Planar Segments Based Three-dimensional Robotic Mapping in Outdoor Environments

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von
Junhao Xiao
aus Hebei, China

May, 2013

Gutachter:
Prof. Dr. Jianwei Zhang
Prof. Dr. Reinhard Koch
Dr. Peer Stelldinger

Acknowledgements

Doing a PhD was a long and exciting journey, which would not be possible without the help and support of many people. First of all I am most grateful to my supervisor Prof. Dr. Jianwei Zhang for giving me the opportunity to pursue a PhD at the University of Hamburg and providing me such a good and liberal academic environment.

Next, I would like to extend my thanks to my daily supervisor and good friend Prof. Dr. Houxiang Zhang, whose support was always there for both technical and personal problems, not only in the first two years of my study when he was in Hamburg, but also for the second two years after he moved to Ålesund. His constructive criticism helped focus my ideas and his constant encouragement kept me going in spite of many stumbling blocks on the road.

Writing a dissertation is no easy task, but the same can be said of reviewing it. I would like to thank all the people who helped by reviewing parts of this work, especially my external reviewers — Prof. Dr. Reinhard Koch and Dr. Peer Stelldinger.

All the members of our TAMS group are greatly acknowledged, the time spent with you was wonderful. I own a great deal to Tatjana (Lu) Tetsis, thank you for helping me so much during the past time. My sincere thanks also go to Benjamin Adler for always being excited about my problems and ideas, always sharing his ideas with me, and the engaging discussions we had during and after work.

I am grateful to my former colleagues from NuBot, Huimin Lu, Xiangke Wang, and Shaowu Yang are especially acknowledged, thanks a lot for valuable comments and suggestions on my conference and journal papers, as well as this dissertation.

Being away from my family was not easy, however, many friends from Hamburg and other places provided another family away from home. I own special thanks to all of my friends. In this regard, Jianhua Zhang, Guoyuan Li, Gang Cheng, Bo Sun, and Weiwei Kong deserve special mention.

Finally, I would like to thank the people closest to me for their love and support — my parents and brother, my girl friend Fang, especially to my little nephew who brings a lot of happy moments.

The China Scholarship Council (CSC) is gratefully acknowledged for funding me during my studies in Hamburg.

Junhao Xiao
Hamburg, July 2013

Abstract

This dissertation focuses on the problem of three-dimensional (3D) outdoor robotic mapping. Laser scanners are chosen as the primary sensors and a novel approach for scan registration based on planar segments is developed. Unlike most existing approaches, it does not require an a-priori pose estimation from other sensors such as odometers and inertial measurement units (IMUs). Instead, the transformation is determined globally by searching corresponding planar segments between overlapping scans.

There are three steps in the approach:

The first step is to segment each point cloud into planar segments. Depending on the clutter level of the environment and whether the point cloud is organized, four complementary strategies have been proposed, namely a point based, a subwindow based, a hybrid and a cached octree region growing algorithm. Among them, the former three are limited to organized point clouds; in addition, the second one is restricted to structured environments. Based on observations from field experiments, the hybrid/cached octree region growing algorithms are recommended for organized/unorganized point clouds.

The second step is to calculate the area of each segment resulting from the first step. Again, segments from organized and unorganized point clouds are distinguished, where an alpha-shape based algorithm is proposed for unorganized point sets and a range-image based method is proposed for organized point sets, respectively.

The third step is to find segment correspondences and compute the transformation based on matched segments. The correspondences are searched globally in order to maximize a spherical-correlation-like metric, wherein the search space is pruned by both self-similarity and interrelations (geometric constraints). The novelty of the search algorithm is that only the area and plane parameters of each segment are required.

Four datasets acquired by scanners with different field of views have been used to evaluate the proposed approach: three are publicly available and one stems from our custom-built platform. Based on these datasets, the following evaluations have been done: segmentation speed-benchmarking, segment area calculation accuracy- and speed-benchmarking, and registration accuracy comparison with ground-truth. Also, the robustness of the approach with respect to occlusions and partial observations has been proven. The approach has been compared to the Iterative Closest Point (ICP)

and Minimum Uncertainty Maximum Consensus (MUMC) algorithms; furthermore, it has been successfully extended to the domain of map merging. Experimental results confirm that the approach offers an alternative to state of the art algorithms in plane-rich environments.

Kurzfassung

Die vorliegende Dissertation behandelt das Problem der dreidimensionalen Rekonstruktion von Außenlandschaften durch Roboter. Basierend auf den Daten von Laserscannern wird ein neuer Ansatz zur Scan-Registrierung anhand von Flächensegmenten entwickelt. Im Gegensatz zu den meisten bestehenden Verfahren benötigen die hier vorgestellten Algorithmen keine a-priori Schätzungen der Roboterposen von anderen Sensoren wie Rad-Encodern oder Trägheitsnavigationssystemen (IMUs), da die nötigen Transformationen zwischen überlappenden Scans durch globale Suche übereinstimmender Flächensegmente ermittelt werden.

Der vorgestellte Ansatz besteht aus drei Schritten:

Der erste Schritt besteht aus der Segmentierung einer Punktwolke in ihre Flächensegmente. Hierfür werden vier sich gegenseitig ergänzende Strategien vorgestellt, die für strukturierte und unstrukturierte Punktwolken mit unterschiedlicher Planheit der Oberflächen geeignet sind. Alle basieren auf dem Konzept des "Region-Growing", werden jedoch in Punkt-basiert, Fenster-basiert, hybrid und Cached-Octree-basiert unterteilt. Die ersten drei operieren ausschließlich auf strukturierten Punktwolken wobei die zweite strukturierte Umgebungen voraussetzt. Feldtests ergeben, dass die hybride bzw. Cached-Octree Region-Growing Ansätze für strukturierte bzw. unstrukturierte Punktwolken am besten geeignet sind.

Im zweiten Schritt wird die Größe der Flächensegmente aus dem ersten Schritt berechnet. Hierbei wird eine auf Reichweitenbildern basierende Methode für strukturierte und ein auf Alpha-Shapes aufbauender Algorithmus für unstrukturierte Punktwolken verwendet.

Der dritte Schritt setzt sich aus der Ermittlung übereinstimmender Flächensegmente und der Bestimmung der räumlichen Transformationen zwischen ihnen zusammen. Eine globale Suche wird verwendet, um eine sphärische Korrelationsmetrik zu maximieren, wobei der Suchraum durch Selbstähnlichkeit und geometrische Bedingungen beschnitten wird. Das Novum dieser Methode ist, dass hierfür ausschließlich die Fläche und die hessesche Normalform der Flächensegmente erforderlich ist.

Zur Evaluierung der aufgeführten Methoden wurden vier Datensätze herangezogen, die unter Verwendung verschiedener Laserscanner mit unterschiedlichen Sichtfeldern entstanden sind. Drei dieser Datensätze sind öffentlich verfügbar, einer wurde

mit einer selbst gebauten Roboterplattform aufgenommen. Anhand dieser Daten wurden folgende Untersuchungen ausgeführt: zum einen werden die Geschwindigkeit der Flächensegmentierung sowie die Genauigkeit und Geschwindigkeit der Größenermittlung ermitteln, zum anderen werden Genauigkeit und Zuverlässigkeit der Flächensegmentzuordnung untersucht. Ferner wird die Robustheit der Methode gegenüber Verdeckungen und unvollständigen Abtastungen validiert. Der Ansatz wird mit Iterative Closest Point (ICP) und Minimum Uncertainty Maximum Consensus (MUMC) verglichen und in seiner Anwendung auf die Registrierung ganzer Karten erweitert. Die ausgeführten Experimente zeigen, dass die vorgestellte Methode eine leistungsfähige Alternative zu etablierten Verfahren bei Umgebungen mit vielen Flächen darstellt.

Contents

List of Figures

List of Tables

Symbols

The list below contains the mathematical notations that are used most frequently throughout the thesis.

$v, \mathbf{v}, \hat{\mathbf{v}}$	a scalar, a vector and a unit vector
$\mathbf{u} \cdot \mathbf{v}$	the dot product between $\mathbf{u}$ and $\mathbf{v}$
$\mathbf{M}, \mathbf{M}^{-1}, \mathbf{M}^{+}$	a matrix, its inverse and its Moore-Penrose pseudoinverse
$\lvert M\rvert, \lvert M\rvert_{+}$	the determinant and pseudodeterminant of a matrix

The list below contains the mathematical symbols that are used most frequently throughout the thesis.

$\mathcal{F}_i$	the ith robot coordinate frame
${}^{t}\mathbf{p}_i$	a 3D point with coordinates $\{x_i, y_i, z_i\}$ in frame $\mathcal{F}_t$
$\mathcal{C}_n$	a nD point cloud (also represented by $\mathcal{C}$ for simplicity)
$\hat{\mathbf{n}}, d$	Hessian plane parameters
$\mathbf{m}$	the geometric center of a set of points, $\mathbf{m} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{p}_i$
$\mathbf{R}$	a rotation matrix
$\mathbf{t}$	a translation vector
$\mathbf{T}$	a transformation matrix

Glossary

3D-NDT Three-Dimensional Normal Distributions Transform.

aLRF actuated Laser Range Finder.

EGI Extended Gaussian Image.

FILO First-In-Last-Out.

FoV Field of View.

GNSS Global Navigation Satellite Systems.

ICP Iterative Closest Point.

IMU Inertial Measurement Units.

LRF Laser Range Finder.

MSE Mean Square Error.

MUMC Minimum Uncertainty Maximum Consensus.

NBV Next Best Viewpoint.

NDT Normal Distributions Transform.

PDF Probability Density Functions.

PTU Pan-Tilt Unit.

RANSAC RANdom SAmple Consensus.

SLAM Simultaneous Localization And Mapping.

SVM Support Vector Machine.

ToF Time of Flight.

Chapter 1

Introduction

A journey of a thousand miles
starts beneath ones feet.

Laozi

1.1 Motivation

HUMAN beings have been dreaming to create intelligent *robots* since the birth of science. Consequently, robots have been a popular topic of science-fiction literature and -films for a long period. The first use of the word "robot" can be traced back to Karel Čapek's play R.U.R. (Rossum's Universal Robots) in 1920. More than thirty years later, the first digital and programmable robot "Unimate" was invented by George Devol, which was employed to lift hot metal pieces from a die casting machine and stack them. Although earlier automatic machines can be found in the mythologies of many cultures, they cannot be considered robots, because they are not programmable and thus limited to performing repetitive tasks. Programmability is a fundamental part in most definitions of a robot.

