in Figure 3.4. The reconstruction of wall facades, doors, windows and roof faces are fully automatic, because there are a sufficient number of laser points from these large

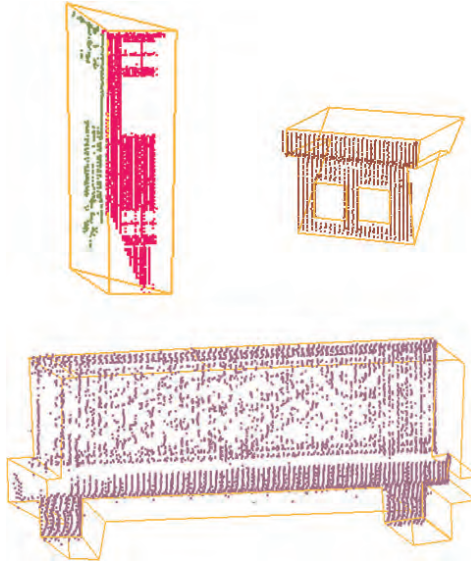


Figure 4.4: Hypotheses of the occluded areas

features to recover all of the outlines. The reconstruction of protrusions is semi-automatic, because the method cannot distinguish some noise segments in front of the wall facade. Manual interventions are given to select the correct protrusion facade segments from all potential segments which are provided by the knowledge based feature extraction.

The original laser point cloud is visualized together with the model to check the reconstruction quality. Errors occur at the following parts (see the corresponding numbers in Figure 4.5):

1. The reconstructed dormer is much smaller than its actual size. This is because the dormer frames are too narrow for surface growing algorithm to grow throughout the dormer. This results into many small segments on the dormer. Only the largest segment yields a sufficient reliability score to be considered as a dormer facade segment, so the reconstructed dormer is only based on this segment.
2. There is a round corner on the bottom of this protrusion which is not recognized.
3. The window outline is smaller than its actual size, because a curtain covers half of this window and the curtain is near the wall plane. As a consequence, no hole is detected from that part of the window.
4. There is a 10 cm offset between a vertical wall outline edge and its actual position. The upper boundary of a wall is calculated from the highest points

on a left-right scan line. The eaves on a higher wall will make this wall wider, and its neighbouring wall more narrow.

5. There is also an offset between a roof outline edge and its actual position. This is also caused by an inaccurate wall upper boundary. After the left and right edge of a roof's outline is determined by connecting the ridge line or top point with a wall outline edge, the roof outline might be wider or more narrow if the wall outline edge is longer or shorter.



Figure 4.5: Reconstructed polyhedron model of a building facade. Errors occur at: 1. dormer; 2. round corner; 3. window; 4. wall boundary; 5. roof boundary.

4.4.2 Curved surfaces

Figure 4.6 gives an example of the B-spline surface approximation. The data set contains two curved wall facades and a small flat plane connecting the two wall facades. After segmentation by the smooth surface method, two curved segments (yellow and pink) and two flat segments are distinguished. The flat segments are fitted with concave outline polygons as described in the previous section, and the curved segments are fitted using the B-spline approximation method. As the point set is combined from two scans, we first need to sort or sample the curved segments so that the points are organized by rows. This is achieved by slicing each

curved segment with horizontal planes which are equally distributed with 5 cm spacing from bottom to top. Suppose the u direction is vertical and v direction is horizontally along the map outline, with the error tolerance set to 10 cm (distance between the fitted surface and the input points not exceeding 10 cm), the yellow segment is fitted to a B-spline surface comprised of 2 by 9 control points network, and the pink segment is fitted to a B-spline surface comprised of 2 by 6 control points network. Both B-spline surfaces are degree 3 (cubic polynomial) in v direction. Although visually it seems to be a cylindroid surface and a quadratic degree should be enough, the noisy laser points introduced extra degree and knots during the approximation. The degree of the surfaces in u direction is 1 (linear polynomial), which is correct since the wall facade is not curved in a vertical direction. The current tolerance value is suitable to approximate satisfactory surfaces for this data set. Stricter error tolerance would result in more “exact” surfaces to the laser points, while more control points and even higher degrees would be produced as well.

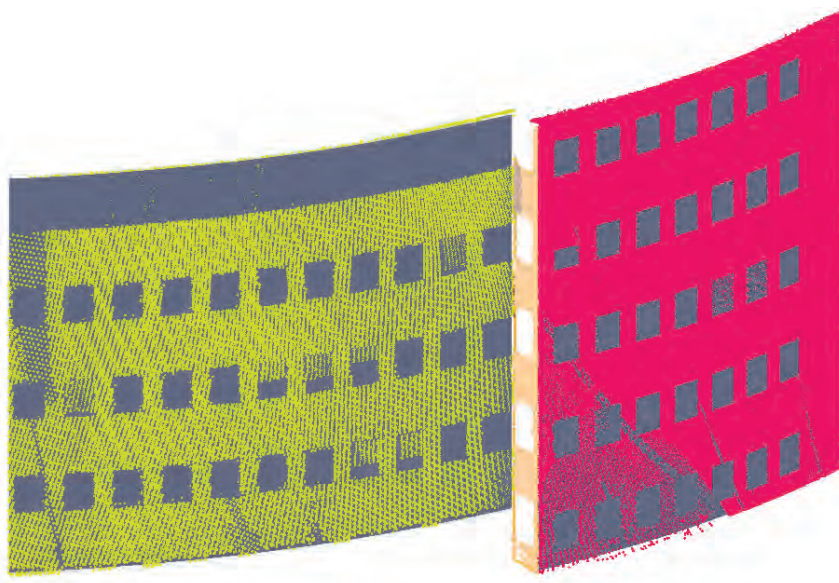


Figure 4.6: Approximate curved laser segments with B-spline surfaces.

As mentioned in the previous chapter, curved segments need to be manually selected from flat segments and proceeded with the reconstruction steps for curved geometry. In fact, flat segments can also be approximated with B-spline surfaces, which will be degree 1 in both u and v direction. For data storage consideration, representing a flat segment with a B-spline surface concerns only a few more parameters than polygons. However, the geometric operations (intersection, area, distance, etc.) are rather complex and computational heavy for B-splines to B-splines and B-spline to polygons, because of the iteration nature of the B-spline basis functions. The local modification of a B-spline surface is also not easy. Fur-

thermore, quite a number of terminal applications (such as ArcGIS and Oracle Spatial) have not given support to B-spline representations, so that tessellation has to be applied in addition. Therefore, the B-spline surfaces should only be used when the curvature of a surface is too complex to be outlined with polygons or when the requirement for reconstruction details is high.

Model refinement with imagery

This chapter presents the integration of imagery to achieve more accurate reconstruction. Image based building reconstruction has been researched for many years. From multiple 2D images captured from different positions, 3D coordinates of the image features (lines for example) can be calculated. Although acquisition of images is cheap and easy, the difficulties of image understanding make it still difficult to automate the reconstruction using only images. Laser altimetry has been used more and more in recent years for automated building reconstruction. This can be explained by the explicit and accurate 3D information provided by laser point clouds. Researches (Vosselman, 2002; Frueh et al., 2004; Brenner, 2005) suggest that the laser data and images are complementary to each other, and efficient integration of the two data types will lead to a more accurate and reliable extraction of three dimensional features.

In the previous chapters we presented a knowledge based building facade reconstruction approach, which extracts semantic facade features from terrestrial laser point clouds and combines the feature polygons to water-tight polyhedron models. Some modelling errors still exist, and some of them can be hardly corrected by further exploiting the laser data. Therefore, we develop an image based refinement method which uses strong line features extracted from images to improve the building facade models generated from only terrestrial laser points. The refinement not only fixes the models' geometry errors, but also solves inconsistencies between laser and image data (for example when they are not acquired at the same time), so that a more accurate texturing can be achieved.

This chapter is organized as follows: Section 5.1 gives an overview of the refinement method. Section 5.2 explains the registration algorithms for referencing the laser data space and the image space. Section 5.3 elaborates the image processing algorithms used for significant line extraction and the matching and refinement strategies. Experiments on three test cases are provided in Section 5.4. Finally, some conclusions and outlooks are drawn in the final section.

5.1 Method overview

A building facade model may contain various errors. For a model reconstructed from terrestrial laser points, the model edges may have certain offset with their actual positions. These errors are caused by gaps in laser points and the limitations of laser data based reconstruction algorithms. Edges are delineated accurately in images. After registering to the model space, image lines can provide excellent reference from which the model edge errors can be fixed. Another necessity of this refinement is to accurately reference the image space to the laser space and in turn obtain better texturing results.

Before starting the refinement, a 2D image needs to be referenced to the 3D model space, a problem often referred as spatial resection in photogrammetry. We use the standard resection solution of collinearity equations, which requires a minimum of three image points with their coordinates in model space. To find significant line features from an image, we first detect edges using the Canny algorithm (Canny, 1986), then apply the Hough transform (Hough, 1962) to further extract strong line features from edges. Then model edges are projected to the image space and matched with the image lines. The best match is determined by the geometric properties of candidates and the geometric relations between candidates and the model edge. Finally, each model edge with successful matching is projected to the matched image line accordingly, and model edges without any matching are also adjusted to maintain a reasonable shape. Figure 5.1 gives a flowchart of the refinement process.

5.2 Registration

In this section, a “laser points to image” registration process is elaborated which aims at calculating accurate exterior orientations of images. This process (see Figure 5.2), also referred as the spatial resection, is necessary before information from image and laser points can be fused for building facade reconstruction. The key to the spatial resection problem is to choose enough tie pairs between laser points and image pixels to form collinearity equations. Although automated corresponding selection via Scale-invariant Feature Transform (SIFT, Lowe (2004)) matching between laser range images and optical images are reported (Barnea and Filin, 2008), we find that satisfactory results are difficult to achieve in practice. The laser range images are coloured by the reflectance strength of the laser beams, while the colors of optical images are determined by the strength of visible lights, shadows, and the own colour of the objects. The two kinds of light have rather different natures (compositions, wavelength, etc.), and their exposure directions are also different. These incompatibilities make the automated feature matching difficult to accomplish between range images and optical images. Instead, the tie pairs are manually selected in our approach to provide inputs for the spatial resection of a single image. When there are multiple images available for the same facade,

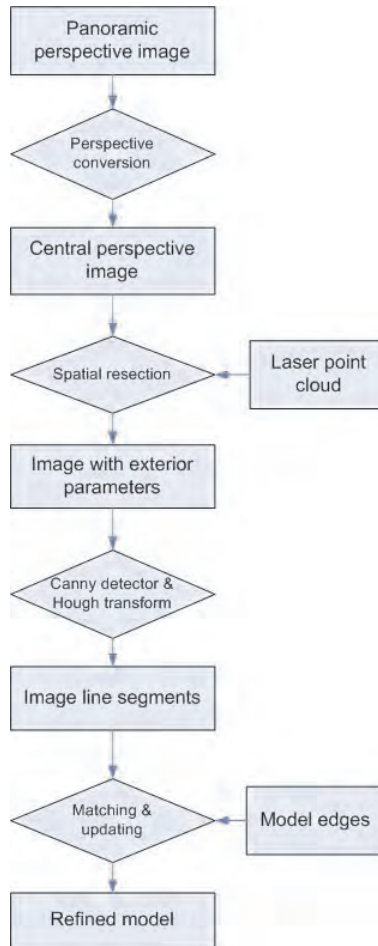


Figure 5.1: Model refinement process.

we only undertake the spatial resection for the central image, and then estimate the homography (also referred as the projective transformation) matrices between the other images to the central image. The exterior orientations of each image can be indirectly determined by multiplying the homography transformation with the orientation of the central image.

5.2.1 Perspective Conversion

A kind of panoramic images called Cyclorama (van den Heuvel et al., 2007) is used in this research. Two fisheye cameras with a field of view of 185 degrees each are turned 180 degrees against each other, and a full sphere image (Cyclorama) is combined from the two images. The organization of a Cyclorama image is

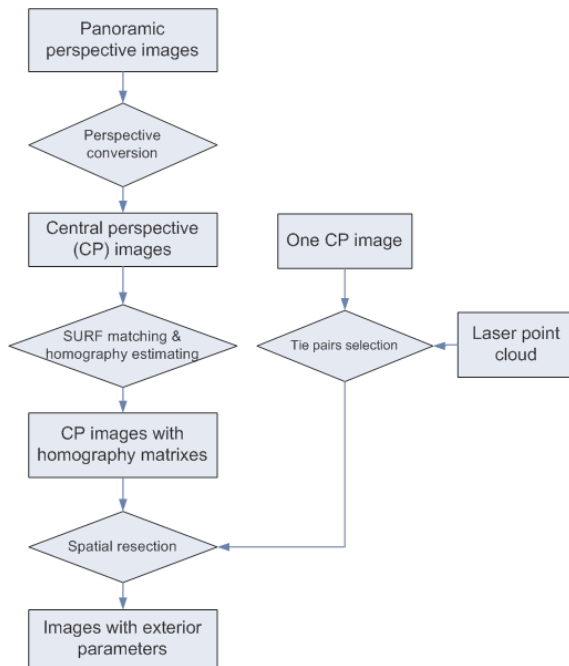


Figure 5.2: Registration between laser points and images.

illustrated in Figure 5.3. Each Cyclorama is stored with 4,800 by 2,400 pixels (scalable), corresponding to 360 degree in the horizontal direction and 180 degrees in the vertical direction. Thus, in both directions the angular resolution is 0.075 degree per pixel. With the integrated GPS and IMU devices, all Cyclorama images are provided with north direction aligned at $x=2,400$ and horizontal plane aligned at $y=1,200$. The acquisition positions of Cycloramas are also provided from GPS, but they are not very accurate.