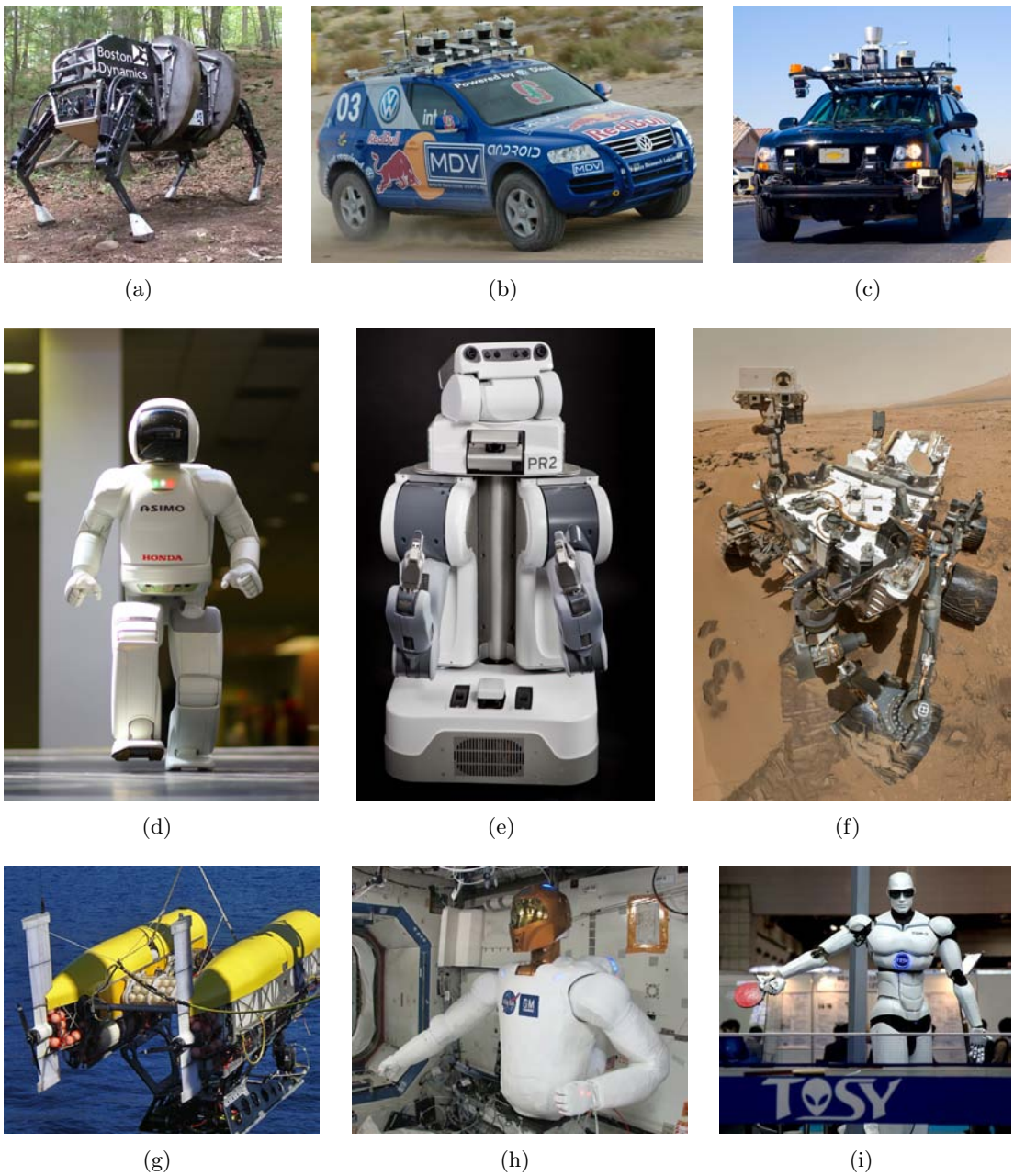
In the following decades, robotics technology entered a phase of rapid development, in which researchers have achieved a number of great milestones. Today, robots can be found almost everywhere, e.g. on Mars and in oceans, in hospitals and homes, in factories and restaurants, in museums and schools. Particularly, many countries such as Germany and Japan are facing aging populations, with an increasing number of elderly people in need of care and relatively fewer young people to care for them. Although robots are currently not capable of performing elderly care, the hope that they might become in future fosters intense research in these countries. In general, robots are playing more and more important roles in society, and robotics undoubtedly is one of the key technologies in this century.

Possibly because of influence from popular movies, society's perception and expectations of robotic skills has advanced much faster than the state of the art: many robots appearing in science-fiction films are almost omnipotent (see *Knight Rider, WALL-E,*

and A.I. Artificial Intelligence for examples). They are agile and intelligent, have emotions and the ability to reason, move fluidly in any environment. On the contrary, real-life robots are usually designed to outperform humans in single, well-defined tasks. Some state of the art robots are shown in Figure 1.1: BigDog (Boston Dynamics, 2013),shown in Figure 1.1(a), is capable of traversing difficult terrain while carrying a significant payload (150 kg capacity) and climbing a 35 degree incline. Stanley (Thrun et al., 2006) and Boss (Urmson et al., 2008), the vehicles have won the DARPA Grand Challenge 2005 and 2007, are shown in Figure 1.1(b) and 1.1(c) respectively. ASIMO (Honda, 2013), designed to help people who lack full mobility, is able to walk and run at speeds up to 6 km/h, see Figure 1.1(d). The PR2 robot (Figure 1.1(e)) is a service robotics testbed, aimed at research and innovation; moreover, its accompanying software suite ROS (Robot Operating System) (Quigley et al., 2009) has quickly become a standard tool in the robotics community. The Mars Curiosity rover (NASA, 2012a) shown in Figure 1.1(f) has been exploring the Red Planet since August 2012, and has provided key information about the planet. The Nereus underwater vehicle (Woods Hole Oceanographic Institution, 2013) (Figure 1.1(g)) has reached a depth of 10,902 meters in the sea. Being sent to the International Space Station in 2011, Robonaut 2 (NASA, 2012b) (Figure 1.1(h) is the first humanoid robot in space, and has been aiding astronauts on dangerous missions and freed them from mundane tasks. The robot shown in Figure 1.1(i), named TOPIO (TOSY, 2013), is a bipedal humanoid robot designed to play table tennis against humans. While playing, it can continuously learn and improve its skills through an artificial intelligent system.

Unfortunately, most humans rarely encounter autonomous robots in daily life; because modern robots are still working either in tightly controlled environments or under human supervision. This is mainly due to robots having difficulties in responding to unplanned changes in their environment. Exceptions in Figure 1.1 are Stanley and Boss, which have completed challenging courses in the DARPA challenge; however, they have been equipped with significant computational capabilities and expensive high-end sensors. Researchers have been working to enable robotic operations in unstructured environments in recent years, because this ability has the potential to profoundly impact our lives: for every robot employed for painting, welding, assembling in car production lines, assisting in medical surgery, cleaning and maintenance in offices and homes, there are a hundred more that could be used outdoors. The natural world raises challenges for robotics as we go from structured to unstructured, from static to dynamic, from controlled to natural. Needless to say, in order to execute tasks or interact with the surroundings, a very basic ability for a robot is to model the world using on-board sensors. Without a proper world model, it is impossible to answer the three general problems of mobile robot navigation (Leonard and Durrant-Whyte, 1991): "Where am I?", "Where am I going?", and "How should I go there?". Clearly, an accurate map is an essential part of the world model.

Mapping an unknown outdoor environment belongs to the research domain of Simultaneous Localization And Mapping (SLAM). Even though it is not a trivial task, SLAM in two-dimensional (2D) space has been solved using probabilistic methods — an intensive discussion of related techniques can be found in Thrun et al. (2005). However, it is difficult to extend those techniques to three-dimensions (3D), because when



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 1.1: Some state of the art robotic platforms. (a): BigDog, the rough-terrain Robot from Boston Dynamics; (b) and (c): Stanley and Boss, two self-driving cars which won DARPA Grand Challenge 2005 and 2007 respectively; (d): ASIMO, a humanoid robot created by Honda; (e): PR2 research platform from Willow Garage; (f): Self-portrait of Curiosity Rover in Gale Crater on the surface of Mars; (g): Nereus underwater vehicle which had reached a depth of 10,902 meters in the sea; (h): Robonaut R2, the first robot on the International Space Station (delivered in 2011); (i): TOPIO, a humanoid robot which can play table tennis against a human being.

the robot-pose's degrees of freedom grow from three to six, computational complexity grows beyond the capabilities of current hardware. Therefore, researchers turned to solving the problem using *scan registration*. Registration is the process of fusing several datasets into a globally consistent map, which is usually done by determining corresponding features between scans and minimizing a distance metric between them.

Hence, registration has attracted significant attention in recent years. Weingarten (2006) states that "up to now, all approaches successfully applied to 3D SLAM are based on the Iterative Closest Point (ICP) algorithm." However, ICP has bottlenecks in application to outdoor environments due to relying on a proper initial pose and being susceptible to local minima. Although some newer approaches on the topic like registration based on Extended Gaussian Images (EGIs) (Makadia et al., 2006), Three-Dimensional Normal Distributions Transform (3D-NDT) (Magnusson, 2009), and planar patches (Pathak et al., 2010b) were published after Weingarten's thesis, the area is still challenging and being explored. This motivated us to develop a global, fast, and accurate registration approach, which can provide an alternative to existing algorithms.

Cameras and Laser Range Finders (LRFs) are commonly used in robotic systems. However, cameras turn out to be problematic in outdoor mapping scenarios: First, severe illumination changes remains a monumental problem for cameras. Second, in order to gather range information using stereo vision, the scenario must feature abundant texture information. Furthermore, the range accuracy depends on the distance from the cameras to the object in relation to stereo baseline (i.e., the distance between the two cameras). Third, an alternative to stereo vision is to use a single camera, which does not suffer from such baseline-related problems. However, the inherent disadvantage with this approach is that building a realistically scaled map requires identification of objects with known extents in order to determine the global scale factor. Therefore, a LRF is chosen as the primary sensor in this dissertation.

On the one hand, as the appearance of outdoor environments can differ wildly, proposing a generally applicable algorithm is beyond the scope of this thesis. On the other hand, we do favor an approach valid for as many different scenarios as possible. As a compromise, we decided to focus on sceneries containing abundant planar surfaces, considering planar surfaces such as façades, roads, and walls are present in almost all urban and indoor environments. Therefore, the resulting technique can be employed to move service robots from indoor to urban surroundings, as well as for urban search and rescue missions.

1.2 Outline

The rest of this thesis is organized as follows:

Chapter 2 broadly introduces to the research domain and problems of robotic mapping.

Chapter 3 presents four datasets employed in this dissertation, where one is from our custom-built perception platform and the other three are publicly available. The sensors used for gathering these datasets feature different Field of View (FoV) and sensing ranges.

Chapter 4 discusses and benchmarks four proposed complementary strategies for 3D point cloud segmentation with regard to the clutter level of the environment and whether the point cloud is organized.

Chapter 5 studies methods of calculating the area covered by a set of coplanar points. Three methods are presented and benchmarked.

Chapter 6 proposes a constraint search for segment correspondence determination, where both a self-similarity metric and interrelations have been employed to prune the search space. A transformation matrix then is then computed from matched segments and refined by a closed-form optimization method. The algorithm is then benchmarked with regard to accuracy and speed and compared with existing approaches.

1.3 Research questions and contributions

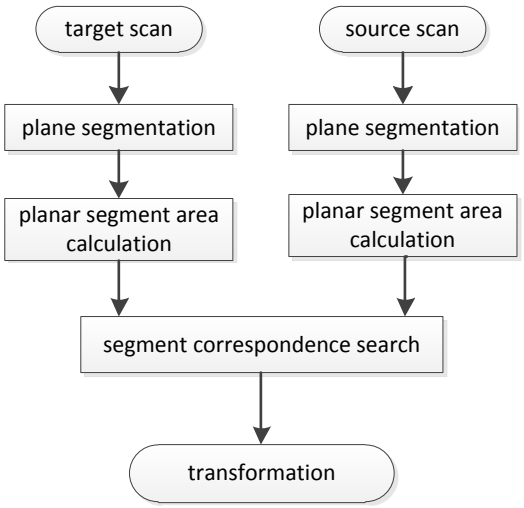


Figure 1.2: Diagram of the proposed scan registration pipeline.

The proposed registration pipeline is illustrated in Figure 1.2, presenting the three main research topics:

- How to efficiently segment a given 3D scan into planar segments.
- How to calculate the covered area of each resulting planar segment.
- How to determine the pose relation between two overlapping scans based on their area attributed planar segments.

Consequently, the following contributions have been achieved:

- A custom-built 3D perception platform, which is capable of delivering omni-directional range and intensity images simultaneously (Chapter 3).
- A point based region growing algorithm for organized point cloud segmentation (Chapter 4).