In order to be feasible for facade modelling and texturing, the Cyclorama images first need to be converted to the central perspective. The equiangular projection of the fisheye camera model is described in Schneider and Maas (2003). The projection from panoramic perspective to central projective can be understood as projecting a panoramic sphere part to an assumed plane. Firstly, two lines are created by connecting the image acquisition point (perspective center) with the most left and most right vertices of the initial polyhedron model. The angle of the two lines with the north direction derive the longitude boundaries of the region of interest (ROI). In practice it is necessary to widen the ROI to both left and right by a few pixels, because of the errors of the perspective center may cause miss selection of some desired region. Suppose the GPS error is e meters, the distance between the acquisition position and a building facade is d meters, and the angular resolution is r degree/pixel. The offset o of a vertical building edge is between

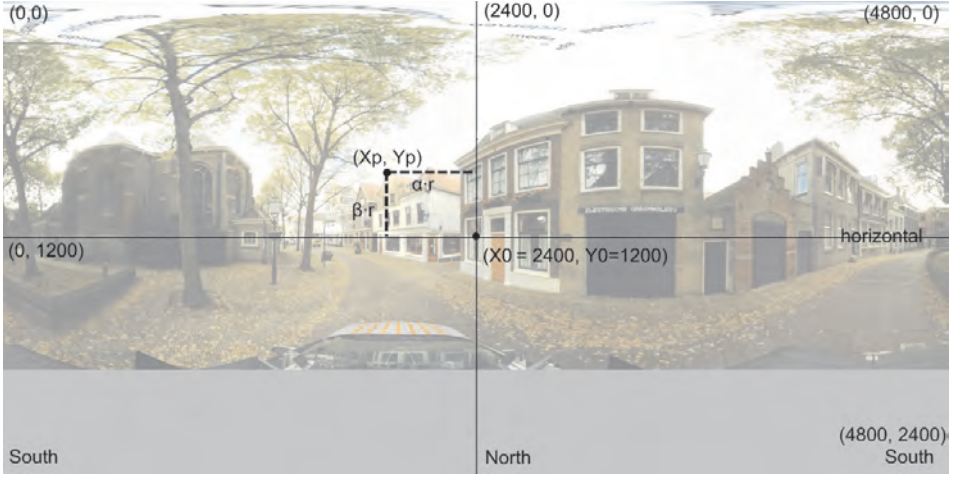


Figure 5.3: The organization of a Cyclorama (α and β are the longitude and latitude of a pixel on the panoramic sphere; r is the angular resolution)

$\tan(e/d)/r$ and $-\tan(e/d)/r$ degrees. The ROI need to be widened by o pixels to both left and right. The principal point is set on the sphere equator, with middle longitude of the two boundaries. Assuming the perspective centers coincide in both perspectives, the pixels inside the ROI are converted from panoramic perspective to central perspective according to the following equations:

$$\alpha = \frac{X_p - X_0}{r} \quad (5.1)$$

$$\beta = \frac{Y_p - Y_0}{r} \quad (5.2)$$

$$\tan \alpha = \frac{x_c - x_0}{f} \quad (5.3)$$

$$\tan \beta = \frac{(y_c - y_0) \times \cos \alpha}{f} \quad (5.4)$$

where (X_p, Y_p) is the pixel coordinate in panoramic perspective; (X_0, Y_0) is the principle point's coordinate in panoramic perspective, usually at the center of the panoramic image; (x_c, y_c) is the pixel coordinate in central perspective; (x_0, y_0) is the principle point's coordinate in central perspective, usually at the center of the central perspective image; r is the angular resolution; α and β represent the longitude and latitude of the pixel on the panoramic sphere; f is the distance of the panoramic sphere center to the assumed plane, can also be seen as the focal length of the converted central perspective image. With equation 5.1 to 5.4 the unique relation between (X_p, Y_p) and (x_c, y_c) can be determined. The perspective conversion model is illustrated is Figure 5.4.

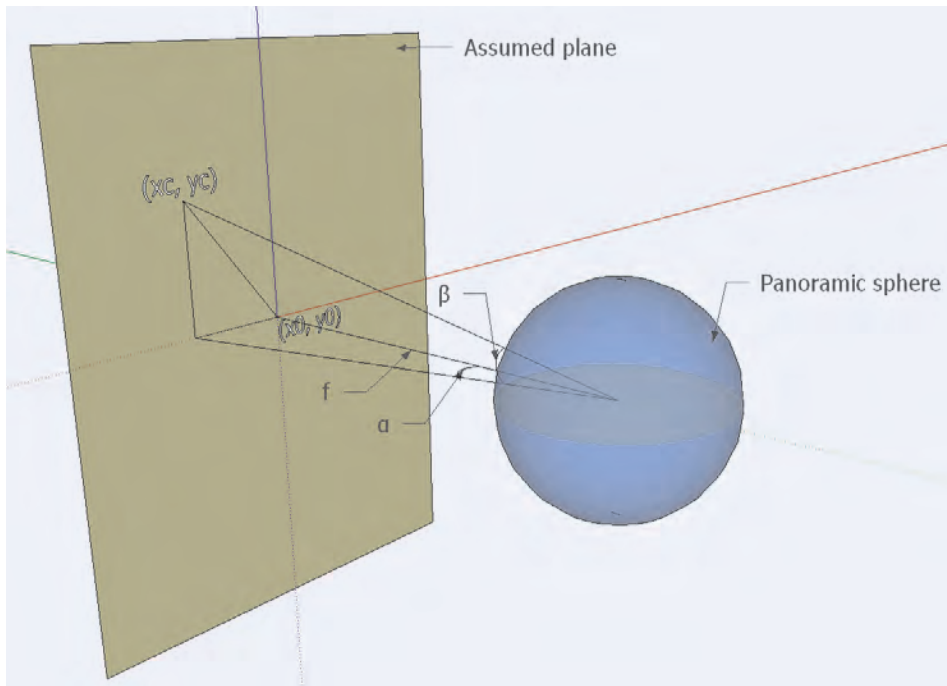


Figure 5.4: Conversion from panoramic perspective to central perspective

5.2.2 Spatial Resection

In order to get a unique solution for the six unknown exterior orientation parameters, at least observations of four image control points should be available to form eight collinearity equations. Figure 5.5 illustrates the interface for selecting tie points from a laser point cloud and an image. In the implementation it is required to select at least four tie pairs, with one pair for error checking. If more than four pairs are selected, a least square adjustment is performed to obtain better results.

5.2.3 Relative Orientation

The Cyclorama images are systematically acquired with an interval of a few meters, so a building facade is usually visible from multiple Cyclorama images, and it is possible to choose images with the best visibility for different building parts. This specialization does not only make sure the image resolution is high enough to provide accurate edge positions, but also guarantees the optimal texturing effects (see Chapter 6 for details). It is time-consuming and unnecessary to manually select tie points from all images for spatial resection. There are a number of robust feature detection algorithms from image processing field, which can extract sufficient image features from two images for homography model estimation. In

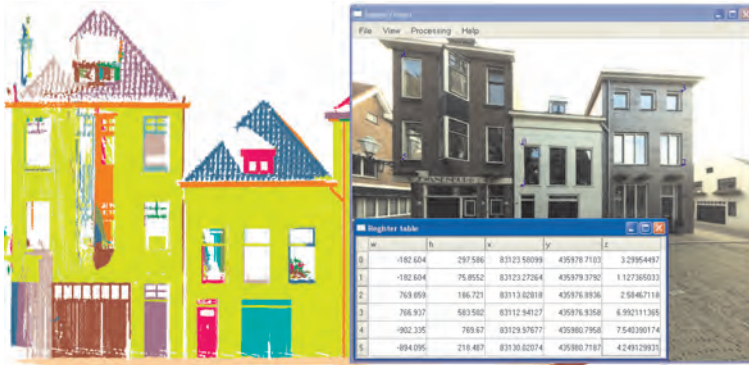


Figure 5.5: Selecting tie points for spatial resection.

the spatial resection explained in 5.2.2, the manually selected laser points are tied with point features in only one image. However, occurrences of the corresponding points in other images can be automatically located if the homography models are known. The required tie pairs are therefore generated for each image for spatial resections.

A scale- and rotation-invariant detector and descriptor called Speed-Up Robust Features (SURF) is used because it can extract satisfactory correspondences between different perspectives, and the processing speed is several times faster than SIFT. A detailed explanation of SURF can be found in Bay et al. (2008). The extracted SURF from two images are compared according to same Laplacian signs and minimal sum of Haar wavelet responses. Then homography matrix between the two images is estimated by applying RANSAC (RANDOM SAMPLE CONSENSUS, Fischler and Bolles (1981)) to the SURF pairs. Figure 5.6 shows the linked SURF pairs (red lines) between two perspectives of a building facade and the homographic transformed viewing plane (black quadrangle). Each image pair with neighbouring acquisition positions is processed with the above procedure to calculate their homography matrixes. Figure 5.7 shows the projections of four laser points in different image perspectives, and the numbers indicate the images' acquisition positions from right to left. The four tie points are manually selected from the image No.5 in Figure 5.7, and the marks in other images are automatically plotted by multiplying homography matrixes to the tie points' image coordinates in image No.5. Sufficient collinearity equations can be formed for each image using the same 3D object coordinates and different image coordinates, and each exterior orientation is therefore calculated.



Figure 5.6: Homography estimation using extracted SURF in two images (red lines: linked SURF pairs; black quadrangle: homographic transformed viewing plane of the upper image).



Figure 5.7: Locating occurrences of the same laser points in different image perspectives.

5.3 The model refinement

If the model reconstruction and the spatial resection are both accurate, every model edge should coincide with a significant image line after projecting to the image. But this is seldom the case in practice. Beside the limitations of modelling

methods, the inconsistencies can also be caused by inaccurate tie points selection and spatial resection, or the changing of the environment between laser scanning and image acquisition. Even a very small inconsistency between model space and image space can lead to a few pixels' texture offset at the model boundary, and in turn produce a poor texturing result. A "tuning" between images and model edges should be processed to remove these inconsistencies.

5.3.1 Extraction of Significant Lines from Images

The Canny edge detector algorithm (Canny, 1986) is used for extracting initial line features from images (see Figure 5.8(a) and Figure 5.8(b)). Here two threshold parameters should be specified for edge linking and finding initial segments of strong edges. Thresholds that are set too high can miss important information. On the other hand, thresholds that are set too low will falsely identify irrelevant information as important. It is difficult to give a generic threshold that works well on all images. In addition to the conventional Canny algorithm, a histogram analysis is made on the image gradients in order to adaptively specify the threshold values. However, factors such as illumination, material, and occlusions still result in many irrelevant edges. In the other hand, some desired edges may not be extracted due to the nature of images. For example, outlines of a wall with very similar colour as the surrounding environment will not be detected. Outlines inside shadow areas can hardly be extracted either.

Strong line features are further extracted from Canny edges by Hough transformation (see Figure 5.8(c)). Because of the unpredicted number of edges resulted from the previous step, a lot of irrelevant Hough line segments may also be generated. To minimize the number of these noise lines, instead of adjusting the thresholds of Hough transformation, all Hough line segments are sorted according to their length, and only a certain number of long ones are kept. This is based on the assumption that building outlines are more or less the most significant edges in an image. The limitations of this assumption are already anticipated before applying it into practice. For example, large and vivid patterns on a wall's surface can result in more significant line features than the wall edges.

5.3.2 Matching Model Edges with Image Lines

To match model edges and the image lines for refinement, both should be located either in the 3D model space or the 2D image space. Matching in image space is chosen because projecting geometry from 3D to 2D is much easier than determining the 3D coordinates of an image pixel. With the calculated exterior orientation parameters from spatial resection and the focal length, model edges can be projected to the image space according to the collinearity equations (see the blue lines in Figure 5.8(d)).

Assuming a relatively accurate exterior orientation and the focal length are avail-

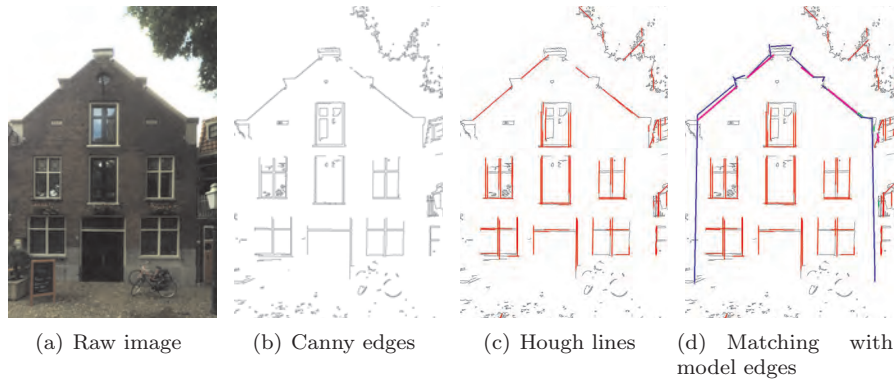


Figure 5.8: Extracting significant lines from an image (black: Canny edges; red: Hough lines; blue: projections of model edges; green: candidate matches; purple: best matches).

able, the best matched image Hough line for a model edge is determined in two stages:

1. Candidates of best matching image lines are filtered by their parallelism and distance with the model edge (see the green lines in Figure 5.8(d)). In other words, the angle between a candidate with the model edges should be smaller than a threshold (5 degrees for example), and their distance should also be smaller than a threshold (half a meter for example). Note the actual distance threshold is in pixel, which are also "projected" from a 3D distance on the wall plane. If the exterior orientation and focal length are perfect, most model edges should coincide very well with a strong image line. However, in practice there may be a small offset and angle between a model edge and its corresponding image line. The optimal angle and distance threshold value are dependent on the quality of the exterior and interior orientations.
2. A best match is chosen from all candidates according to either the collinearity of the candidates or the candidate's length (see the purple lines in Figure 5.8(d)). It is a common case that a strong line is split to multiple parts by occlusions or shadows. If a number of Hough line segments belong to a same line, this line is set as the best match. If not, the longest candidate is just chosen as the best match.

No spatial index is established in the image space to improve the comparison efficiency, because the search space is already localized to a single building facade, which includes only dozens of edges and Hough lines.

A limitation of this matching method is that it can hardly determine the correct corresponding edge if too many similar line features are within searching range. Simply comparing the geometry properties of position, direction and length is not sufficient in this case. For example, eaves often result in many significant lines and

they are all parallel and close to the wall's upper boundary edges. These eave lines can be distinguished if the eave is also reconstructed and included in the facade model, but ambiguity caused by pure colour pattern is still difficult to solve.

5.3.3 Refinement Strategy

After matching, most model edges should be associated with a best matched image line. These model edges are updated by projecting to their best matched image line. There are some model edges which don't match any image lines. If no change is made to an edge with its previous or next edge changed, strange shapes like sharp corners and self-intersections may be generated. Therefore, interpolations of the angle and distance change from the previous and next edges, are applied to the edges without matched image lines. With these refinement strategies, an original model is updated to be consistent with the geometry extracted from images, and the model's geometry validity and general shape are also maintained.

Finally, the refined model edges in image space need to be transferred back to the model space. We assume the model edges are only moved on their original 3D planes. The collinearity equations are used together with the mathematical equation of the 3D plane to calculate the new 3D positions of all the modified model vertices.

5.4 Test cases

In this section, three data sets are experimented with the presented refinement method. The building models are produced with the reconstruction approach presented in Chapter 3 and 4. All the images are originally provided as Cycloramas. The central perspective conversion and exterior orientation calculation follow the processes explained in Section 5.2.

5.4.1 The restaurant house

The inconsistencies between the model edges and image lines in Figure 5.9 are mainly due to inaccurate exterior orientation of the image. It is difficult to pick an image point accurately by manual operation. Picking the corresponding point in a laser point cloud is also a difficult job. Automated texturing of building facade models is desired in the context of our research. The quality of the exterior orientation is a key issue to the texturing effect. Even a minor inaccuracy in the exterior orientation parameters can lead to a poor texture result, as shown in Figure 5.10(a). Applying our refinement method, several model edges are linked with their matched image lines (see Figure 5.9(b)), and are updated accordingly. The texture result is significantly improved as shown in Figure 5.10(b), with the sky's background colour removed. However, the middle top part of the facade

model is still not refined, because this image part is too blurred to output a Hough line.

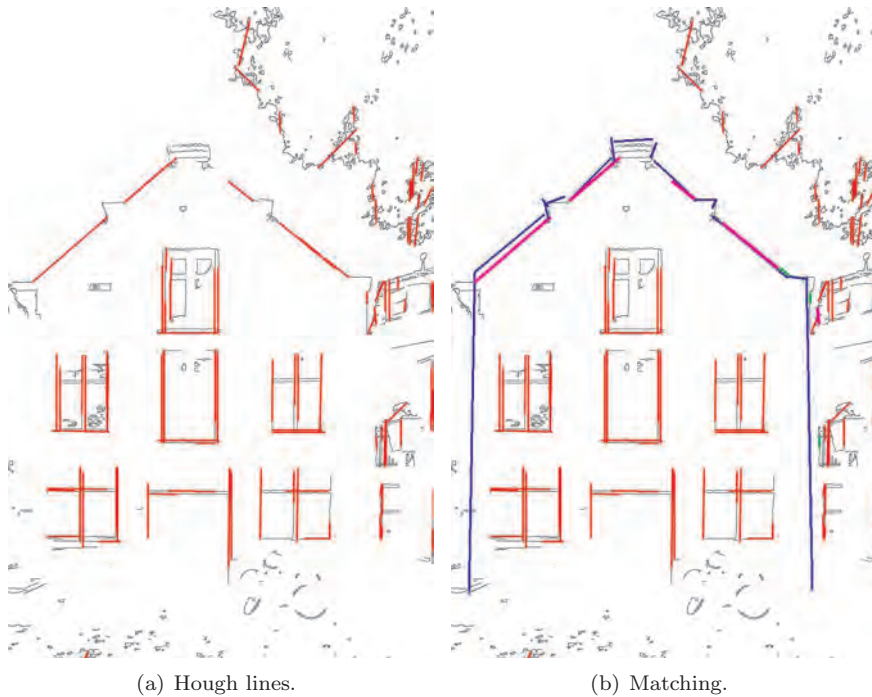


Figure 5.9: Matching model edges with image lines for refining the restaurant house's model.

5.4.2 The town hall

The upper boundary of the town hall in Figure 5.11(a) contains a lot of tiny details, which are well recorded by laser scanning and modelled as sawtooth edges in the building facade model. Instead of adjusting the outline generation parameters in the reconstruction stage, we can also use the presented image based refinement to smooth the model outline. Figure 5.11(b) shows the matching step. The model's upper edges are successfully matched to the strong lines, which actually come from the eave. In this example the wall's upper boundary is not detected in the image, because it is occluded from sunlight by the eaves when the image was taken.

The left boundary of the building model is modelled correctly from laser points by intersecting two large wall planes. However, in practice it is matched to a strong contrast caused by a water pipe on the wall (see Figure 5.11(b)). This again, reveals the limitations of this refinement method.



Figure 5.10: Comparison of textured restaurant house model before and after refinement.

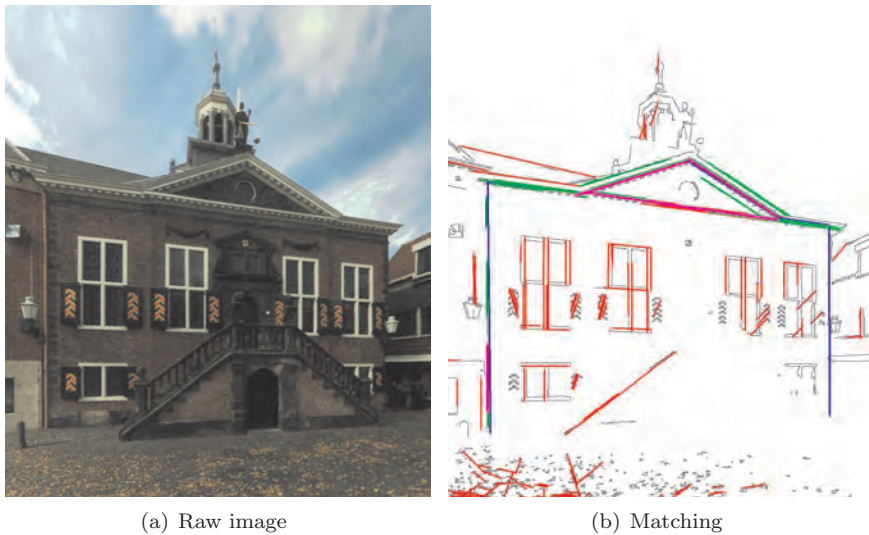


Figure 5.11: Matching model edges with image lines for refining the town hall's model

5.4.3 The wall with high windows

In this example, the refinement is applied to improve the windows extracted from the holes from laser points of a wall facade (see Section 3.3.2 and 4.1.3). The

contrast of a window and its surrounding wall are usually rather obvious in optical data. Strong line features are frequently found at the windows boundaries and frames, and can be used to refine the information from the laser altimetry. Figure 5.12(b) shows the window rectangles extracted from laser points, which contain a lot of errors due to limitation of segmentation and modelling algorithms. In Figure 5.12(a) we apply the same matching stretchy for refinement purpose, and the final result is shown in Figure 5.12(c). Most windows' boundaries are well corrected according to the image lines. The second left window in the upper row is not improved, because the difference between the modelled shape and the actual shape is too large to correlate them. There are some remaining errors, such as the first, third and sixth window (from left to right) in the lower row. This is because the parameters of Hough transform are too strict to generate any candidate line.

The textured final model after manual adjustment is shown in Figure 5.13. Without the refinement, 42 vertices need to be manually adjusted. Only 10 vertices need to be adjusted after the refinement. Note that the window rectangles are intruded inside the wall plane, and the intrusion offset is derived from laser points reflected from window glass and window frames.



(a) Matching.



(b) Before refinement



(c) After refinement

Figure 5.12: Matching and refining window boundaries.



Figure 5.13: A textured building facade model with intruded windows.

5.4.4 Summary

The effectiveness as well as limitations of our refinement method are examined through the three test cases. We realize that the refining effect relies on the following prerequisites:

- Accurate exterior and interior orientations. In particular, the selected tie points for spatial resection should be sufficient (four or more), and should be equally distributed in both horizontal and vertical directions in order to minimize the computation error.
- No large occlusions in front of the building facade.
- Stronger contrast by building edges than optical factors (illumination, colour pattern, etc.).

Some limitations of the current refinement method have been located at:

- It cannot solve ambiguities caused by multiple lines with similar geometry properties.

- It does not distinguish whether a model-image inconsistency is caused by reconstruction errors or inaccurate exterior orientation yet.

Knowledge based reasoning of the image information is the key to the first problem. The current matching strategy is rather local. Experiments show that the offset direction between the model edges and their matched image lines are mostly same, which is obviously caused by inaccurate exterior orientation. A globe matching process (RANSAC over offsets for example) should be able to estimate the correct exterior orientations.

5.5 Conclusions and outlook

In this chapter we presented a model refinement method, which used the lines extracted from close-range images to improve building models reconstructed from terrestrial laser point clouds. With the refinement, several modelling errors caused by either gaps in laser data or reconstruction algorithm, are corrected with image information. Texturing is also improved after the refinement.

Nowadays it is more and more common for acquisition platforms to acquire laser data and optical data simultaneously. Line extraction from images is very accurate, while laser points are more suitable to extract planar features. Efficient fusing of laser points and image naturally avoids many barriers for building reconstruction from either sides. The attempt through our refinement method shows a promising future for automated building reconstruction by fusing laser altimetry and optical methods.

Two possible directions for future work: knowledge based image reasoning and global matching, have been suggested earlier. Besides, nowadays it becomes popular to mount cameras and laser scanners on mobile platforms and obtain the position and pose information via GPS and IMU. Unless both optical data and laser data are captured simultaneously with an integrated system, the relative orientation between the two data will not be very accurate. If we match image lines with model edges from laser points (similar to the presented methods in this chapter), there should be enough control points to align the images accurately to the laser points.

Texture mapping

In this chapter, the textures are automatically selected from images and mapped to the building model surfaces. In computer graphics, **texture mapping** (sometimes referred as texturing) is a method for adding detail, surface texture, or colour to a computer-generated graphic model. Texture mapping makes significant improvement to the visualization of a virtual scene. In addition to the geometric information, correctly textured building models will provide extra information in colour which is easily understandable.

Assigning the correct textures is probably the most time-consuming step during a manual reconstruction process. To texture one face of a model, the operators first have to execute the following operations:

Selection Browse through a number of candidate texture images to choose one with a satisfactory visibility, and then select corresponding texture parts for all model faces.

Rectification Remove the perspective effect by stretching and scaling.

Mapping Assign each vertex of the model face to a corresponding pixel on the texture image.