- A subwindow based region growing algorithm, limited to organized point clouds obtained from structured environments (Chapter 4).
- A fast, hybrid region growing algorithm for segmentation of organized point clouds (Chapter 4).
- A cached octree region growing algorithm, imposing no constraints on the structure of the point cloud (Chapter 4).
- A sound evaluation of the Delaunay triangulation and alpha-shape algorithms when being applied to planar segment area calculation (Chapter 5).
- An area calculation method proposed for planar segments from organized point clouds, utilizing the image-like structure (Chapter 5).
- A segment correspondence search algorithm for scan registration, using both self-similarity metric and interrelations for search space pruning. The search is performed globally, hence no prior pose estimation from other sensors is required (Chapter 6).
- A closed-form registration refinement method, based on fitting corresponding segments to one plane (Chapter 6).
- An investigation of how the registration pipeline can be extended to map merging and scan-map registration (Chapter 6).

1.4 Publications

Some of the work presented in this dissertation has been published in several journal and conference papers. The following list summarizes all the publications accomplished during the period of working towards this thesis, as well as the precise chapters that each article contributed to.

- Junhao Xiao, Jianhua Zhang, Jianwei Zhang, Houxiang Zhang, and Hans Petter Hildre (2011). Fast Plane Detection for SLAM from Noisy Range Images in Both Structured And Unstructured Environments. *IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1768–1773, Beijing, China. (Xiao et al., 2011)
  Part of Chapter 4
- Junhao Xiao, Jianhua Zhang, Benjamin Adler, Houxiang Zhang, and Jianwei Zhang. 3D Point Cloud Plane Segmentation for SLAM in Structured and Unstructured Environments. *Robotics and Autonomous Systems*. Under review, submitted on August 28, 2012. Advanced copy available (Xiao et al., 2012b)
  Part of Chapter 4

- Junhao Xiao, Bejamin Adler, and Houxiang Zhang (2012). 3D Point Cloud Registration Based on Planar Surfaces. *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 40–45, Hamburg, Germany. (Xiao et al., 2012a)
  Part of Chapter 6

- Junhao Xiao, Benjamin Adler, Houxiang Zhang, and Jianwei Zhang (2013). Planar Segment Based Threedimensional Point Cloud Registration in Outdoor Environments. *Journal of Field Robotics*. (Xiao et al., 2013)
  Related to Chapter 4 and 5, main part of Chapter 6

- Benjamin Adler, Junhao Xiao, and Jianwei Zhang(2012). Towards Autonomous Airborne Mapping of Urban Environments. *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 77–82, Hamburg, Germany. (Adler et al., 2012)
  Briefly introduced in Chapter 2

- Benjamin Adler, Junhao Xiao, and Jianwei Zhang (2013). Finding Next Best Views for Autonomous UAV Mapping through GPU-Accelerated Particle Simulation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Under review, Submitted to IROS 2013. Advanced copy available (Adler et al., 2013)
  Briefly introduced in Chapter 2

Not included in this dissertation:

- Hui Zhang, Xiangke Wang, Huimin Lu, Shaowu Yang, Shengcai Lu, Junhao Xiao, Fangyi Sun, Dan Hai, and Zhiqiang Zheng (2009). NuBot Team Description Paper 2009. *RoboCup 2009*, Graz, Austria. (Zhang et al., 2009)

- Dan Hai, Hui Zhang, Junhao Xiao, Zhiqiang Zheng (2010). Cooperate Localization of a Wireless Sensor Network (WSN) Aided by a Mobile Robot. *IEEE International Workshop on Safety Security and Rescue Robotics (SSRR)*, pages 1–6, Bremen, Germany. (Hai et al., 2010)

- Jianhua Zhang, Junhao Xiao, Jianwei Zhang, Houxiang Zhang, and Shengyong Chen (2011). Integrate multi-modal cues for category-independent object detection and localization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA. (Zhang et al., 2011)

Chapter 2

State of the art of robotic mapping

> Learn extensively, inquire thoroughly, ponder prudently, distinguish clearly and practice devotedly.
>
> *Liji·Zhongyong*

AUTOMATIC environment sensing and modeling, which has been and still is a hot research topic, serves as one of the fundamental issues in mobile robotics, since the availability of a map is essential for many tasks, such as localization, path planning and navigation. If the robot poses were known, the local sensor inputs of the robot could be aligned into a common coordinate system to create a map. Unfortunately, self-localization sensors such as odometers and Inertial Measurement Unitss (IMUs) suffer from imprecision; especially, no error boundaries can be guaranteed after long-term movements. Therefore, range sensor readings need to be used to create a precise map, which is usually done by scan registration.

In order to execute tasks in unknown environments, the robot should be able to model the environment and localize itself at the same time, which is called SLAM. The state of the art for 2D metric mapping are probabilistic methods, such as (Montemerlo et al., 2002) and Dissanayake et al. (2001). A tutorial on 2D SLAM techniques can be found in Durrant-Whyte and Bailey (2006) and Bailey and Durrant-Whyte (2006).

Theoretically, the probabilistic methods for 2D planar mapping are extensible to 3D mapping with 6D pose estimates. However, the computational cost of multi-hypothesis tracking, e.g., using particle filters, grows exponentially with the degrees of freedom. But no reliable strategy for reducing the computational cost has been proposed. As an alternative, 6D pose estimation by 3D scan registration has attracted increasing attention from the robotics society in recent years, see Makadia et al. (2006); Nüchter et al. (2007b); Magnusson et al. (2007); Pathak et al. (2010b) and Stoyanov et al. (2012) for examples. The accuracy of pose estimation by registration is typically one magnitude higher than odometry, but it still suffers from accumulation of error during long paths. To deal with this problem, places which have been multiply visited by the robot play a vital role. Once a loop closure has been detected, it can be used to bound the error by distributing the previous pose estimation errors, which can increase the

Figure 2.1: State of the art range sensors. (a): SwissRanger SR4000; (b): PMD[vision] CamCube 3.0; (c): SICK S300; (d): Hokuyo UTM-30LX; (e): Velodyne HDL-64E; (f): RIEGL VZ-400; (g): FARO Focus^{3D}; (h): Asus XtionPro.

robustness and reliability of mapping systems. Error distribution is usually optimized in a constraint network or pose graph, where scans and the spatial relation between them are represented as nodes and linked edges in the graph. Another essential input of pose graphs is the pose covariance of each node, which can be computed from scan pair registration.

For a fully autonomous exploration system, the robot should have the ability to determine the Next Best Viewpoint (NBV) for obtaining novel information, and plan a safe path from current pose to the selected viewpoint. In addition, because even one failed registration will make the resulting map unusable, strategies for false detection should be integrated into the mapping system, which can be solved by overlapping metric or shape similarities.

As mentioned above, there are several key techniques in a fully autonomous 3D mapping systems, such as data gathering, model representation, scan registration, loop closure detection, etc. We will give a short overview of recent works on each subtask in successive sections in this chapter. Since this thesis focuses on building an outdoor mapping system which can work round the clock, related work wherein cameras are used as the primary sensor will be isolated, considering vision sensors are sensitive to the amount of background illumination changes in the environment and cannot be employed during night.

2.1 3D scan gathering

The past decade has experienced an emergence of various range sensors, such as Time of Flight (ToF) cameras, inexpensive 2D LRFs, 3D LRFs and the newly introduced RGB-D style cameras, as shown in Figure 2.1. A ToF camera resolves distances based on the known speed of the light, by measuring the time of a light signal between the camera and the object. Different from laser scanners, the entire scene is captured simultaneously using a number of modulated beams. As a result, ToF cameras can provide depth and intensity information of the environment at a high frame rate. For example, SwissRanger SR4000 (Figure 2.1(a)) can work at 50 frames per second and PMD CamCube 3.0 (Figure 2.1(b)) features 40 frames per second. However, such sensors cannot be employed for outdoor map generation, due to unacceptable low signal-to-noise ratio under strong light conditions[1].

Structured light imaging techniques have popularized the RGB-D style cameras into the robotics community besides their applications for entertainment. The working principle for such devices is structured light imaging, i.e., by projecting a predefined light pattern onto an object which is simultaneously observed by a camera. The appearance of the light pattern in a certain region of the camera image varies with the distance between the camera and object, which is utilized to generate a depth image. At the same time, an additional camera can be used to gather the color information. Speckle patterns have been used in popular RGB-D cameras like the Microsoft Kinect and the Asus XtionPro (an Asus XtionPro is shown in Figure 2.1(h)). RGB-D style cameras have been successfully applied to indoor mapping, as in Henry et al. (2012) and Endres et al. (2012). For the same reason as ToF cameras, they are not suitable for outdoor mapping.

ToF based laser scanners also calculate the distance by measuring the time delay between an emitted light (laser) signal returns to the receiver, see Figure 2.1(c) and Figure 2.1(d) for examples. Different from ToF cameras, 2D LRFs sample the scenario point-by-point, commonly by rotating an embedded mirror to change the emitting direction of the signal, at the price of long data acquisition time. In order to get 3D depth information using such sensors, another degree of freedom is needed. This is usually done by installing a 2D LRF on a motor, rotating the LRF step by step to get 2D scan slices which are then joined together as a 3D scan. Such actuated Laser Range Finders (aLRFs) are clearly time costly for acquiring data. On the contrary, real 3D scanners such as Velodyne HDL-64E (Figure 2.1(e)) and VIGEL VZ-400 (Figure 2.1(f)) are much faster by integrating more complex mechanisms, at the same time, they are more expensive, hence unsuitable for low cost robotic systems.

As discussed above, aLRFs have been employed in most existing 3D outdoor robotic mapping systems (three examples are depicted in Figure 2.2), as well as in this dissertation. Normally, several to tens of seconds are needed for an aLRF to take a full scan, depending on the following factors: angle resolution of the third degree of freedom, speed of the motor, and beam resolution of the 2D LRF. Therefore, a common mode of collecting scan data using such sensors is to stop the robot while the scan is being made,

[1]Although PMD CamCube 3.0 is claimed to work in outdoor environment, its noise level is too high for reliable outdoor mapping under direct sunlight.

(a) (b) (c)

Figure 2.2: aLRF equipped platforms. (a): platform of Weingarten and Siegwart (2006); (b): platform of Nüchter et al. (2007b); (c): platform of Pathak et al. (2010a).

which is called stop-and-scan (also known as stop-scan-go) style. For an autonomous robot, it is not convenient to stop every few meters to make a scan, especially when dealing with dynamic environments.

Continuous 3D scanning

There have already been attempts to perform 3D scanning while moving, which actually is a continuous 2D to 3D registration problem. In other words, the acquired 2D scan slice is continuously registered to a fixed coordinate system. This is a trivial task if the sensor's pose is exactly known in real-time by self-localization sensors such as odometry or IMU. However, odometry is not reliable in outdoor terrain due to unstable contacts between the wheels and ground, especially side sliding on slopes. IMUs alone can estimate 3D accelerations and rotations, but suffer from noise and sensor drift during a long operation period. IMUs and Global Navigation Satellite Systems (GNSS) are known to be complementary to each other. While GNSS localizes at a lower precision than IMU but has no drift along with operation time. In our work on autonomous airborne mapping using an octocopter, we have successfully applied to fuse an IMU with RTK-GNSS (Real Time Kinematic GNSS) based on extended Kalman filter for scanning while flying, which is more challenging than that on ground robots. The octocopter is illustrated in Figure 2.3(a), and a typical generated point cloud is shown in Figure 2.4. Unfortunately, in some circumstances, such as urban or low altitude operations, GNSS receiver's antenna is prone to losing line-of-sight with satellites. Thus, GNSS cannot support reliable position information. Furthermore, only self-localization has been employed without considering feature correspondences in our system at present. Please see Adler et al. (2012) for detail.

However, using GNSS and IMU is not only expensive but also limited to GNSS covered environments. Two 2D LRFs have been installed orthogonally in Hänel et al. (2003), where one scans horizontally and the other one is upward-pointed, as shown in Figure 2.3(b). The horizontally installed scanner then is employed to map an unknown environment in 2D for recovering the robot path. At the same time, the upward pointed scanner sample the 3D structure of the environment. Clearly, this system is limited to robots moving over a flat surface, which is not the case of outdoor environments. Even

(a) (b)

(c) (d)

Figure 2.3: Experimental platforms have been employed for continuous 3D scanning. (a): our custom-built octocopter in Adler et al. (2012, 2013); (b) platform used in Hänel et al. (2003); (c) platform used in Stoyanov and Lilienthal (2009); (d): platform used in Bosse and Zlot (2009a).

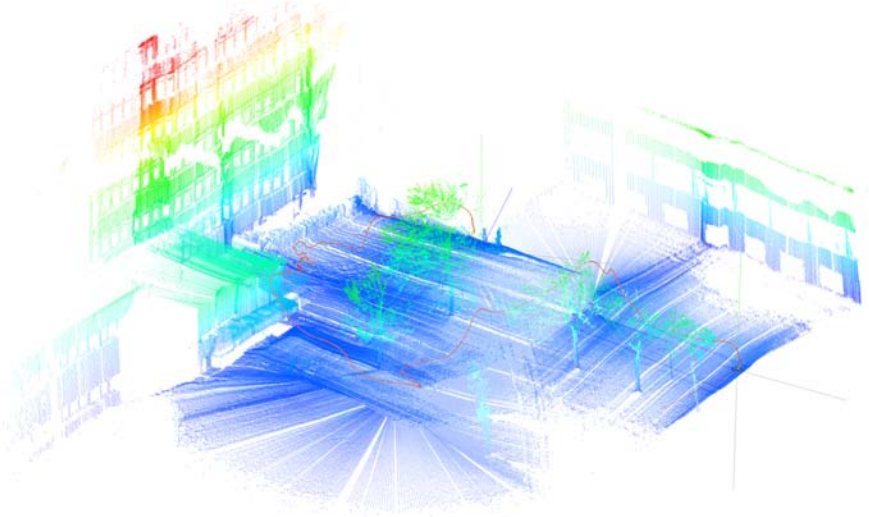


Figure 2.4: A point cloud obtained by our octocopter, see Figure 2.3(a). The recovered trajectory is shown as a red curve.

in indoor environments, as reported in the paper, "the normals are almost uniformly distributed" for a planar surface due to localization error in 2D, wherein the noise level is much higher than scans gathered using the stop-scan-go style (see Section 4.4 for a qualitative comparison).

An interesting idea for recovering motion while scanning is presented in Stoyanov and Lilienthal (2009), where an aLRF is employed for data gathering. For a mobile robot with continuous motion, there should be re-observed structures between the current and previous scans. Therefore, the current scan can be aligned to the coordinate system of previous scans using exist registration algorithms. ICP (it will be introduced in Section 2.3) is performed whenever a new scan is gathered with odometry information as an initial guess. Then, the accumulated error is distributed to the covered path using an estimation algorithm, generating Maximum Likelihood point clouds. The bottleneck of the approach is that it is restricted to planar motion.

Bosse and Zlot (2009a) also proposed a promising continuous 3D scanning approach by registration with a spinning 2D LRF. The main contribution of their work is that they use only laser data and require no self-localization information. The employed 3D sensor is constructed by mounting a SICK laser on a spinning platform, which is shown in Figure 2.3(d). The laser is spun about the sensor front at a rate of 0.5 Hz. Instead of matching each scan slice to previous scans as that in Stoyanov and Lilienthal (2009), they try to match the point cloud from half a spin ("sweep") to the last sweep. Each sweep is divided into 3D cubes, and a shape descriptor for the surface in each cube is estimated in the form of ellipsoids, which is quite similar to the 3D-NDT representation (see Section 2.2). To account for the non-uniform density in the point cloud, a pyramid of grids with increasing cell sizes along with the distance from sensor origin is built. They further determine how planar or cylindrical each cell is, followed by searching correspondences between planar and cylindrical features. The geometric constraints between planar and cylindrical features are used to estimate the transformation in an ICP style between two successive sweeps. To discretize the trajectory over a sweep interval, the points in each sweep are grouped according to recorded time stamps when they are sampled, and a cubic spline is utilized to reconstitute a smooth, continuous trajectory which is employed to reform the scan.

2.2 Map representation

In order to achieve true autonomy for a mobile robot, an internal representation of its surrounding environment is required. Therefore, the question of how to represent 3D robotic maps in a manner that is easily stored, manipulated and processed has received much attention in recent years, and several different approaches have been proposed and successfully applied to robotic systems. Most of them can be classified as either topological or geometric representation – the former are usually based on the topology of space rather than its physical shape, while the latter directly utilize a set of geometric primitives. Due to their predominance and widespread use in robotics, metric maps have been the focus of this thesis, and a short overview on them is given below.

The very basic form is to directly use the aligned raw sensor data, namely the point cloud, which is straightforward and useful for visualization purpose. In such maps,

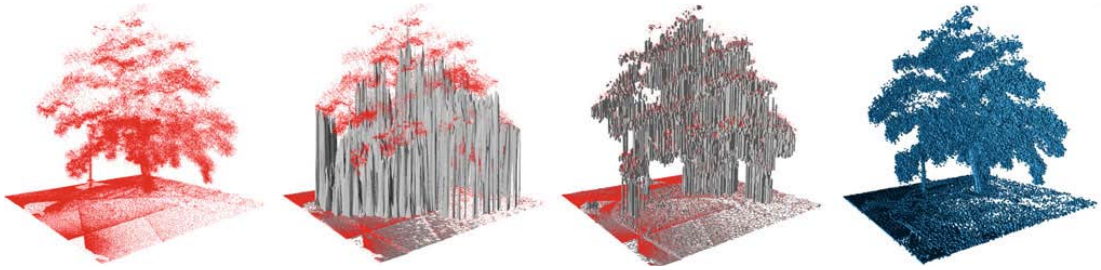


Figure 2.5: 3D representation of a tree as a point cloud (left), elevation map (middle left), MLS map (middle right), and octomap. The figure is reprinted from Wurm et al. (2010).

the endpoints returned by range sensors are used to model the occupied space in the environment. Point clouds have been employed in a number of robotic mapping systems such as Nüchter et al. (2007b); May et al. (2009) and Andreasson and Lilienthal (2010), see Figure 2.4 and 2.5 for examples. One limitation of this method is that the sensor noise has not been considered. Thus, it can only be used for high precision sensors. Another limitation is high memory consumption, which is limited to the number of scans.

If certain assumptions about the mapped area can be made, the 3D space can also be parameterized to 2.5D representation, namely the elevation map. Basically, it is obtained by associating a height value to cells organized in a 2D grid (Herbert et al., 1989). This method clearly results in a significant reduction of memory requirements. However, it is problematic to be utilized for robot navigation because only one height in each cell leads to improper representation of structures with multi-surfaces. For example, a robot cannot plan a path under a table whose top surface is higher than the height of the robot. Multi-level surface (MLS) map (Triebel et al., 2006) is proposed to cope with this problem, storing multiple surfaces in each cell of the grid. An example of elevation and MLS map is shown in Figure 2.5.

A well-known approach for modeling environments in 3D is to use a grid of cubic volumes of equal size to discretize the mapped area, i.e., occupancy grids map, which have demonstrated their capability on 2D mapping (Montemerlo et al., 2002) in the past decades. An occupancy grid, or evidence grid, is a 3D voxel in which each cell stores information about the probability of that area or volume being occupied by some object. Occupancy grids are well suited to integrate the noise and low resolution input, this is one major reason why occupancy grids were used before LRFs became more common. A major drawback of this method in 3D space is the large memory requirement. In large-scale outdoor scenarios or where a fine resolution is needed, the memory consumption becomes prohibitive (increasing the resolution of a 3D grid 10 times requires 1000 times more memory). To deal with this problem, multi-resolution occupied grids (Ryde and Hu, 2010) have been proposed, which are able to represent 3D models while keep a relative low memory requirement. However, it does not differentiate between free and unknown volumes.

Tree-based representations such as octree have also been used for model representation and visualization, which has the advantage to delay the initialization of map

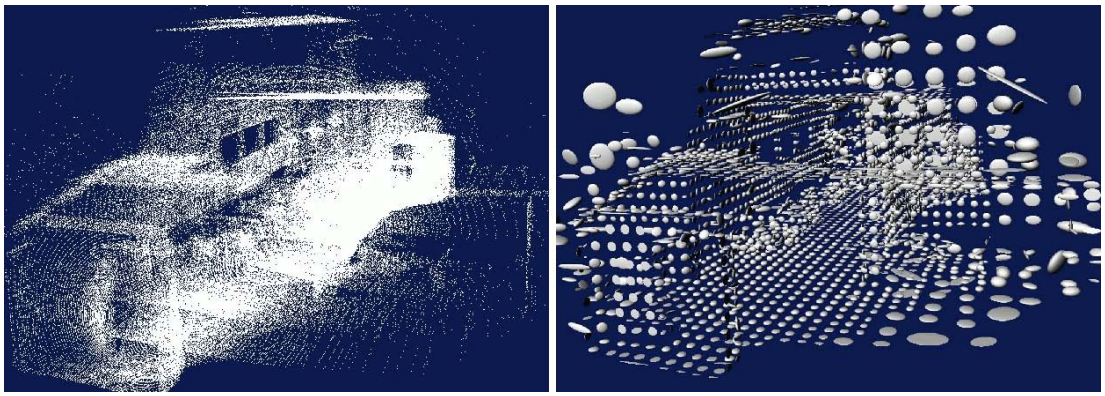


Figure 2.6: Left: A point cloud acquired at an indoor hallway. Right: 3D-NDT representation of the point cloud. Ellipsoids are employed to visualize the Gaussian distributions in each cell, wherein ellipsoids are centered at distribution means, as well as oriented and scaled according to the respective covariance matrices. The figure is reprinted from Stoyanov et al. (2010).

volumes until sensor readings need to be integrated, i.e., a bounding-box is not required beforehand. Moreover, it is suitable for multi-resolution purpose using different depths. However, there is no probabilistic information of each leaf being occupied, which is necessary for dealing with noise. As a result, the idea of occupancy grid has been integrated with octree in Payeur et al. (1997); Wurm et al. (2010), and an example is illustrated in Figure 2.5.