Since this procedure has to be repeated for all model faces, the texturing procedure would be extremely slow when a model contains many faces to be textured. In the context of this research, sufficient spatial information: the model geometries and the camera parameters, have been generated and can be used to automate the texture operations. Specifically, the image which is captured directly in front of a polygon should be selected to texture this polygon; the rectification is automated solved since the camera parameters are known; the relation between the model space and image space are given by the camera parameters. Sometimes the model geometry and camera parameters are not very accurate and may lead to a small degree of inconsistency between the model and the image. The refinement operations presented in the previous chapter can be applied to solve such inconsistencies

and improve texturing. At the algorithm level, a texture mapping procedure consists of four steps (Shreiner et al., 2005): (1)calculating the texture coordinate for each vertex; (2)transformation of the textures to fit the textured geometry; (3)maintaining the perspective mapping between texture space and object space; and (4)antialiasing. The last three steps have been handled by standard graphic APIs such as OpenGL and DirectX, and all we should provide are the geometry which should be textured, the texture images, and the texture coordinates of the vertices.

This chapter concentrates on two issues of texture mapping: the strategies to select textures with the optimal visibility, and the calculation of texture coordinates. The proposed methods are based on OpenGL and implemented with OpenGL. Optimal texture selection is discussed in Section 6.1. Generally, the images which are captured in front of a building feature should be used for texturing this feature. When a background object is occluded by a foreground object, textures for the background will be selected alternatively from oblique views. 6.2 presents the derivation of texture coordinates from model coordinates and camera parameters. Some textured models are shown in 6.3 to demonstrate the applicability of the texturing methods, as well as their advantages and drawbacks.

6.1 Selecting texture images

When there is no optical data available for the complete or part of modelling scene, the reconstructed models can be textured by predefined images or just colours. As presented in the previous chapters, the polygons of a building model are modeled from laser segments with semantic meanings. It is possible to texture the polygons with pre-defined images which are associated with the semantic types. For example, all polygons of wall feature can be textured with a predefined brick image; roof feature can be textured with a predefined tile image.

When images or videos are available, and the exterior orientation of each image or video frame is known, colour information of any facade region can be extracted from a corresponded image region. When multiple images are available for a building facade, there will also be multiple corresponding image regions for texturing. If textures are only selected from one image, factors like resolution, illumination and occlusion can easily lead to poor texturing. For example, the quality of image areas which are far away from the perspective center can be much worse than the central area. Occluded textures are often selected if the angle between the viewing direction and principle direction is large. Selecting textures with the best visibility from multiple candidate images will make the final model much more realistic.

6.1.1 Optimal image selection

The general assumption for optimal image selection is that the best texture image T for a polygon should be taken directly in front of this polygon. Specifically, there are two considerations for picking an appropriate texture image:

- The acquisition position of T should not be too far away from the polygon, so that the resolution is high enough for clear texturing.
- If L represents a 3D line decided by the centroid and the normal direction of the polygon, the principle direction of T should more or less coincide with L .

Taking the two considerations into account, at first all available images are filtered by their distances to the target building facade. Only images taken within a certain distance with the building center are kept as texture candidates. Then the L of each model polygon and the angle θ between L and the texture candidates' principle directions are computed. The image with minimum θ to a polygon is assigned to texture this polygon.

The optimal texture selection strategy as explained above is illustrated in Figure 6.1. Only the camera positions which are within an arbitrary distance from the building facade are kept as texture candidates. After the operations mentioned above, it is decided that the wall's texture should be selected from the image of camera position 3. It is best to select texture from camera 2 for window 1, and camera 4 for window 2. Face 1 of the protrusion can be textured by the image of either camera 1 or camera 2, while the other face should be textured by images of either camera 3 or camera 4. The dormer should be textured by the image of camera 3.

6.1.2 Occlusion removal

Images of building facades often include occlusions such as trees, wall protrusions and people walking by (see Figure 6.2). If the occlusions remain on the background textures, the final visualization effect can be very poor, especially when viewing from a different direction from the image capturing direction.

If the positions of the occlusions are unknown, a voting process can be adopted based on the assumption that occurrence of the foreground colour should be less than the background colour. For each pixel in the background region, colours of the corresponding pixels in all other images are extracted to form a colour histogram, and the most occurrence is believed to be the background colour. This method has two main drawbacks: first, the assumption is not necessarily true; second, the computational demand is quite heavy.

In the context of this research, positions of perspective centers, foreground and background are all known, so that the occluded texture regions can be located.

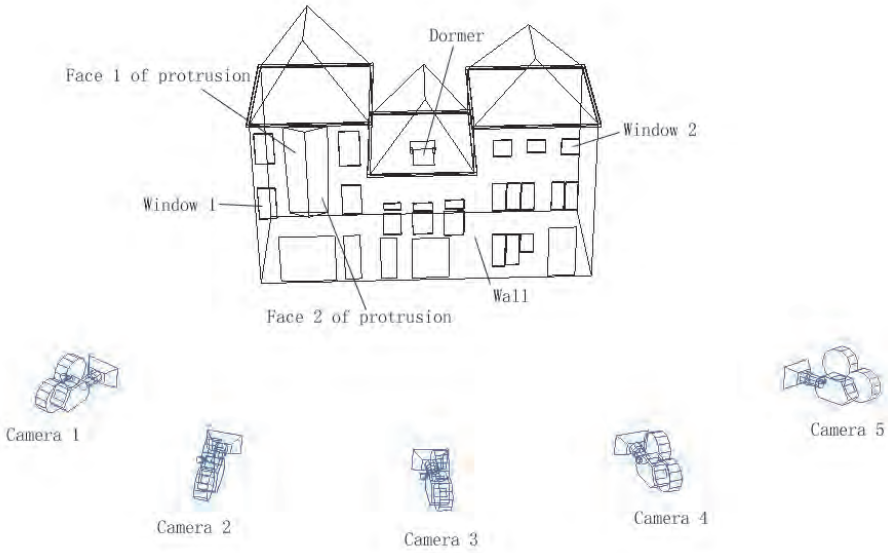


Figure 6.1: Choosing the optimal texture images for different model parts.

Suppose region G is the geometry projection of the occlusion onto the wall (see Figure 6.2), and region T is the image projection under central perspective. The region of T minus G is textured alternatively by pixels from other perspectives, or in other words, by images captured from more left or right positions.

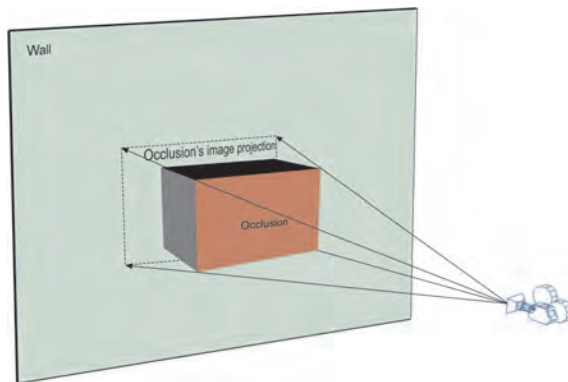


Figure 6.2: An occlusion and its central perspective projection on a wall.

6.2 Calculating texture coordinates

A texture is defined in its own u, v coordinate system, and the texture coordinates indicate how the texture should be aligned to the model before it is “glued on” the model. Specifically, the texture coordinate points to a texture pixel (texel) that should be used for colouring a particular model region. For planar geometries, it is not necessary to specify the texture coordinates for every region that needs to be textured. As long as the texture coordinates of the polygon vertices are specified, the texture coordinates of each region on the same plane can be interpolated with perspective mapping.

The perspective mapping can be represented by a matrix M in homogeneous coordinates. A point (u,v) in texture space is mapped on (x,y) in screen space in the following way:

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = M \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}; \quad x = x'/w; \quad y = y'/w; \quad ;$$

M is an arbitrary 3 by 3 matrix:

$$M = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$$

Because the transformation between object coordinate and screen coordinate is known (from viewing origin, direction and scaling), the texture coordinate of each region in object space can be calculated if M is determined. If we rewrite the above equations, the dependency between screen coordinate and texture coordinate can be expressed as:

$$x = \frac{au + bv + c}{gu + hv + 1}$$

$$y = \frac{du + ev + f}{gu + hv + 1}$$

which is actually the classical photogrammetric rectification equations.

Derivation of the eight unknown elements of M requires input of at least four observations. In the previous step of spatial resection, tie points between models and images have been semi-automatically determined, and they can be directly used to calculate M . A more direct way is to locate the corresponding image pixel of a model vertex directly using the camera parameters, and then the texture coordinate is just a scaled coordinate of this pixel in relative to the texture image's size. The latter approach is adopted in this research.

Polygons

With the coordinates of the polygon vertices and the camera parameters, calculation of vertices' texture coordinates are very straightforward. The steps are give below:

1. Choose the optimal texture image I . Make necessary refinements.
2. For a polygon vertex $V(x,y,z)$, calculate its image projection $P(w, h)$ on I using the collinearity equations.
3. If the size of I is W by H pixels, the texture coordinate of V is simply $(w/W, h/H)$.

The above mapping assumes that the whole image will be directly used as the texture without any cutting. This seems to be computationally inefficient, while OpenGL is capable of selecting only the necessary pixels and store them as rasterized colour values inside the display list, therefore minimal memory resources are consumed.

Curved surfaces

In OpenGL, curved surfaces are tessellated into small triangles before final visualization. To enable textured visualization, the texture coordinates of the triangle vertices should be specified. As a triangle is polygon with three vertices, the texture coordinates can be calculated with the same method.

6.3 Results and discussion

In this section, three test cases are provided to demonstrate the applicability and remaining problems of the presented method. The densities of the laser point clouds are approximately 800 points per square meter. The Cyclorama images of the first two cases were captured with interval of one meter, so usually any building is visible from more than 10 images. However, the actually used Cyclorama images are resampled to intervals of 3 meter to reduce the number of processed images. The Cyclorama images of the last case were captured with an interval of 10 meter.

6.3.1 The three joined houses

Figure 6.3 shows the reconstructed facade model of the three joined houses in Figure 4.5. The blue lines shows the geometries of the model, which are initially modeled from laser points and then refined by strong image lines. There is a

dormer on top of the left roof which is not modeled, because the density of laser points in that part is too low to recognize any features. Although this roof is visible from images (see Figure 5.6), the roof is still missing in the final model because images are only used to refine existing geometries. The texturing of the two-face protrusion is not accurate either. This is because the three houses' facade is considered to be coplanar when modelling the laser points, but in fact the wall of the left house has a small angle (approximately 3 degrees) with the other two walls. During segmentation of the laser points, this level of angle is below the threshold for separating a new segment. However, an offset of a few pixels appears after the texturing stage. The texturing of the most left two windows are also not very accurate because of the neglected wall angle. The left bottom corner is not correctly modeled because we assume "wall facades intersect ground". Poor texturing effect is caused consequently by this modelling error.



Figure 6.3: Textured facade model of three joined houses.

6.3.2 The house with a balcony

Figure 6.4 shows the reconstructed facade model of a house with a balcony on its facade. Note that the projection of the balcony remains on the wall's texture and the middle window's texture in Figure 6.4(a). After the wrong textured region is located, the images captured from the most left and most right locations are used for texturing alternatively (see Figure 6.4(b)). Unfortunately, all the image acquisition locations are below the balcony, so there is no image reaching the region which is behind and above the balcony. A part of the middle window is also inside

this region. Occluded texturing of this window is avoided by replacing the actual texture with a pre-defined window image, as shown in Figure 6.4(b). The car on the right bottom corner remains on the wall texture, because we do not yet handle the occlusions by non-building objects.

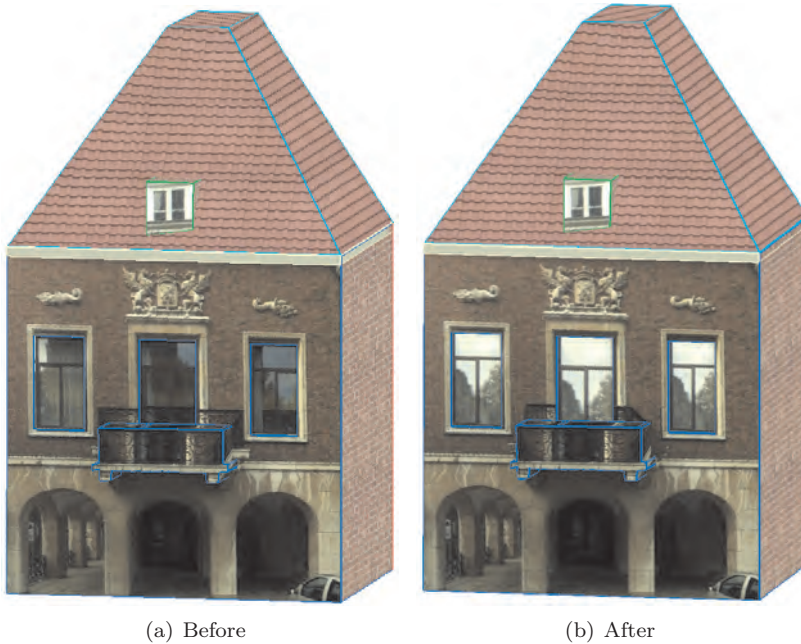


Figure 6.4: A reconstructed facade model before and after removing occluded texturing.