As a method for surface representation, Normal Distributions Transform (NDT) was originally proposed for 2D scan registration by Biber and Strasser (2003). It was extended to three dimensions in Magnusson and Duckett (2005). The main idea is to map a scan to a set of Gaussian Probability Density Functionss (PDFs). The scan is divided into a grid of cells (cubes in 3D) first. Afterwards, for each grid containing more than some minimum number of points, the points in it are assumed to be sampled from a Gaussian distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$. The maximum-likelihood estimation (MLE) of the mean vector and covariance matrix of $\mathcal{N}$ are calculated as

$$\mu = \sum_{i=1}^{N} \mathbf{p}_i \tag{2.1}$$

$$\Sigma = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{p}_i - \mu)(\mathbf{p}_i - \mu)^{\mathrm{T}} \tag{2.2}$$

where $\mathbf{p}_i, i = 1, 2, ...N$ are the points in the corresponding grid. This, however, could be a proper or improper representation of the sampled points, depending on the object model and the scale of grid. A single Gaussian distribution is clearly not enough to represent even simple objects, but could be a good estimate of a small piece of the surface. Therefore the space should be discretized into small parts in order to generate a good estimate. For each cell, only the Gaussian distribution parameters are stored, which results a significantly memory reduce comparing to raw data, see Figure 2.6 as an example. 3D-NDT has found its application in the area of scan registration

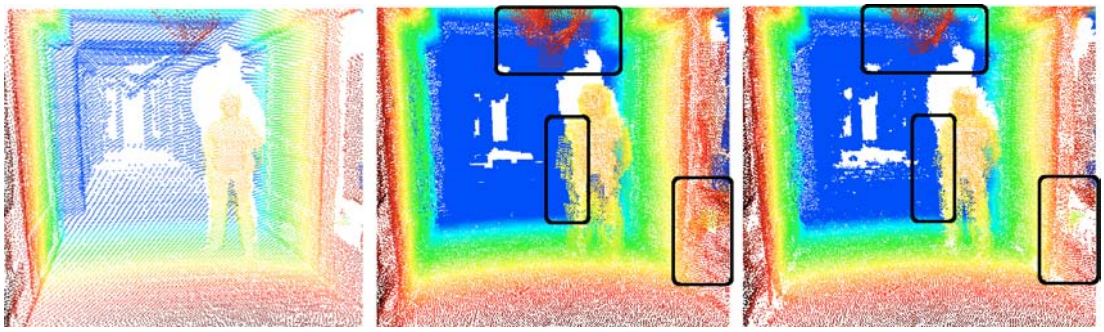


Figure 2.7: Left: the original point cloud. Middle and right: point cloud reconstructed based on an alphabet of size 10 and 70, respectively. The figure is adapted from Ruhnke et al. (2010)

(Takeuchi and Tsubouchi, 2006; Magnusson et al., 2007; Huhle et al., 2008; Kaminade et al., 2008; Stoyanov et al., 2012), loop closure detection (Magnusson et al., 2009a) as well as path planning (Stoyanov et al., 2010). However, as a grid based method, the cell size should be selected properly when using NDT. Choosing a cell size too large often lead to less accurate representation. On the other hand, too small cell size results in less meaningful PDFs due to few points within each cell, and requires more memory. Different discretization methods have been proposed for 3D-NDT, such as fixed subdivision, octree subdivision, iterative subdivision, see Magnusson (2009) for an intensive discussion. The accuracy of 3D-NDT for spatial representation has been analyzed in Stoyanov et al. (2011), and its consistency for spatial representation has been compared to alternative methods in Stoyanov et al. (2013).

An unsupervised model learning approach was proposed in Ruhnke et al. (2010). The main idea is to learn a set of repetitive 3D patterns from local scans first, and then use a combination of these patterns to approximate the raw data. Figure 2.7 illustrates the representation of a 3D scan with different sizes of alphabet. Although it provides a compressed representation, the learning process restricts its application in partially known and unknown environments. Furthermore, offline learning is only suitable for model compression, and some alphabet updating strategies (online learning) should be considered for robotic mapping. In other words, the alphabet should be updated when new 3D structures are learned in new scans.

When dealing with plane-rich environments, the map can be represented as a set of planar patches in the form of polygons (Hänel et al., 2003; Birk et al., 2009; Vaskevicius et al., 2010; Yao et al., 2011). For this purpose, the point cloud needs to be segmented into planar segments first (see Section 4.1). Then the points associated to each segment can be projected to their optimum fitting plane, so each segment can be represented as the optimal plane and a set of 2D points defined on the plane. Afterwards, the outline (known as boundary) of each segment is extracted for data compression. For organized point clouds, this can be done by utilizing the pixel-neighborhood information (see Section 4.1 for the concept of pixel-neighborhood information). Otherwise, the alpha-shape algorithm (Da, 2012) can be employed for finding an approximate shape of each segment. However, outlines constructed by aforementioned algorithms still contain more vertices than necessary, which can be further simplified. A detailed discussion

Figure 2.8: Left: a point cloud from a simulated robot with Velodyne 3D LRF in a scenario with ground-truth data from Mars. Right: Planar patches based representation of the point cloud. The figure is adapted from Birk et al. (2009).

on techniques for planar patch map construction can be found in Poppinga (2010). Planar patch maps have been applied in the domain of scan registration (Fischer and Kohlhepp, 2000; He et al., 2005; Pathak et al., 2010b) and SLAM (Harati and Siegwart, 2007; Kohlhepp et al., 2004, 2006; Weingarten and Siegwart, 2006; Pathak et al., 2010a). An example is shown in Figure 2.8. The main disadvantage of this method is that it is restricted to representing plane-rich environments.

2.3 Scan registration

Due to limited field of view, occlusions and sensing range, multiple 3D scans have to be acquired from different poses for sampling the environment completely. These scans should be registered in a common coordinate system in order to create a globally consistent model. Normally, the global coordinate system is fixed to that of the first scan, where successive scans should be aligned to one by one. Pairwise registration is the problem of finding a transformation (rotation and translation) which can align the (relative) new scan (which will be denoted as the source scan, known as current scan or data set in the literature) into the coordinate system of the (relative) old one (which will be denoted as the target scan, known as reference scan or map set in the literature). If the target scan has been registered to the global coordinate systems, the registration result will transform the source scan to the global coordinate system. In other words, registration cannot only be used for mapping but also for pose tracking, i.e., localization by updating the sensor's pose when the pose of last time step is known. Therefore, pairwise registration is one key issue of robotic mapping systems which has attracted increasing attention from the mobile robotics community in the past decade, and it is becoming more important with the rapid development of range sensors.

According to the search method being employed, registration algorithms can be classified as local or global. Local methods need both range sensor readings as well as a prior pose estimation from self-localization sensors such as odometry and IMU, assuming that the prior pose estimation is a sufficiently good initial guess for the relative transformation between the two scans. On the contrary, no prior pose estimation is required by global methods, the transformation is usually uniquely determined by feature correspondences where different kinds of features can be used.

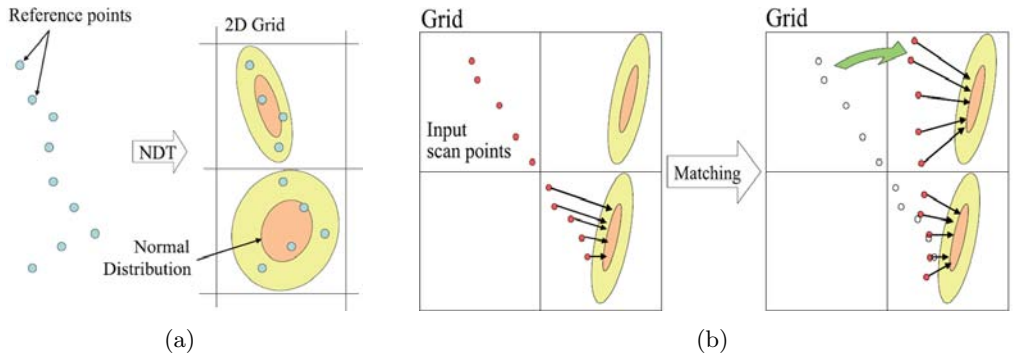


Figure 2.9: Explanation of the NDT registration process in 2D. (a): the point data of each cell are converted into normal distribution for the target scan. (b): the source scan is matched to NDT of target scan. The figure is adapted from Kaminade et al. (2008).

Local registration

One predominant registration method is the ICP algorithm (Chen and Medioni, 1992; Besl and McKay, 1992), which tries to find the transformation between two scans by minimizing the sum of square distances of corresponding points within them in an iterative manner. For each point on the source scan, the nearest point on the target scan is chosen as its correspondence. This, however, may introduce some outliers which should be removed using a predefined threshold. The algorithm is easy to be implemented and can often converge to the correct solution if given a sufficiently precise initial guess, but is time consuming because the nearest neighbor search is expensive. To make it fast, Zhang (1994) proposed a modified kd-tree algorithm for efficient neighbor search. Afterwards, a number of variants of ICP have been proposed (see Nüchter et al., 2007a,b; Segal et al., 2009) in order to to improve its robustness to outliers as well as its convergence speed. Besides working with self-localization sensors, ICP has been commonly employed to refine the registration result of global algorithms which can only give a coarse solution, see Dold (2005); Makadia et al. (2006) and Douillard et al. (2012) for examples.

Although ICP has demonstrated its influence for 3D shape matching, it still suffers from several drawbacks. First, the correspondences determined by smallest Euclidean distance (no matter point-to-point or point-to-plane) are not guaranteed to be correct, especially when the initial guess is far apart from ground-truth. Second, ICP is susceptible to local minima, i.e., it has a small convergence basin. Furthermore, a high number of iteration steps are usually needed until reaching the convergence. Although numerous efforts have been made as mentioned above, still no single approach can address all these shortcomings together.

An alternative local registration method is based on the NDT representation. After assigning a Gaussian distribution to each cell with more than a minimum number of points as in Eq. (2.1) and (2.2), the probability of a point at position $\mathbf{x}$ in the cell it lies in can be calculated by the corresponding Gaussian distribution $\mathcal{N}(\mathbf{u}, \mathbf{\Sigma})$. The

PDF is formulated as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{3/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)(\mathbf{x}-\mu)^{\mathrm{T}}) \tag{2.3}$$

In order to align the source scan to the target scan, Magnusson et al. (2007) proposed to use the Point-to-Distribution (P2D) metric. The scalar fitness function for a transform $\mathbf{T}$ is calculated as

$$s(\mathbf{T}) = \sum_{i=1}^{N} p(\mathbf{T}(\mathbf{p}_i)) \tag{2.4}$$

where $\mathbf{p}_i, i = 1, 2, ...N$ are points in the source scan, the PDF of each cell has effects only on those points located in it (points of the source scan after transformation $\mathbf{T}$). In other words, the fitness function is the probabilities for that all points in the source scan are lying on the target surface. Therefore, the source scan is aligned to the target scan when $s(\mathbf{T})$ is maximized. The idea, in 2D for better visualization, has been depicted in Figure 2.9. In order to do so, standard optimization methods such as Newton's algorithm can be used to iteratively solve the problem.

Since the PDF of each cell only affects points inside it, the grid resolution should be selected properly for registration. On the one hand, when the cells are too small, NDT may convergence to wrong poses when the initial guess from self-localization sensors is far away from ground-truth. On the other hand, if the cells are too large, NDT cannot give a proper representation of the surface structure, which also results failing cases. In order to deal with this problem, different discretization methods have been evaluated in Magnusson et al. (2007). Another NDT based registration algorithm is proposed in Takeuchi and Tsubouchi (2006), where a similar iterative optimization process is presented. For scan discretization, a smaller resolution is used for cells near the scanner and a larger resolution is used for cells further than a threshold distance from the scanner in the beginning of the iterative process. Afterwards, when the fitness function becomes stable, only the smaller resolution is employed until convergence. A comparison of ICP and NDT with regard to registration reliability and speed has been performed in Magnusson et al. (2009b), where both ICP and NDT are found to have advantages and disadvantages compared to each other.

Global registration

Different from local registration, most global registration algorithms rely on explicit feature correspondences. They usually consist the following steps:

1. feature extraction;
2. feature correspondence search;
3. globally consistent determination;
4. transformation refinement by local registration algorithm.