6.3.3 The curved walls

Figure 6.5 shows the curved walls of Figure 4.6 after automated texture mapping. The resolution of the textures is not high because the images are taken from 30 meters away. Some linear features (such as windows) seem to be curved after texturing, because the sample rate of tessellation is set not so high that a few triangles are generated to approximate the B-spline surfaces, and consequently produces such a (relatively) poor texturing and visualization effect. The small planar wall in between the two curved walls is not textured (white is the default colour), because no Cyclorama image has been captured with a suitable perspective to this small wall.

6.3.4 Discussion

It is realized from the test cases that accuracy of geometric modelling and image orientations are both vital to reconstructing photorealistic models. The wireframe models from only laser points are sufficient for certain purposes (navigation for

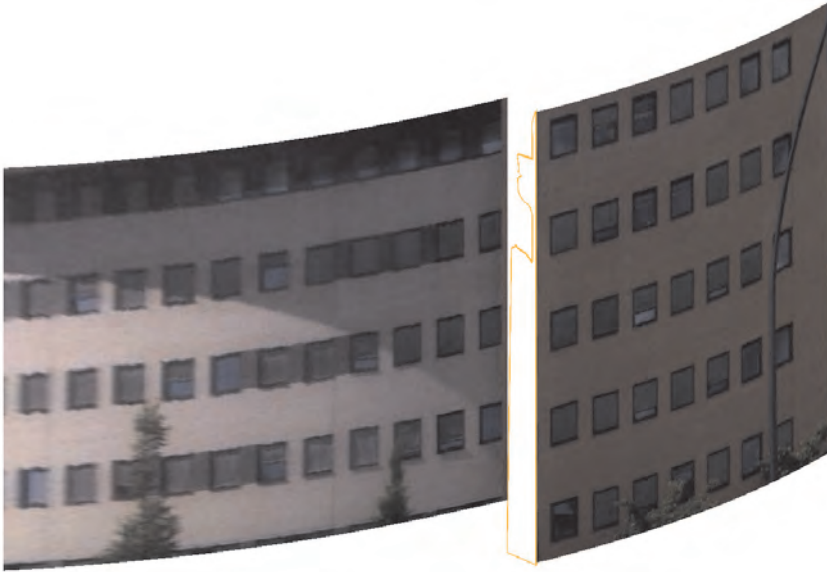


Figure 6.5: Two curved walls after texturing.

example). Decimeters even meters' inaccuracy of such building models can be tolerated, as long as the footprints coincide with 2D maps, or the general structures are recognizable together with other building models. But the accuracy requirement is much stricter when texturing is concerned, because even a few centimeter's modelling error may lead to serious colour contrast near the edges. The three tested cases are both small sized buildings, and only one group of manual selected tie points are used for spatial resections. It is anticipated that covering of a large building requires more images. The accumulated error in homography estimation might be too large for the images which are faraway, and in turn leads to errors in their spatial resection. Additional manual assistance should be made to make sure that exterior orientations of all images are accurately computed. However, the required minimum distance between two manually processed images still need to be examined.

Method evaluation

This chapter examines the efficiency and accuracy of our reconstruction methods which are presented in the previous chapters. As mentioned earlier, the manual reconstruction is both time-consuming and inaccurate, therefore our quality analysis will focus on the speed of the reconstruction procedure and the correctness of the reconstructed models. In total, three data sets (a TLS data set of Vlaardingen and two MLS data sets of Enschede and Esslingen) are used for the method evaluation. The Vlaardingen buildings were reconstructed with a manual approach before. The details of the manual approach are also provided to make a comparison with our approach. The reconstructed Vlaardingen models are textured with the Cycloramas available, while the Enschede and Esslingen models are geometric wireframes without textures since no optical data is available or suitable. Some manual edits are applied in the Vlaardingen and Enschede experiments to fix the errors caused by the automated methods. While in the Esslingen case, the buildings are fully automatically reconstructed because we wish to test the methods' performance under full automation mode.

This chapter is organized as follows. Section 7.1 describes the procedures of our approach and the manual approach. Section 7.2 to 7.4 provide the results and method performances of the three experiments respectively. The analysis conclusions and considerations for future work are drawn in the final section.

7.1 The reconstruction approaches

This section explains the two experimented reconstruction approaches: our approach and the traditional approach by a project partner company. We believe such a comparison between the new and traditional approaches are beneficial to realize the promotions of the new approach as well as its limitations.

7.1.1 Our approach

Based on the pipeline proposed in Figure 1.8, the methods presented in the previous chapters are organized as a semi-automatic reconstruction process:

1. Select single working data sets according to the ground plan (as explained in Section 3.1). Each data set should be a single building or a row of buildings whose facades are on the same plane.
2. Retrieve the candidate texture images which are taken within 15 meters from the center of the facade footprint.
3. Apply spatial resection to the central image (as explained in Section 5.2.2).
4. Calculate homography transformations between the neighboring images, and then apply the homography matrices to the exterior orientation of the central image to obtain the exterior orientation of each candidate texture image (as explained in Section 5.2.3).
5. Segmentation of the point cloud using Surface Growing method (Vosselman et al., 2004).
6. Automatically extract features (as explained in Chapter 3). The detection of windows should be enabled only when the point density of the wall facade is sufficient and even.
7. Manually select the miss detected features and assign their feature types. Remove any incorrect features.
8. Reconstruct the feature geometries (as explained in Chapter 4).
9. Manually edit the incorrect model edges. The possible editing operations are listed in Table 7.1.
10. Improve model edges according to image lines, and apply real or pre-defined textures (as explained in Chapter 5 and 6). The roofs and side walls are textured with pre-defined images, because usually they have quite poor visibility from the actual close-range images. The eaves are hypothesized and often differ with the actual shapes and positions, so they are also textured with pre-defined images.

Among these steps, 1, 2, 4, 5, 6, 8 and 10 are automatic operations, and 3, 7 and 9 are manual operations. The efficiency of the reconstruction approach can be directly reflected by the reconstruction time, but the evaluation of method correctness is less straightforward. Despite the errors in the pre and post reconstruction steps such as spatial resection and texturing, there are different kinds of errors on the geometric models, such as missed detected dormers, incorrect hypothesis of eaves and side walls, or incorrect roof outlines. We do not prefer to measure these errors directly because it is very time consuming. Instead, we believe the

time spent on the manual edits is a good indicator of the method’s correctness, because the better the automated process works, the less human intervention should be required. In the Vlaardingen and Enschede experiments, manual edits are introduced to fix the remaining errors of automated reconstruction, and the time is recorded in order to indicate the method’s correctness. Finally, it should be noted that in the discussions above we assume that the correctness of the final models mainly depends on the reconstruction methods, although the data quality (especially data gaps) also plays an important role.

7.1.2 The traditional approach

The Vlaardingen data was previously reconstructed with a traditional approach, which represents the current standard building reconstruction methods. It consists of the following steps:

1. Choose an appropriate texture image from the photo database.
2. In Photoshop, remove the perspective and save it to a temporary file.
3. In 3DStudio Max extrude the ground plane of the building.
4. Map the temporary texture to the model and refine the model to fit the image.
5. Repeat steps 1 to 4 for the rest of the facades and other elements.
6. Measure all the 3D elements and rescale the textures so that they have a consistent 32 pixels per meter.

Operation name	Description
Move vertex (keyboard)	Use the arrow keys to move a vertex on the plane determined by the polygon.
Move vertex (mouse)	Use the mouse to drag a vertex and magnetize it to a laser point.
Move edge	Use the arrow keys to parallel move a edge on the plane determined by the polygon.
Delete vertex	Delete a vertex (if the polygon has more than three vertices) and connect its left and right vertices.
Add vertex	Select a vertex and make a clone, then add the cloned vertex next to the original vertex in the polygon.
Delete edge	Delete a model edge (if the polygon has more than four vertices) and connect its left and right vertices.
Delete polygon	Delete a entire polygon.

Table 7.1: Manual editing operations in our approach.

7. Organize all the textures into 1 final texture with dimensions of 2^n by 2^n pixels (less than 1024) and remap the building.
8. Retouch texture, i.e. remove blocking objects (people, street lights, cars and so on) and fill in unknown areas.

7.2 The Vlaardingen case

This section provides the experiment details of the Vlaardingen data. A central area of the Dutch town Vlaardingen (see Figure 7.1) was scanned with a Leica terrestrial laser scanner HDS3000 in 6 scans (Miteck, 2006). The maximum scanning range of HDS3000 is 100 meters. The position accuracy is 6 mm and the distance accuracy is 4 mm for scanning range between 1 and 50 meters (Leica, 2004). Multiple targets of high reflectance were set up over the area, so that they can be detected by Cyclone (the postprocessing software of Leica scanners) as the tie objects for registration. The registered point cloud was then geo-referenced to the large-scale Dutch base map GBKN (Grootschalige BasisKaart Nederland). The laser data acquisition took 10 hours in total, which means 1.5 hour for each scan on average. Actually, each single scan only took 15 minutes, where a lot of time was spent on placing targets and moving the scanner. The average point density of the Vlaardingen data is about 500 points per square meter on walls, although the large variation in point density is due to the overlapping of neighboring scans. In total 21 building facades were selected for the experiment, and some of them have been shown in the previous chapters. The experiment was processed with a standard configuration PC with 2.5 GHz CPU and 2 GB memory.

The reconstructed geometry models and textured models are shown from Figure 7.14 to Figure 7.15. If a building was previously modeled by the traditional approach, the result is shown together for comparison. The time spent on each reconstruction step is provided in Table 7.2. Finally, a comparison of the reconstruction speed between the traditional approach and the our approach is given in Table 7.3.

The reconstructed building models still have some remaining errors, which are caused by either the limitations in our methods or the program's limitation. Firstly, so far our methods only deal with curved walls, but not other feature types in curved shapes. The intersection of curved shapes and planar shapes are also not considered yet. Secondly, the manual operations can make adjustment to the segments' feature types and the generated outlines, but cannot create outlines from scratch. There should be at least a few points for human operator to recognize the feature outline. Thirdly, the program currently does not allow manual switch of the Cyclorama used for texturing mapping even when the automatically specified ones are not satisfactory, and also does not allow manual edits of the hypothesized parts. These remaining errors are:

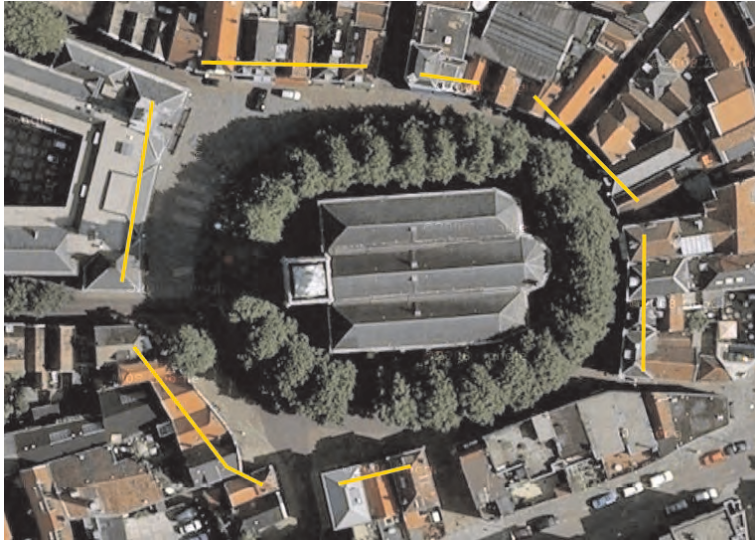


Figure 7.1: The Vlaardingen town center (yellow lines indicate the reconstructed buildings).



Figure 7.2: Reconstruction of Markt 1 (left: geometry model; middle: textured model; right: by traditional approach)



Figure 7.3: Reconstruction of Markt 2 (left: geometry model; right: textured model)



Figure 7.4: Reconstruction of Markt 3-4 (left: geometry model; right: textured model)



Figure 7.5: Reconstruction of Markt 7-8 (left: geometry model; right: textured model)



Figure 7.6: Reconstruction of Markt 9 (left: geometry model; right: textured model)

Markt 2 The texturing effect of the wall is poor because of the occlusion of cars.

Markt 7-8 The outline of the right dormer is not correct because of an insufficient number of laser points.

Markt 9 The hypothesis of the eave position is not correct.

Markt 11 The three holes between the pillars are recognized as windows, so they are removed.



Figure 7.7: Reconstruction of Markt 11 (left: geometry model; right: textured model)



Figure 7.8: Reconstruction of Markt 12 left part (left: geometry model; right: textured model)



Figure 7.9: Reconstruction of Markt 12 (left: geometry model; right: textured model)



Figure 7.10: Reconstruction of Markt 20 (left: geometry model; right: textured model)



Figure 7.11: Reconstruction of Markt 21-22 (left: geometry model; right: textured model)



Figure 7.12: Reconstruction of Markt 28 (left: geometry model; right: textured model)



Figure 7.13: Reconstruction of Markt 30 (left: geometry model; right: textured model)

Markt 12 The texturing effect of two dormers are poor, because there is no appropriate perspective from any Cyclorama.

Markt 20 The detection of windows is not complete, therefore the detected windows are removed before texturing. The door is not recognized.

Markt 21-22 The wall facade is slightly curved but treated as planar facade. This is because currently we cannot attach planar roofs to freeform wall facades, so the wall has to be modeled planar to keep the roof. The door to the right was not recognized, but has no influence on the visualization. The



Figure 7.14: Reconstruction of Markt 31-34 (left: geometry model; right: textured model; bottom: by traditional approach)



Figure 7.15: Reconstruction of Markt 42-43 (left: geometry model; middle: textured model; right: by traditional approach)



Figure 7.16: Reconstruction of Markt 44-46 (left: geometry model; middle: textured model; right: by traditional approach)

texturing effect of the dormer is poor, because the Cyclorama was taken in front of the building and the vision to the dormer was blocked by the eave.

Markt 28 The hypothesis of the eave position is not correct.

Markt 31-34 The left dormer has not been modeled because the point density is too low. There is a curved protrusion below the two-faced protrusion, and a curved corner at the left bottom position of the wall facade.

Markt 42-43 The dormer is missing because no point is available. The roof is



Figure 7.17: Reconstruction of Markt 47 (left: geometry model; middle: textured model; right: by traditional approach)

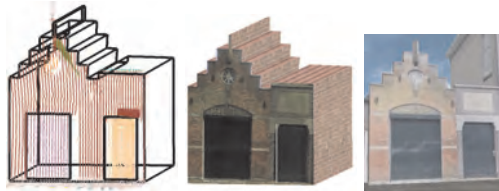


Figure 7.18: Reconstruction of Markt 48 (left: geometry model; middle: textured model; right: by traditional approach)



Figure 7.19: Reconstruction of Markt 49 (left: geometry model; middle: textured model; right: by traditional approach)

found smaller after a comparison with the image.

Markt 44-46 The outline of the right dormer is not correct because of insufficient points. The texturing effect of the dormers are poor due to a tilted image acquisition angle. The roof is found to be smaller after comparing with the image.

Markt 47 The texturing effect of the dormer is very poor, because the image was taken too close to the building and could not reach the dormer.

Markt 60 middle The texturing effect is poor because the spatial resection for



Figure 7.20: Reconstruction of Markt 57 (left: geometry model; middle: textured model; right: by traditional approach)



Figure 7.21: Reconstruction of Markt 60 middle facade (left: geometry model; right: textured model)



Figure 7.22: Reconstruction of Markt 60 right facade (left: geometry model; middle: textured model; right: by traditional approach)

such a narrow part is difficult.

Markt 60 right The roofs are not reconstructed because there is almost no point available on the roofs.

7.3 The Enschede case

This section provides details about the experiment of the Enschede data. A few streets in the Enschede city of the Netherlands were scanned with an Optech Lynx MLS system. A brief introduction of MLS systems has been given in Section 1.1. The range accuracy of Lynx is $\pm 7mm$ (Optech scanner), and the absolute accuracy is $\pm 5cm$ assuming a good GPS signal (Optech, 2009). The complete Enschede data are made up of in several strips (parts), and we selected a small strip of the street H.J. van Heekstraat (Heekstr. for short) for the experiment. This strip had a point density of about 400 points per square meter on walls, and the points were quite evenly distributed. Note that the hardware environment used was the same as for the Vlaardingen experiment.

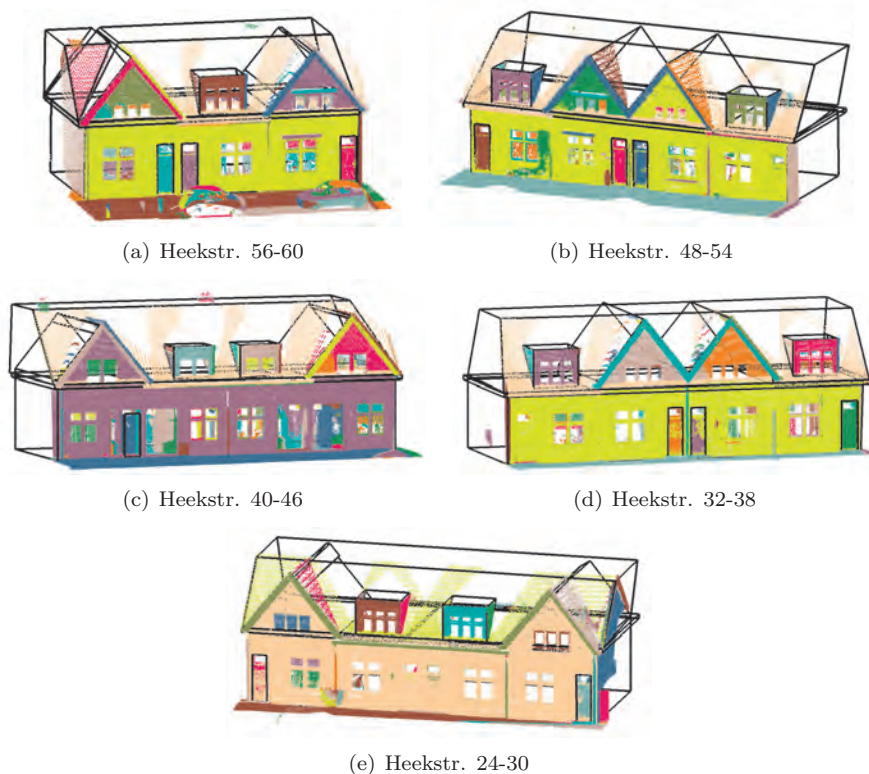


Figure 7.23: Reconstructed building models of Enschede

Figure 7.23 shows the reconstruction result of the Enschede data. The time spent on each step is provided in Table 7.4. Notice that the models are not textured in this experiment because the images captured by Lynx were not calibrated and therefore unsuitable for texture mapping. We found that the reconstruction of the Enschede data was much faster than the Vlaardingen data. This was due to 1) the Enschede points were less dense, more evenly distributed, and contain less data

gaps (than most parts of the Vlaardingen data), 2) the buildings are simpler, and 3) the disabling of texture mapping. There were also a few remaining errors in the Enschede models, which are summarized below:

- The eaves were incorrectly hypothesized in all five buildings. In our approach, the eave is just hypothesized to be a block object below a roof if the roof has a horizontal lower boundary. This hypothesis has been examined to be too simple.
- Some doors were not recognized because they were too close to the wall facade and did not result in separated segments.
- In Heekstra. 40-46, the reconstructed doors were fixed on the wall plane because the door feature is defined to be “on the wall” in the knowledge base, but these doors were actually intruded inside the wall.

7.4 The Esslingen case

This section provides details about the experiment of the Esslingen data. The Esslingen town in Germany was scanned with a StreetMapper 2 MLS system. The StreetMapper 2 system has a similar system structure as the Lynx system, but there are two principle differences: the deflecting style of the laser scanner, and the configuration of laser scanners. In Lynx, the laser beams are deflected from the scanner in a circle by a rotating mirror with a 360 degree field of view, while in Streetmapper 2 the laser beams are deflected in a line by a polygon with multiple reflective surfaces. In Lynx, two scanners are mounted and oriented perpendicularly to each other as illustrated in Figure 7.24 left. In StreetMapper 2, up to four scanners are mounted as illustrated in Figure 7.24 right. The range accuracy of StreetMapper 2 is $\pm 6mm$ (Riegl Z390), and the absolute accuracy is $\pm 3cm$ (3DLaserMapping, 2009). Comparing with the Vlaardingen and Enschede data, we found that the point density of Esslingen data was much lower: around 100 points per square meter on walls. It should be noted that the same PC was used to process the data from all of the experiments described herein.



Figure 7.24: The scanner configuration of Lynx (left) and Streetmapper 2 (right) (3DLaserMapping, 2009)

Figure 7.25 shows the reconstruction result of Esslingen data. The models are fully automatically generated, which took less than 5 minutes in total. No manual

adjustment is applied in this experiment. The textures are absent because there is no optical data available. The detection of windows and doors was disabled in this case, because the point density was not sufficient to reliably extract their geometry. The visual quality checking against the raw laser data detected a maximum outline offset of 5 cm on the facade outlines, which was still not accurate enough for automatic texturing and further processing. The roofs and dormers which are partially visible from street side might be incorrectly modeled, for example the second most right house in Figure 7.25. This is because outline generation is based on the hypothesis for parts where laser points are unavailable or partially available. Reasonable balance between knowledge based modeling and data based modeling should be further researched.

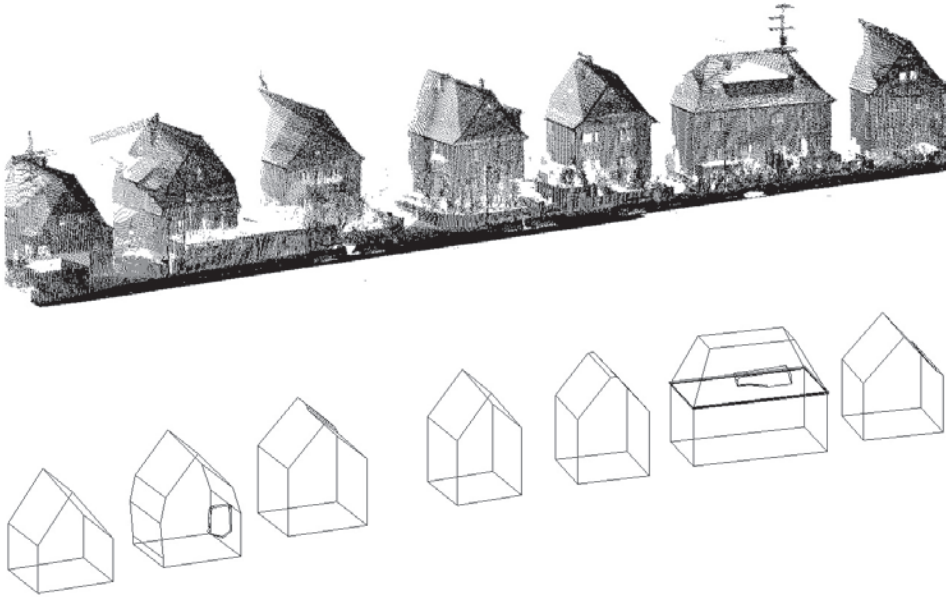


Figure 7.25: Reconstructed building models of Esslingen

7.5 Conclusions

By analyzing the three experiments results and speeds, the following conclusions and considerations for future research can be drawn:

- The Vlaardingen experiment proves that our approach is significantly faster than the traditional approach. The overall reconstruction time of our approach is only 7 percent of the traditional approach on average. The speed increase is higher at the texturing stage than the geometric modeling stage.

About 68 percent of our reconstruction time is spent on geometry, and 32 percent on texturing. In the traditional approach, about 34 percent of the time is spent on geometry and 66 percent on texturing.

- The overall reconstruction time of our approach depends primarily on the manual steps, which again depends on the number of building facades in a building and the number of small features. Currently, we only deal with one single building facade at a time. When a building has several facades, reconstructions have to be processed separately for each facade, and a final building model is combined from the single facade models. It can be anticipated that the reconstruction time would be much longer for buildings with multiple small facades. Within the reconstruction procedure of a single facade, most manual editing is dedicated to editing outlines of dormers and protrusions. It is usually very fast to reconstruct buildings without dormers and protrusions, or at least with good scanning of them.
- The number of laser points and wall size have little influence on the automated reconstruction time, since the computations are not very heavy.
- The time of point selection, image selection (of the Vlaardingen experiment) and segmentation are almost the same for each sub data set, and leave little room for improvement in the future. The time of spatial resection is also equal for each sub data set of the Vlaardingen data, because the same number (4 in this experiment) of tie pairs are selected.
- The models reconstructed with our approach may sometimes look less realistic, although not necessarily less accurate. On one hand, we missed a number of features which are modeled by the traditional approach. On the other hand, although the manually modeled models appear realistic, several model parts are actually incorrect. The manual observation and measurement over optical data is not always reliable. For example, some roofs are found more horizontal than their actual orientation, probably because the operators did not have enough time to measure every plane orientation on the images.
- The even distribution of laser points is vital to the reconstruction process, especially for small features. Our algorithms rely largely on good segmentation, while unevenly distributed laser points are often over segmented. Normally it is easier to recombine the large features using knowledge, while the small features are very vulnerable to data gaps or low point density. For example, if a roof is partially occluded by its dormer and in turn separated into multiple isolated segments, it will be very difficult to recombine them automatically.
- Extensive manual edits are required when the point density is too low, for example Markt 60 middle and Markt 43. From the experiments we realize that 100 points per square meter (on walls) is the minimum point density

requirement for our approach. We find no dependency between the algorithms' performances and the point density as long as the point density is higher than 100 points per square meter.