A number of different features have been proposed for 3D scans, such as spin-images (Johnson, 1997), Fast Point Feature Histograms (FPFH) (Rusu et al., 2009b), Depth-Interpolated Local Image Features (DIFT) (Andreasson and Lilienthal, 2010), and planar patches (Pathak et al., 2010b). Depending on the features, different distance metrics

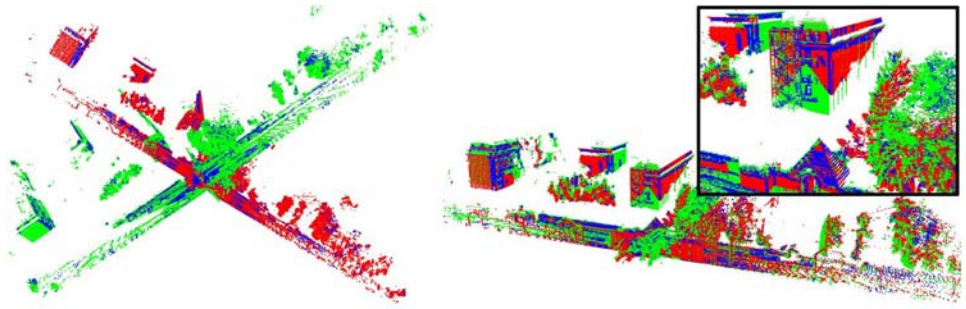


Figure 2.10: Left: two overlapping scans before registration (shown in red and green, persistent FPFH points are shown in blue). Right: the alignment results based on correspondence FPFH points with outliers rejection. The figure is reprinted from Rusu et al. (2009b).

can be used to measure similarities between them, where Euclidean distance is usually chosen. However, the correspondence found in pure feature space may induce false positives for various reasons: the feature descriptor is not completely distinctive, recurrent structures are present in the scenario, sensor noise, to name a few. Even a single false positive can cause wrong registration results. Therefore, those false positives must be removed from the correspondence set. A commonly used technique for false positive (noisy correspondence) rejection is the RANdom SAmple Consensus (RANSAC) (Fischler and Bolles, 1981) paradigm, which is an iterative method to estimate parameters (transformation here) from a set of observed data which may contain outliers. For 3D scan registration, the consensus of features is constrained by spatial relation, i.e., false positives do not agree with the transformation computed from true positives. Sometimes, the transformation computed from feature correspondences is not accurate enough for registration; fortunately, it is already good enough to be the initial guess for local registration algorithms. As a result, the transform is usually refined by local registration algorithms. ICP (and its variants) has been found to be a good solution for the refine phase, since it does not require the space division process compared to 3D-NDT.

Various algorithms have been proposed under the above steps. Here we give a short overview of them. In Rusu et al. (2009b), a FPFH is computed for each point which describes the local geometry around it. For the sake of selecting discriminative points, the distances from the mean FPFH to all the features are computed and approximated with a multi-dimensional Gaussian distribution. Then, according to statistical heuristics, features whose distance are outside of the $\mu + n\delta$ intervals are selected, where μ and δ denote the mean and standard deviation of the Gaussian distribution; n controls the discriminative level of the selected points. Afterwards, a sample consensus based method is employed to find a coarse transformation which can be refined by local registration methods. An example has been depicted in Figure 2.10.

Andreasson and Lilienthal (2010) proposed to solve the registration problem using the DIFT feature. An aLRF and a monocular camera are integrated on their mobile robot and externally calibrated. Local image features are computed from the image channel first, followed by finding visual correspondences between images. Then the depth for each visual feature is estimated by interpolation from the range channel.

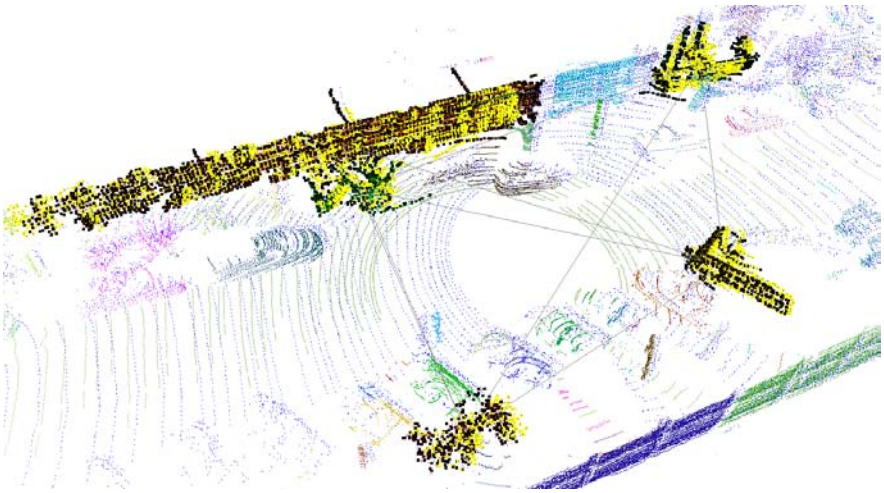


Figure 2.11: An example of segment correspondence based registration. The source scan is colored by its segmentation and the target scan is displayed in blue. The segments selected to compute the alignment are indicated by yellow and black points (these two colors respectively indicating the segment selected in the source and target scan to compute the alignment). The links plotted between these segments represent the segment-to-segment distances which are found to be consistent across the scan pair. The figure is reprinted from Douillard et al. (2012).

Afterwards, the matched depth-interpolated image features are employed to compute the transformation using a modified ICP algorithm. The main problem of this approach is that it needs special setups which can provide both range and color information at the same time.

Douillard et al. (2012) proposed a method for pairwise point cloud alignment by associating segments across scans. The ground and objects (trees, cars, traffic lights, etc.) on the ground are extracted as segments first (see Douillard et al., 2011). Then a shape distance metric is employed to generate segment association hypothesis, followed by filtering wrong associations based on spatial arrangement of segments. After fixing accurate segment associations, a modified version of ICP — where the search for point-to-point correspondence is constrained to associated segments — is run to get the final transformation. An example for segments matching is shown in Figure 2.11.

Geometric features have also been explored for point cloud registration, such as spheres, cylinders, as well as planar surfaces. In (Rabbani et al., 2007), geometric object models are fitted to segmented point clouds whose correspondences are determined by a constrained search algorithm. The transformation is then refined by the RANSAC paradigm. Since we are interested in planar segments based registration, related work on it will be introduced in Chapter 6.

Based on global descriptors, featureless global registration algorithms have also been proposed in recent years. Global consistency is solved by searching the transformation which aligns the underlying descriptor. EGIs based approaches fall into this category. An EGI is constructed by mapping the surface normal vectors onto a unit sphere, i.e., the Gaussian sphere. Then the Gaussian sphere surface is divided into small regions for the purpose of building a vector descriptor. EGIs are useful for surface representation,

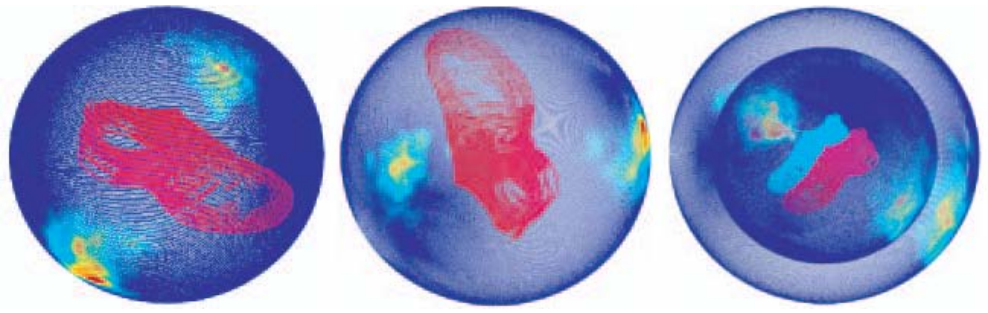


Figure 2.12: Left and middle: two different scans of a running shoe separated by a rotation, the corresponding EGI representation is shown on the sphere encompassing the shoe, encoded by heat map. Right: the two scans after rotational alignment, note that the correct alignment of the shoe corresponds to the correct alignment of EGIs. The figure is adapted from Makadia et al. (2006).

the detailed description of the technique can be found in Horn (1984). EGI has found its applications in the area of object recognition and attitude determination. Since the problem of point cloud rotational alignment is similar to object attitude determination, EGIs have been employed for point cloud registration in Dold (2005) and Makadia et al. (2006), where the rotational alignment is done by spherical correlation. A coarse-to-fine strategy was proposed in Dold (2005), where EGIs with coarse resolution (20 faces) are created for each point cloud first, then a rotation matrix $\mathbf{R}_0$ is found by traversing the whole grid of EGIs. Afterwards, EGIs with more faces are created for each point cloud, and a more accurate rotation matrix $\mathbf{R}_1$ is determined by traversing the grids near $\mathbf{R}_0$. Further fine resolution EGIs are constructed until a rotation matrix with predefined accuracy is reached. However, as stated in their paper, the algorithm sometimes failed to align overlapping scan pairs. In Makadia et al. (2006), the rotation estimate is obtained by traversing a discretized grid of SO(3) to find the rotation which maximized the correlation between EGIs. In order to make the correlation process fast, the spherical harmonics of the EGIs and the rotational Fourier transform are employed. However, the time complexity is still $O(N^3 \log^2 N)$, where N is the sampling bandwidth of each Euler angle. N should be set to at least 180 if the desired Euler angle accuracy is smaller than 1 deg, which means the complexity is still high. A rotational alignment of two scans based on EGI is depicted in Figure 2.12. Note that both above algorithms need a transform refine step.

Another featureless global registration algorithm has been presented in Stoyanov et al. (2012) which is based on 3D-NDT histograms. 3D-NDT histograms are originally proposed for loop closure detection (Magnusson et al., 2009a). The basic idea of the 3D-NDT histogram is to cluster the cells based on the surface's shape and orientation of the points inside each cell. The shape of each cell is determined by the covariance matrix of its Gaussian distribution. Suppose $\mathcal{N}(\mu, \mathbf{\Sigma})$ is the Gaussian distribution of a cell c, the shape of c is labeled as linear, planar or spherical according to the ratio of the three eigenvalues of $\mathbf{\Sigma}$. Furthermore, the orientation of the surface in each cell can be determined by the eigenvectors of $\mathbf{\Sigma}$. Afterwards, a 3D-NDT histogram is constructed by accumulating the cell number for each of the three shape categories. Each shape category can be split into bins with cells in each bin have a similar orientation. Then a

coarse rotation matrix is estimated by aligning two 3D-NDT histograms, which is used as an initial guess for 3D-NDT local registration.

2.4 Loop closure detection

The straightforward application of pairwise registration algorithms is to incrementally align new scans into a common coordinate system, thus creating a 3D model of the surrounding environment. In order to model the environment, a common way of robotic mapping is to let a robot move around and take scans regularly. However, any registration algorithm is tolerance-prone and will result in inconsistencies after a long distance, where the estimated pose may be far from ground-truth. Closed loops, i.e., a second encounter of a previously visited place, can be used to correct the pose estimation and distribute the error during pairwise registrations in order to build a globally consistent map. Therefore, loop closure detection has attracted increasing attention in recent years. In particular, numerous approaches have been proposed for vision sensors (see Ho and Newman, 2007; Angeli et al., 2008; Cummins and Newman, 2008; Milford and Wyeth, 2008; Konolige et al., 2010), while range sensors have attained relatively little attention so far. Since this thesis focuses on 3D scanner based outdoor mapping, only related work on range data will be reviewed.

A very basic way for loop closure detection is to make use of the estimated poses (from incremental pairwise registration or GNSS), i.e., detecting loops using the Euclidean distance between the current scan and all previous poses (known as proximity-based or heuristic loop detection). In Sprickerhof et al. (2011), two pre-defined thresholds are required to bound the computational cost. One is a distance threshold, only scans which are nearer than it are investigated for closing loops. The other one is of a minimum number of intermediate scans which is used to avoid loop closing within successive scans. A pair registration is performed between the current scan and all scans falling in the above two thresholds; and a loop closure is detected for each successful registration. A similar approach has been employed in (Pathak et al., 2010a), which has not been presented in detail.