- The automated reconstruction of wall facades and roofs are satisfactory. There is seldom a need for manual editing.
- Automated extraction of small features is not very accurate, not only because there are less laser points, but also because their differences with decorations are less obvious. For example, awnings can be easily identified as wall protrusions. The feature constraints for small features should be further considered in the next level of research.
- The availability of windows' outlines does not always improve the visualization, while the modeling is quite time-consuming. It is more cost effective to disable the extraction of windows.
- Eaves are incorrectly modeled in several buildings because they are only hypothesized instead of adapting the sensor data. The eaves were considered not so important before. However, from the experiments we realized that eaves can have very large influence on the visualization. Therefore, a feature type for eave should be added in the future.
- It is very interesting to realize that the feature composition trees (see Figure 2.3) are almost the same between Heekstr.48-54 and Heekstr.32-28, and between Heekstr.40-46 and Heekstr.24-30. The features' truth factors (see Section 2.3) are also identical in the two pairs, for example the walls of Heekstr.48-54 and Heekstr.32-28 are both around 98% true, and the roofs are around 85% true. The buildings with the same design are very common in the real world. Detection of the similarities between buildings should make it possible to fill the data gaps on one building by referencing other similar buildings, or even clone the building models completely.
- The texturing effect by the traditional approach is usually better, because certain texture optimizations are applied. For example, the illuminations of all images are normalized to remove the influence of the sunlights' angle. In the future we should consider integrating similar operations, too.
- The usage of predefined textures at some parts is due to the unavailability of real textures or poor texture qualities. In the future we should at least retrieve the mean color of the corresponding image region, and combine it with the predefined texture in order for it to be much more natural in appearance.

	Markt 1	Markt 2	Markt 3-4	Markt 7-8	Markt 9	Markt 11
Number of points	596400	555181	207160	114424	101392	314320
Wall area (m^2)	45	32	57	75	89	94
Select points	0.5	0.5	0.5	0.5	0.5	0.5
Select images	0.5	0.5	0.5	0.5	0.5	0.5
<i>Spatial resection</i>	1	1	1	1	1	1
Homography	1	1	1	1	0.5	1
Segmentation	0.5	0.5	0.5	0.5	0.5	0.5
Feature extraction	0.5	1	0.5	0.5	0.5	1
<i>Feature adjustment</i>	1	2	0	3	0.5	1
Geometric modeling	0.5	0.5	0	0	0	1
<i>Model adjustment</i>	2	1.5	1	2	0.5	1
Texturing	0.5	0.5	0.5	0.5	0.5	0.5
Manual time	4	4.5	2	6	2	3
	(50%)	(41%)	(36%)	(63%)	(40%)	(38%)
Total time	8	11	5.5	9.5	5	8

	Markt 12 left	Markt 12	Markt 20	Markt 21-22	Markt 28	Markt 30
Number of points	710262	334159	128053	138554	143548	174732
Wall area (m^2)	124	158	76	49	25	33
Select points	0.5	0.5	0.5	0.5	0.5	0.5
Select images	0.5	0.5	0.5	0.5	0.5	0.5
<i>Spatial resection</i>	1	1	1	1	1	1
Homography	1.5	1.5	1	0.5	0.5	0
Segmentation	0.5	1	0.5	0.5	0.5	0.5
Feature extraction	2	2.5	0.5	0.5	0.5	0.5
<i>Feature adjustment</i>	1	2	0	0.5	0.5	0
Geometric modeling	1	1	0	0	0	0
<i>Model adjustment</i>	1	4	0.5	0.5	0.5	1
Texturing	1	0.5	0.5	0.5	0.5	0.5
Manual time	3	7	1.5	2	2	2
	(30%)	(48%)	(30%)	(40%)	(40%)	(44%)
Total time	10	14.5	5	5	5	4.5

	Markt 31-34	Markt 43	Markt 44-46	Markt 47	Markt 48	Markt 49
Number of points	371910	4463	97116	25003	10923	14867
Wall area (m^2)	185	65	102	63	28	44
Select points	0.5	0	0.5	0	0	0
Select images	0.5	0.5	0.5	0.5	0.5	0.5
<i>Spatial resection</i>	1	3	1.5	1	1	1
Homography	1.5	1	2	1	0.5	1
Segmentation	0.5	0	0.5	0	0	0
Feature extraction	1	0	0.5	0	0	0
<i>Feature adjustment</i>	2	2	3	1	0	1
Geometric modeling	0.5	0	0.5	0	0	0
<i>Model adjustment</i>	3	5	5	2.5	2	2
Texturing	1	0	1	0.5	0	0.5
Manual time	6 (55%)	10 (87%)	9.5 (63%)	4.5 (69%)	3 (75%)	4 (67%)
In total	11.5	11.5	15	6.5	4	6

	Markt 57	Markt 60 middle	Markt 60 right
Number of points	32438	48571	175708
Wall area (m^2)	49	26	157
Select points	0	0	0.5
Select images	0.5	0.5	0.5
<i>Spatial resection</i>	1	1.5	1
Homography	0.5	0	1.5
Segmentation	0	0	0.5
Feature extraction	0.5	0	0.5
<i>Feature adjustment</i>	0	0.5	1
Geometric modeling	0	0	0
<i>Model adjustment</i>	0	0.5	1.5
Texturing	0.5	0	0.5
Manual time	1 (25%)	2.5 (83%)	3.5 (47%)
In total	4	3	7.5

Table 7.2: Reconstruction speed of the Vlaardingen data with our approach (The time of each step is given in minutes; the italic names indicate manual operations)

	Traditional approach	Our approach	Remarks
Markt 1	92	8	The traditional approach textured the right facade and modeled 4 awnings.
Markt 31-34	98	11.5	The most right roof of the traditional approach is more horizontal than the actual orientation.
Markt 42-43	37	11.5	The dormer is missing in our approach.
Markt 44-46	74	15	The dormer and roof are better modeled by the traditional approach.
Markt 47	36	6.5	The dormer is better textured by the traditional approach, while lower than the actual position.
Markt 48	48	4	
Markt 49	65	6	The dormer is better textured by the traditional approach.
Markt 57	33	4	The eave is more accurate by the traditional approach.
Markt 60	106	10.5	The model is more complete by the traditional approach.

Table 7.3: Comparison of the reconstruction speeds (in minutes) of the Vlaardingen data

	Heekstr. 56-60	Heekstr. 48-54	Heekstr. 40-46	Heekstr. 32-38	Heekstr. 24-30
Number of points	127806	113686	138323	102563	110527
Wall area (m^2)	81	99	110	105	110
Select points	0.5	0.5	0.5	0.5	0.5
Segmentation	0	0	0	0	0
Feature extraction	0.5	0.5	0.5	0.5	0.5
<i>Feature adjustment</i>	0.5	0.5	0.5	0.5	0.5
Geometric modeling	0	0	0	0	0
<i>Model adjustment</i>	0	0.5	0.5	0.5	0.5
Manual time	0.5 (33%)	1 (50%)	1 (50%)	1 (50%)	1 (50%)
In total	1.5	2	2	2	2

Table 7.4: Reconstruction speed of the Enschede data with our approach (The time of each step is given in minutes; the italic names indicate manual operations)

Conclusions and recommendations

In this thesis, we have presented a knowledge based framework for automated building facade reconstruction, and provided a number of algorithms to solve the local problems within the framework. The conclusions on each procedure step have been given in the previous chapters. This chapter draws the overall conclusions of the presented works in Section 8.1 and the recommendations for the future work in Section 8.2.

8.1 Conclusions

- Generalized human knowledge about building structures provides a solid theory background for automated building reconstruction. Buildings are the products of human constructions, where similar rules can be traced throughout the construction processes. As reconstruction is a reverse process of the construction, any pre-acquired knowledge is also useful to the decomposition of building structures.
- The balance between data-driven and model-driven highly depends on the purpose of the reconstruction. The hypotheses lead to geometrically consistent models, but do not guarantee accurate models. When the reconstruction is demanded for urban development planning, security analysis or heritage documentation, the approach should focus on the data and make minimal hypotheses, especially when textures are mapped from actual images. For purposes like computer games and navigation, the models need to be realistic more than accurate, so more hypothesized models and pre-defined textures can be included.
- It is essential to first understand the semantic meaning of each feature and the relationships between features, instead of simply generating and combining the features outlines. The main reason is that undesired data and data gaps are very common in both laser point clouds and images, due to the line-of-sight nature of laser scanning and image acquisition. Identification

of these undesired data and data gaps requires known context information. Besides, the availability of semantics makes the building models applicable to more purposes.

- The knowledge based feature extraction compares quite a number of attributes and spatial relationships, most of which can be derived easily from laser data but not from optical data. The availability of optical data may add some clues, but the success of the reconstruction mainly depends on the quality of the laser data: sufficient point density and evenly distributed points.
- Model refinement with significant image lines has been proven to be beneficial to both geometric modeling and texture mapping, but the reliability of the images should be considered. In airborne images, the image lines mainly come from roof edges, so the data's effectiveness to roof reconstruction is high. While in close range images, considerable image lines might be designated from the non-building objects which occluded the buildings during image acquisition. If too many of those lines are present, it is better not to refine models using images.
- Successful texture mapping depends on the correct building models, accurate alignment between the laser space and the image space, and the occlusion conditions of the texture images.
- The manual interactions are beneficial assistance to the automated reconstruction, and they seem to be inevitable. There are a great variety of facade structures in reality, and our knowledge base is only able to explain a part of them. Determination of curved features' semantics meanings are especially difficult, although the geometric modeling (by B-spline surface fitting) can be automatically done. The data gaps and inconsistencies between different sensor data also demand manual interactions. Unless all these difficulties are well solved, manual interactions are still necessary to correct the reconstruction errors.

8.2 Recommendations

- The knowledge base should be further developed to achieve a more accurate feature recognition and incorporate more feature types. The improvements should be focused on two aspects. Firstly, a more exact model to represent the uncertain situations should be raised. For example, the conditional belief between clauses should be considered. Secondly, spatial relations concerning curved features should be considered.
- The optimal parameters (claus weights, likelihood thresholds, etc.) of the proof tree are strongly related to the architecture styles. In the next level of research, the agent should be empowered the ability to "remember" and

“learn from” its experiences by statistical analysis over training samples. The feature recognition should be more accurate when the agent processes a similar building with its experiences.

- The terrestrial laser scanning technology has been more and more developed nowadays. Especially, the density of MLS point cloud is sufficiently high for building reconstruction, and there are a few data gaps on building facades (3DLaserMapping, 2009; Optech, 2009). Because of the market’s interest on MLS technology, it is anticipated that the quality of MLS data will only get better in the next few years. The future research could consider less about the effects of poor data quality on building reconstruction, but focuses on other aspects of the approach.
- Currently, the spatial resection of images are done semi-automatically. The accuracies of the resected camera parameters are strongly related to the manual tie points selection, which is slow and not really reliable. A fully automated method for accurate alignment of multiple images should be developed in the future.
- ALS and TLS have mutually compensated acquisition perspectives, so more completed point clouds of buildings can be obtained by fusing these two data. It would be interesting to reconstruct buildings with reliable geometries on both facades and roofs by fusing ALS and TLS data. The seamless integration of the two data sources should be achieved first.
- The semantic feature based representation of our building models coincide naturally with the CityGML models (Kolbe et al., 2005), especially with LOD (Level of Detail) 3. In the next step, we will make support to the CityGML to exchange and store our building models in a standard geospatial schema.

Bibliography

- 3DLaserMapping (2009). Streetmapper 2 datasheet. Technical report, <http://www.streetmapper.net> (accessed 2009.11.21).
- Barber, C., D. Dobkin, and H. Huhdanpaa (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)* 22(4), 469–483.
- Barnea, S. and S. Filin (2008). Keypoint based autonomous registration of terrestrial laser point-clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* 63(1), 19 – 35. Theme Issue: Terrestrial Laser Scanning.
- Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool (2008). Speeded-up robust features (SURF). *Comput. Vision Image Understanding* 110, 346–359.
- Becker, S. (2009). Generation and application of rules for quality dependent faade reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing* 64(6), 640 – 653.
- Becker, S. and N. Haala (2007). Combined Feature Extraction for Façade Reconstruction. In *Proceedings of the ISPRS Workshop Laser Scanning 2007 and SilviLaser 2007, Espoo, Finland, 12-14 Sept. 2007*, pp. 241–247.
- Brenner, C. (2005). Building reconstruction from images and laser scanning. *Inter. J. Appl. Earth Obs. Geoinf.* 6, 187–198.
- Brenner, C., C. Dold, and N. Ripperda (2008). Coarse orientation of terrestrial laser scans in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing* 63(1), 4–18.
- Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 679–698.
- Cornelis, N., B. Leibe, K. Cornelis, and L. Van Gool (2008). 3D urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision* 78(2), 121–141.
- Dick, A., P. Torr, S. Ruffe, and R. Cipolla (2001). Combining single view recognition and multiple view stereo for architectural scenes. In *Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings*, Volume 1.