A loop closure detection technique based on 3D-NDT histograms has been reported in Magnusson et al. (2009a), see Section 2.2 for 3D-NDT histogram construction; an example has been visualized in Figure 2.13. Rotation invariance of the histogram is achieved by aligning the corresponding scan to the dominant planar surface orientation, and multiple histograms are constructed for ambiguity dominant planar surface orientations. In order to handle different densities caused by the data gathering method, three different histograms are computed, i.e., one for low, one for medium, and one for high distance from the sensor origin. Then the differences between normalized histograms are measured by summing the Euclidean distances between each histogram according to the range intervals. A loop closure is detected if the difference between two histograms is smaller than a fixed threshold. The threshold is important and should be set properly. Automatic threshold selection has also been discussed in their paper.

Regional shape descriptors have been employed for place recognition for 3D point clouds in Bosse and Zlot (2010), where place recognition is defined as the problem of loop detection plus finding the relative pose, as an extension of their previous work on

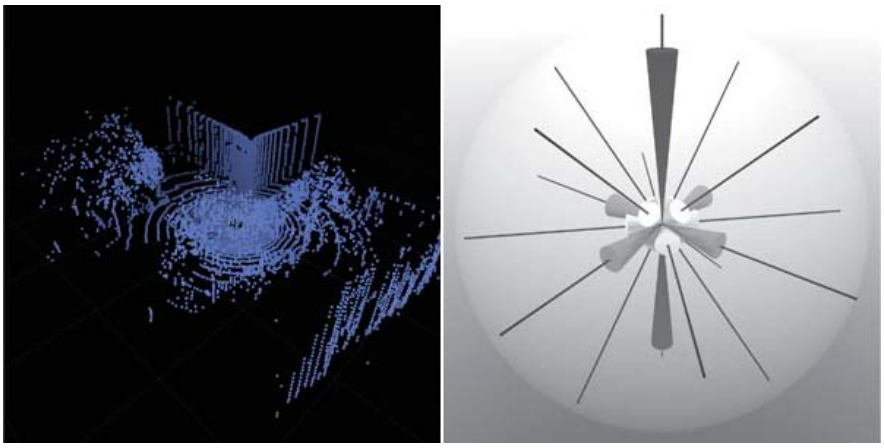


Figure 2.13: Left: the raw point cloud. Right: visualization of the planar part of a 3D-NDT histogram vector created from the point cloud. The thin black lines correspond to selected directions of the histogram, and the cones are scaled according to the values of the corresponding histogram bins. There are two cones for each direction, one for each side of the origin. Directions along z- and y-axis are shaded, and others are white. The figure is adopted from Magnusson et al. (2009a).

2D (Bosse and Zlot, 2009b). In their work, the structure of a local region is encoded as a descriptor built on the distribution of 3D points in the region. Three key properties have been considered for descriptor construction, i.e., shape, scale, and statistics. Afterwards, the regional descriptors are used in a nearest neighbor voting scheme to identify previously visited places. Its main limitation, as reported in the paper, is that the descriptors must be calibrated using aligned data with a sufficient number of labeled true positives.

Steder et al. (2010) proposed an algorithm which combines Bag-of-Words (BoW) and point features. Each point cloud is represented as a 2.5D range image. A training set is required to build a dictionary for the BoW approach using Normal Aligned Radial Features (NARFs) extracted in each range image. In order to limit the number of words, the learned features are grouped using k-means clustering. Then each scan is represented as a set of words from the dictionary, and a histogram is constructed by counting the number of each word. Loop closure is detected by measuring the Euclidean distance between the histogram of the current scan and that of all previous scans.

A more recent learning approach has been reported in Granström et al. (2011), which is extended from Granström and Schön (2010). In their work, each point cloud is described as a set of different (rotationally invariant) global feature descriptors such as volume, average range, and range histogram. The features are used as input of the adaptive boosting (AdaBoost) algorithm (Freund and Schapire, 1997), which can learn a "strong" nonlinear classifier as a linear combination of "weak" binary classifiers resulting from each feature space. Different publicly available datasets have been utilized to evaluate the approach compared with related work. Similar to Steder et al. (2010), the approach needs an offline learning phase, which limits its application for exploring unknown environments.

Remark on evaluation methods for loop closure detection algorithms: To achieve a low false alarm rate is important for loop closure detection, since even a single false loop can make the map unusable. Therefore, loop closure detection algorithms are commonly evaluated using the precision-recall rates (Bosse and Zlot, 2009b; Magnusson et al., 2009a; Steder et al., 2010; Granström et al., 2011), where *precision* (known as detection rate) is the ratio of true positive loop closures to the sum of loop closures, and *recall* is the ration of true positives to the number of ground-truth loop closures. Note that 100% precision equals 0% false alarm rate. False positive does not affect the recall rate but decreases the precision. Moreover, a high recall rate itself does not mean a good solution; instead, a high recall rate at a low false alarm rate (high precision) is desired. As a result, the algorithms are commonly compared with the recall rate at a fixed false alarm rate. Another issue, which should be considered for counting the recall rate, is how to define the ground-truth. Different from registration, where the ground-truth can be gathered without ambiguity using high precision localization sensors or manual registration, the loop closure ground-truth is usually determined by the distances between scan spots using a pre-defined threshold (see above papers). However, it is difficult to answer the difference between to use "3.0 m" or "3.1 m". What can be observed is that a too large threshold will induce false positive to the ground-truth because scans with quite different appearance will still be considered as from the "same" place. On the other hand, a too small threshold will result in low precision rate since two scans with sufficient overlapping with distance larger than the threshold will be considered as false positive. Actually, loop closure ground-truth determination should be built on appearance similarities (overlapping ratio), but not on distances between scan spots. For example, the appearance of two scans gathered at different sides of a wall or a door should be quite different and cannot be registered, while their appearance can be quite similar when gathered far away in a same room. To deal with this problem, some overlapping metrics, such as the metrics used for predicting false registrations in Pathak et al. (2010c) can be investigated. However, this has not been addressed according to the literature, which is still an open research area.

2.5 Global relaxation

When a loop closure has been detected, two transformation metrics can be computed for the first scan (Ψ_1) and last scan (Ψ_N) in the loop: one (${}^{N}_{1}\mathbf{T}$) is from *cumulative* registration, and the other (${}^{1}_{N}\mathbf{T}$) is from *direct* registration, where

$$ {}^{N}_{1}\mathbf{T} = {}^{N}_{N-1}\mathbf{T}\,{}^{N-1}_{N-2}\mathbf{T}\cdots{}^{3}_{2}\mathbf{T}\,{}^{2}_{1}\mathbf{T} \tag{2.5} $$

If every pairwise registration is perfect, the following relation can be yield

$$ {}^{N}_{1}\mathbf{T} = ({}^{1}_{N}\mathbf{T})^{-1} \tag{2.6} $$

Unfortunately, no registration algorithm is tolerance-prone, resulting in pose difference for the last scan estimated from cumulative and direct registration. Evidently, the pose estimation from direct registration, i.e., aligning the last scan to the first, is more confident than the cumulative estimation, thus can be used to correct the past

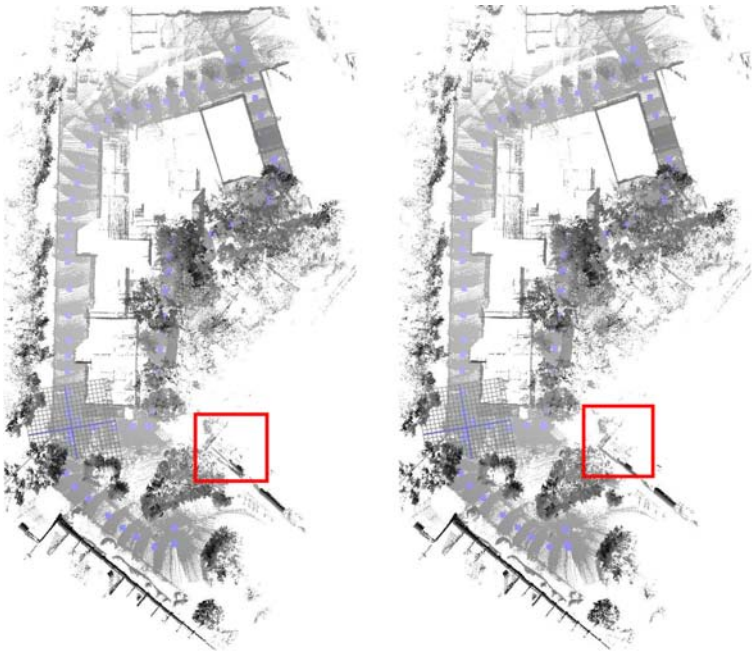


Figure 2.14: Left: registration result of 77 scans before global relaxation. Right: final model with loop closing and global relaxation. Difference between before and after relaxation can be found in the rectangle-marked area. The figure is adapted from Nüchter et al. (2007b).

path in the loop, which is called global relaxation (known as error distribution, loop optimization, and map refinement).

The idea of using closed loop for map refinement has been researched in 2D mapping (Lu and Milios, 1997; Frese et al., 2005; Thrun and Montemerlo, 2006). Along with the emergence of 3D range sensors and their applications in robotic mapping, 3D map relaxation has been a hot research question and different solutions on it have been reported. The solutions can be classified into two major categories. The first category addresses the global consistency in the feature space, where relaxation is solved by minimizing the overall error between associated features. Feature association can be determined using extended Kalman filter (Kohlhepp et al., 2004; Weingarten and Siegwart, 2006), constrained global search Pathak et al. (2010a), etc. Clearly, reliable feature extraction and association is a necessary condition for such solutions.

The second category iteratively reforms the path by matching each scan to its adjacent scans within a distance until reaching a convergence. Please note that there is no constraint on the matching method, both global and local registration algorithms can be employed. However, according to the literature, refining a map built upon feature based global registration is usually done by feature association, and refining a map resulted from local registration is usually done using methods in this category. For this purpose, the scans are usually put into a pose graph (known as pose network), where each node represents a scan, each edge denotes the estimated transformation between two nodes connected by it. The graph-based formulation is proposed in Lu and Milios (1997), which is well known as LUM style SLAM. It is then extended to 3D

by Borrmann et al. (2008). A nice tutorial on graph-based methods can be found in Grisetti et al. (2010).

2.6 Autonomous exploration

Most state of the art robotic mapping systems rely on human-robot cooperation, i.e., a mobile robot is asked (or even manually controlled) to move from the current viewpoint to the next viewpoint which is usually planned by a human-operator. Although human operations can be effective, such systems have the following problems: First, human beings can only give qualitative solutions judged from the senses. Second, operating a robot in unknown cluttered environments, for example in search and rescue scenarios, is very challenging. Doroodgar et al. (2010) reported that "operator can become stressed and fatigued very quickly due to a loss of situational awareness". Therefore, fully autonomous exploration is desired, i.e., by simply defining a region of interest (ROI) in 3D space and leaving the details of the procedure to a mobile robot. While the implementations of SLAM have improved greatly in recent years, autonomous exploration of 3D unknown environments has been given comparatively little attention. Autonomous exploration can be accomplished by repeating the following steps:

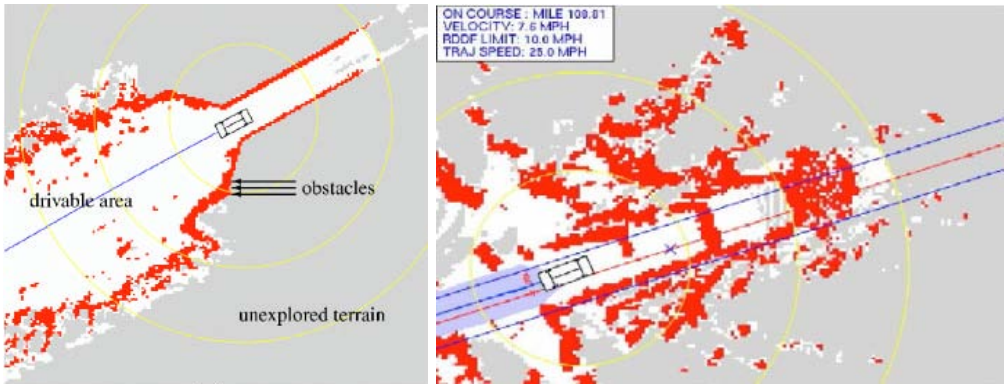
1. conducting a scan;
2. aligning the scan to the global coordinate system;
3. extracting the drivable surface;
4. planning the next viewpoint with drivable ability constraint;
5. moving the robot to the NBV.