- Faro (2009). Photon 120 datasheet. Technical report, <http://www.faro.com> (accessed 2009.11.1).
- Fischler, M. and R. C. Bolles (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 381–395.
- Frueh, C., S. Jain, and A. Zakhor (2005). Data processing algorithms for generating textured 3D building facade meshes from laser scans and camera images. *Inter. J. Comput. Vision* 61, 159–184.
- Frueh, C., R. Sammon, and A. Zakhor (2004). Automated texture mapping of 3D city models with oblique aerial imagery. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pp. 396–403.
- Haala, N. and C. Brenner (1999). Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing* 54(2-3), 130–137.
- Hough, P. (1962, December 18). Method and means for recognition complex patterns. US Patent 3,069,654.
- Kolbe, T., G. Gröger, and L. Plümer (2005). CityGML—Interoperable access to 3D city models. In *First International Symposium on Geo-Information for Disaster Management GI4DM*. Springer.
- Leica (2004). Leica hds3000 product specifications. Technical report, <http://www.cyra.com> (accessed 2009.11.21).
- Lemmens, M. (2001). Terrestrial Laser Scanners 1st Survey. *GIM International* 1.
- Lemmens, M. (2004). Terrestrial Laser Scanners 2nd Survey. *GIM International* 12.
- Lemmens, M. (2007). Terrestrial Laser Scanners 3rd Survey. *GIM International* 8, 41–45.
- Lemmens, M. (2009). Terrestrial Laser Scanners 4th Survey. *GIM International* 8, 62–67.
- Lindenberger, J. (2009, 2). Presentation on optech lynx and topscan gmbh. Presentation.
- Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Inter. J. Comput. Vision* 60, 91–110.
- Maas, H. and G. Vosselman (1999). Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing* 54(2-3), 153–163.

- Miteck, E. (2006). Bsik report: Rapportage 3d laserscanning. Technical report, Oranjewoud.
- Optech (2009). Lynx datasheet. Technical report, <http://www.optech.com> (accessed 2009.11.1).
- Oude Elberink, S. and G. Vosselman (2009). Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations. *Sensors* 9(8), 6101–6118.
- Pereira, F. and S. Shieber (2002). *Prolog and Natural-Language Analysis*. Microtome Publishing.
- Piegl, L. and W. Tiller (1997). *The NURBS book*. Springer Verlag.
- Piegl, L. and W. Tiller (2000). Surface approximation to scanned data. *The visual computer* 16(7), 386–395.
- Pollefeys, M., D. Nister, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, et al. (2008). Detailed real-time urban 3d reconstruction from video. *Inter. J. Comput. Vision* 78, 143–167.
- Pu, S. and G. Vosselman (2009). Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing* 64(6), 575 – 584.
- Rabbani, T., S. Dijkman, F. van den Heuvel, and G. Vosselman (2007). An integrated approach for modelling and global registration of point clouds. *ISPRS J. Photogramm. Remote Sens.* 61, 355–370.
- Rabbani, T., F. van den Heuvel, and G. Vosselmann (2006). Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36(5), 248–253.
- Riegl (2009). Lms-z620 datasheet. Technical report, <http://www.riegl.com> (accessed 2009.11.21).
- Ripperda, N. (2008). Grammar Based Facade Reconstruction using RjMCMC. *Photogrammetrie Fernerkundung Geoinformation* 2008(2), 83.
- Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education.
- Sapkota, P. (2008). Segmentation of coloured point cloud data. Master’s thesis, International Institute for Geo-information Science and Earth Observation (ITC).
- Schindler, K. and J. Bauer (2003). A model-based method for building reconstruction. In *First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003. HLK 2003*, pp. 74–82.

- Schneider, D. and H. Maas (2003). Geometric modelling and calibration of a high resolution panoramic camera. *Opt. 3D Meas. Tech. VI 2*, 122–129.
- Shreiner, D., M. Woo, J. Neider, and T. Davis (2005). *OpenGL (R) Programming Guide: The Official Guide to Learning OpenGL (R), Version 2 (OpenGL)*. Addison-Wesley Professional.
- Sowa, J. (1992). Semantic networks. *Encyclopedia of Artificial Intelligence*.
- Suveg, I. and G. Vosselman (2004). Reconstruction of 3D building models from aerial images and maps. *ISPRS Journal of Photogrammetry and remote sensing* 58(3-4), 202–224.
- Taillandier, F. and R. Deriche (2004). Automatic building reconstruction from aerial images: a generic Bayesian framework. In *Proceedings of the XXth ISPRS Congress, Istanbul, Turkey*.
- Tian, Y., Q. Zhu, M. Gerke, and G. Vosselman (2009). Knowledge-based topological reconstruction for building facade surface patches. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38(5).
- van den Heuvel, F., R. Verwaal, and B. Beers (2007). Automated Calibration of Fisheye Camera Systems and the Reduction of Chromatic Aberration. *Photogramm. Fernerkund. Geoinf.* 3, 157.
- Vosselman, G. (2002). Fusion of laser scanning data, maps, and aerial photographs for building reconstruction. In *2002 IEEE International Geoscience and Remote Sensing Symposium, 2002. IGARSS'02*, Volume 1.
- Vosselman, G., B. Gorte, G. Sithole, and T. Rabbani (2004). Recognising structure in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 46(8), 33–38.
- Wikipedia (2009). Building. Technical report, <http://en.wikipedia.org> (accessed 2009.6.1).
- Zisserman, A., T. Werner, and F. Schaffalitzky (2001). Towards automated reconstruction of architectural scenes from multiple images. In *Proc. 25th workshop of the Austrian Association for Pattern Recognition*, pp. 9–23.

List of publications

- S. Pu and G. Vosselman. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(part 5):25–27, 2006.
- S. Pu and G. Vosselman. Extracting windows from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(part 3):320–325, 2007.
- S. Pu. Automatic building modeling from terrestrial laser scanning. *Advances in 3D Geoinformation Systems, Springer*, pages 147–160, 2008a.
- S. Pu. Generating building outlines from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(part B5-1):451–456, 2008b.
- S. Pu and G. Vosselman. Refining building facade models with images. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(part 3):217–222, 2009a.
- S. Pu and G. Vosselman. Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6):575 – 584, 2009b. ISSN 0924-2716. doi: DOI: 10.1016/j.isprsjprs.2009.04.001.
- S. Pu and G. Vosselman. Building facade reconstruction by fusing terrestrial laser points and images. *Sensors*, 9(6):4525–4542, 2009c. ISSN 1424-8220. doi: 10.3390/s90604525.

List of Figures

1.1	Documenting facade details by terrestrial laser scanning and close range imagery	2
1.2	Two mainstream terrestrial laser scanners	4
1.3	A standard design structure of MLS system (Lynx Optech (2009))	5
1.4	Reconstructed facade models by Frueh et al. (2005)	9
1.5	Reconstructed 3D city models after augmented by virtual car models (Cornelis et al., 2008)	9
1.6	Interpretation of a building facade with the formal grammar defined in Ripperda (2008)	10
1.7	A building facade model reconstructed by Becker (2009)	11
1.8	Building reconstruction pipeline	13
2.1	Features types	19
2.2	Composition of building features in a semantic network.	23
2.3	Interpretation of a simple house with hierarchical feature composition.	24
2.4	A simple proof tree.	25
2.5	A simple building with four features.	28
2.6	Two buildings with multiple possible interpretations.	30
3.1	The process of feature extraction	34
3.2	Partitioning of a laser point cloud.	35
3.3	Building's 2D outlines.	35
3.4	Segmentation result of a building facade	37
3.5	Segmentation using colour information (left: coloured point cloud; right: segmentation result).	38
3.6	Segmentation with the smooth surface method.	39
3.7	Convex hull used as approximation of laser segment: (a) a roof segment with its convex hull; (b) approximated by convex hull's properties; (c) a wall facade segment intersects a ground segment; (d) approximated by checking intersected convex hulls.	40
3.8	A feature extraction result.	41
3.9	Extracting windows from holes in the wall: (a) inner hole points; (b) outer hole (boundary) points; (c) directly fitted rectangles from inner hole points; (d) after filtering out holes of extrusions and doors.	43

4.1	Wall outline generation (Left up: upper contour points; left down: least squares fitting; right: generated wall outline).	46
4.2	Wall outline generation process.	47
4.3	Generating window outline.	49
4.4	Hypotheses of the occluded areas	52
4.5	Reconstructed polyhedron model of a building facade. Errors occur at: 1. dormer; 2. round corner; 3. window; 4. wall boundary; 5. roof boundary.	53
4.6	Approximate curved laser segments with B-spline surfaces.	54
5.1	Model refinement process.	59
5.2	Registration between laser points and images.	60
5.3	The organization of a Cyclorama (α and β are the longitude and latitude of a pixel on the panoramic sphere; r is the angular resolution)	61
5.4	Conversion from panoramic perspective to central perspective	62
5.5	Selecting tie points for spatial resection.	63
5.6	Homography estimation using extracted SURF in two images (red lines: linked SURF pairs; black quadrangle: homographic transformed viewing plane of the upper image).	64
5.7	Locating occurrences of the same laser points in different image perspectives.	64
5.8	Extracting significant lines from an image (black: Canny edges; red: Hough lines; blue: projections of model edges; green: candidate matches; purple: best matches).	66
5.9	Matching model edges with image lines for refining the restaurant house's model.	68
5.10	Comparison of textured restaurant house model before and after refinement.	69
5.11	Matching model edges with image lines for refining the town hall's model	69
5.12	Matching and refining window boundaries.	70
5.13	A textured building facade model with intruded windows.	71
6.1	Choosing the optimal texture images for different model parts.	76
6.2	An occlusion and its central perspective projection on a wall.	76
6.3	Textured facade model of three joined houses.	79
6.4	A reconstructed facade model before and after removing occluded texturing.	80
6.5	Two curved walls after texturing.	81
7.1	The Vlaardingen town center (yellow lines indicate the reconstructed buildings).	87
7.2	Reconstruction of Markt 1 (left: geometry model; middle: textured model; right: by traditional approach)	87
7.3	Reconstruction of Markt 2 (left: geometry model; right: textured model)	87

7.4	Reconstruction of Markt 3-4 (left: geometry model; right: textured model)	88
7.5	Reconstruction of Markt 7-8 (left: geometry model; right: textured model)	88
7.6	Reconstruction of Markt 9 (left: geometry model; right: textured model)	88
7.7	Reconstruction of Markt 11 (left: geometry model; right: textured model)	89
7.8	Reconstruction of Markt 12 left part (left: geometry model; right: textured model)	89
7.9	Reconstruction of Markt 12 (left: geometry model; right: textured model)	89
7.10	Reconstruction of Markt 20 (left: geometry model; right: textured model)	89
7.11	Reconstruction of Markt 21-22 (left: geometry model; right: textured model)	90
7.12	Reconstruction of Markt 28 (left: geometry model; right: textured model)	90
7.13	Reconstruction of Markt 30 (left: geometry model; right: textured model)	90
7.14	Reconstruction of Markt 31-34 (left: geometry model; right: textured model; bottom: by traditional approach)	91
7.15	Reconstruction of Markt 42-43 (left: geometry model; middle: textured model; right: by traditional approach)	91
7.16	Reconstruction of Markt 44-46 (left: geometry model; middle: textured model; right: by traditional approach)	91
7.17	Reconstruction of Markt 47 (left: geometry model; middle: textured model; right: by traditional approach)	92
7.18	Reconstruction of Markt 48 (left: geometry model; middle: textured model; right: by traditional approach)	92
7.19	Reconstruction of Markt 49 (left: geometry model; middle: textured model; right: by traditional approach)	92
7.20	Reconstruction of Markt 57 (left: geometry model; middle: textured model; right: by traditional approach)	93
7.21	Reconstruction of Markt 60 middle facade (left: geometry model; right: textured model)	93
7.22	Reconstruction of Markt 60 right facade (left: geometry model; middle: textured model; right: by traditional approach)	93
7.23	Reconstructed building models of Enschede	94
7.24	The scanner configuration of Lynx (left) and Streetmapper 2 (right) (3DLaserMapping, 2009)	95
7.25	Reconstructed building models of Esslingen	96

List of Tables

1.1	Density (pts/ m^2) of MLS data acquired by Optech Lynx (Lindemberger, 2009).	6
2.1	Recognize grounds in Figure 2.5 with truth factors	28
2.2	Recognizing wall facades in Figure 2.5 with truth factors.	29
2.3	Value list to calculate reliability scores.	31
4.1	Handling irregular edges	48
7.1	Manual editing operations in our approach.	85
7.2	Reconstruction speed of the Vlaardingen data with our approach (The time of each step is given in minutes; the italic names indicate manual operations)	100
7.3	Comparison of the reconstruction speeds (in minutes) of the Vlaardingen data	101
7.4	Reconstruction speed of the Enschede data with our approach (The time of each step is given in minutes; the italic names indicate manual operations)	101

Curriculum Vitae

Shi Pu was born in Jining, China in 1983. From 1999 to 2003, he studied computer science at the Beijing Institute of Technology, and received a Bachelor's degree in 2003. After graduation in China, he enrolled for a Master's degree at the Technical University of Delft, the Netherlands, with a specialization in computer graphics. He received the Master degree in 2005 with a thesis on integrating GIS and CAD with geo-databases. After successfully finishing his Master's degree, he started a PhD study at the International Institute for Geo-Information Science and Earth Observation (ITC), on the topic of knowledge based reconstruction of building facade models from laser point clouds and images. Since September 2009 he has been working as a Postdoc at the ITC on the topic of feature extraction from mobile laser data for 3D road inventory.