The procedure is terminated iff the ROI has been completely covered. Among the 5 steps, step 1 and 2 have been discussed in Section 2.1 and 2.3. A large amount of literature can be found for step 5, since it equals to the problem of path planning in known environments (drivable surface is determined in scanned area). We will discuss the two remaining issues in the following parts of this section.

2.6.1 Drivable area detection

Drivable area detection (known as terrain traversability analysis) has been used as a means for navigating a ground robot within environments of varying complexity, and it is becoming more and more important as the trend to employ mobile robots into environments of increasing complexity. It has been addressed as a binary classification problem, i.e., each part of the terrain is classified as either drivable or non-drivable. Terrain classification with regard to drivability can be determined if a complete 3D environment model is available, however, the model is normally the goal of an exploration which does not exist during the mapping procedure. According to the literature, there are two main research directions in this area, namely geometry-based or appearance-based. Appearance-based methods are mostly proposed for vision sensors, which is beyond the scope of this thesis. Therefore, we only give a brief overview on geometry-based approaches which are proposed for range sensors.

Up to now, the most impressive results regarding to drivable surface detection were achieved during the DARPA Grand Challenge 2005, aiming to create the first fully



(a) (b)

Figure 2.15: Two typical occupancy map based terrain classification results. (a): successful; (b) failed. The figure is adapted from Thrun et al. (2006).

autonomous ground vehicles capable of completing a substantial off-road course within a limited time. In the end, five vehicles out of 23 finalists successfully completed the course of a 212 km desert drive (Iagnemma and Buehler, 2006a,b). However, everything is a double-edged sword: despite its impressive success, each vehicle in the race has been equipped with multiple on-board computers which allows to run computational expensive algorithms at real-time. Moreover, those vehicles have a significant payload, allowing them to carry a set of high end sensors, which can provide a rich and accurate information of the surrounding environment. Up to now, those experimental platforms are still far beyond low-cost robots.

Stanley (see Figure 1.1(b)), the autonomous vehicle won the Challenge, was developed by the Stanford Racing Team (Thrun et al., 2006). It is equipped with five 2D LRFs mounted on the roof, tilted downward with different angles to scan the road ahead. The obtained point clouds are then aligned to a global coordinate frame according to the pose estimation of the vehicle, resulting a 3D point cloud which is represented using occupancy grids. The drivability of each grid is determined by the local elevation difference in it. If two nearby points whose vertical distance $|z_i - z_j|$ exceeds a critical threshold, it is marked as non-drivable. Unfortunately, this simple method yields bad results due to pose estimation error; it is especially sensitive to roll/pitch angle error. Two typical classification results have been depicted in Figure 2.15, including a false positive result. To deal with the high false positive rate (12.6%), the drift of the pose error over time is modeled as a first-order Markov model, which leads the terrain classification problem to a probabilistic test for the presence of an obstacle. However, the performance of the probabilistic test heavily depends on a couple of parameters. The system is hence trained by a human driver, who is instructed to drive only over obstacle-free terrain. This data-driven parameter tuning resulted a significant decrease of the false positive, which is then only 0.002%.

Two simple yet sufficient algorithms have been presented in Nüchter et al. (2006) and Poppinga et al. (2008a). In the former, labeling floor points in 3D scans is resolved by computing gradient within cylindrical coordinate systems. According to the

gradient, each point is labeled as "ground", "object" or "ceiling". Afterwards, neighboring ground points are merged together into drivable surface by region growing. In the latter, a Hough based terrain classification algorithm is proposed. By designing a parameter space, the drivable surfaces can be easily detected regarding to the number of hits in the bins corresponding to drivability. Then a decision tree is employed to increase robustness which allows the algorithm to handle sensor noise. More than the binary distinction of drivable/non-drivable ground, a finer classification has been performed.

A learning approach for traversability classification has been proposed in Howard et al. (2006), with stereo vision as the primary sensor. According to the distance from the robot, the scene is divided into four zones, namely underfoot, near-field, midfield, and far-field. For the underfoot zone, the terrain geometry is already known since it is present in previous maps. For the near-field zone, the range data from stereo vision is used to build a local elevation map. Then a Support Vector Machine (SVM) is employed to learn a proper classifier, with the parameters estimated by cross validation. However, the learned classifier still has an error rate of 14% on a test set in the paper. Then the mid- and far-field traversability is learned by fusing the result of near-field and visual appearance similarities between adjacent cells. The bottleneck of their approach is the time-consuming learning phase cost by SVM.

2.6.2 Viewpoint planning

Generating safely reachable viewpoints from previously acquired data, known as NBV planning (Connolly, 1985), is an extension of the NP-hard art-gallery problem which involves employing the minimum number of guards to observe the whole gallery. The art gallery problem has been researched extensively, with most contributions presenting algorithms operating on polygons in 2D space (O'Rourke, 1987). Even considering the art-gallery problem in 3D space, there are two main differences between the art-gallery and NBV planning problem: First, the map is prior known in the former which is not the case in the latter. Second, there is no constraint for safe navigation in the former which should be considered for the robot in the later. Therefore, algorithms proposed for the art-gallery problem are not applicable to autonomous exploration, and researchers from the robotics society have presented methods in recent years.

Nüchter et al. (2003) presented an online greedy version based on the algorithm of González-Baños et al. (2000), resolving NBVs as a set cover problem in a randomized manner. Candidate sensor positions are generated randomly in the interior of the polygon (map), then calculate the visible part of the polygon for each generated position, and approximate the set cover problem using those visible parts. Clearly, this approach is limited to 2D movements which cannot be satisfied on outdoor terrain.

A two stage approach is proposed in Blaer and Allen (2007) with the aim to construct dense and detailed 3D models. The first stage of the modelling process utilizes a 2D ground map (available a-priori) to derive a set of views, which is close to Nüchter et al. (2003). Based on this set of views, an initial model of the scenario can be constructed. Then the initial model and views in the first stage are employed to plan more-refined views that resolve occlusions occurred in the first stage, based on building voxel-grids from acquired data and ray traversal. The approach has been used for

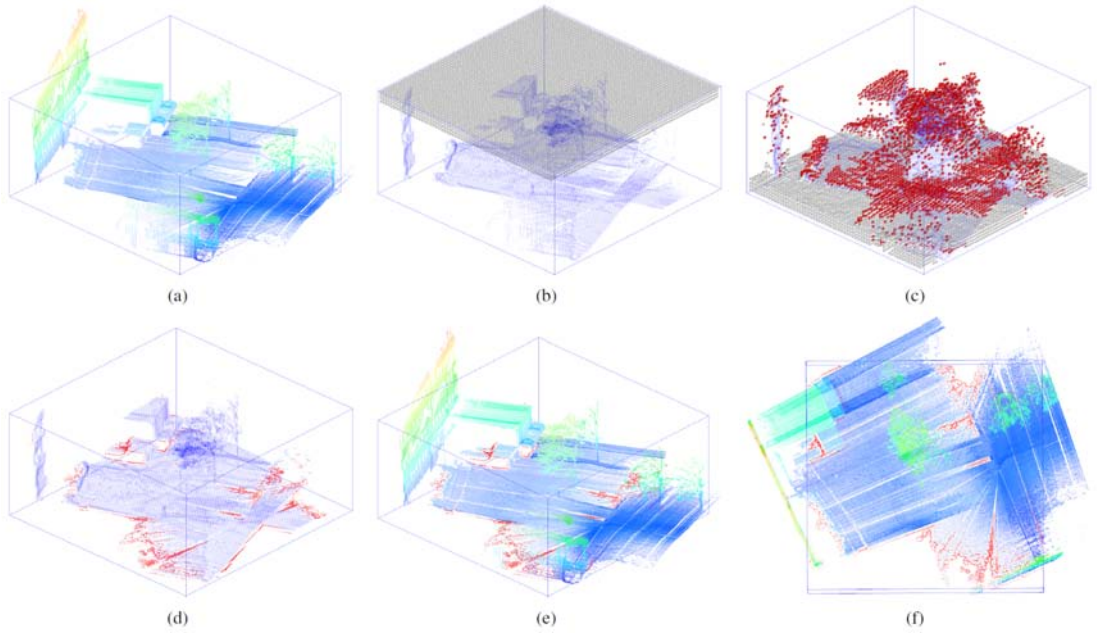


Figure 2.16: Overview of our NBV planning procedure. (a): predefined bounding box b and the initial point cloud from the platform shown in Figure 2.3(a). (b): 16 k particles in gray being poured over the down-sampled cloud in blue. (c): particles that have collided with any point are shown in red, others remain gray. (d)-(f): overlay a visualization of the information gain grid over sparse and dense point clouds, showing cells with high information gain in red.

modeling some historical buildings, and has been found to be efficient. However, the requirement of a known 2D map restricts its application in unknown environments.

A more recent approach has been reported in Nagatani et al. (2010), where the map is represented by MLS (MLS is introduced in Section 2.2). Based on this representation, the target region is divided into three classes, namely scan-completed, low-resolution, and unscanned. Then viewpoints are evaluated with regard to how much (F_1) of unscanned and low-resolution region will be changed to scan-completed region, and how much (F_2) unscanned region will be changed to low-resolution region, after taking a scan at each pose. Depending on the importance of exploring unscanned region and detailing low-resolution region, different weights can be assigned to F_1 and F_2. Then a hill-climbing search algorithm is employed to calculate the change after a new scan, as it is time-consuming to use the ray-tracing method. However, as a local search algorithm, hill-climbing is not guaranteed to find the best possible solution.

A frontier-based approach is commonly employed in NBV problems (Yamauchi, 1998; Shade and Newman, 2011; Mobarhani et al., 2011), which results sensor-poses located between known and unknown regions. Evidently, these poses are reachable, because a trajectory can be planned through known parts of the environment. Furthermore, given poses orientating towards unmapped part of the environment allows the sensor to deliver valuable information, boosting the exploration process. In 2D case, the frontier can be easily obtained using grid cells, i.e., "frontier cells are defined as unknown cells adjacent to free cells and this way a global frontier map can be pro-

duced" (see Mobarhani et al., 2011). This, however, is not a trivial task in 3D space, since to update occupancy grid map with fine scale in 3D is already computational costly, let alone searching frontier in it. Techniques such as multi-resolution occupied grids map have been proposed to bound the updating cost, it should be interesting to investigate whether such representations are suitable for frontier finding.

We have also presented a promising approach in Adler et al. (2012), inspired by Holenstein et al. (2011) which aims at creating watertight 3D models of real-world environments. Actually, watertightness is not only a desirable property for completely reconstructed models, but also a helpful test for finding gaps that have been remained throughout the mapping process. The challenge here is to find gaps of a desired minimum size in a small amount of time. Our algorithm requires a predefined bounding-box b that contains both the robot and the environment to be mapped. Then b is discretized into a 3D uniform grid which stores a scalar value in each cell indicating the information gain achievable by scanning it. After an initial point cloud is populated into b, gaps are detected by using a particle system that simulates pouring water in the form of fine-scale particles. Whenever a particle first collides with a point of the cloud and later arrives at b's bottom plane, it has successfully passed through a gap, and the gap is the last collision position. Each cell in the information gain grid stores how many points have passed through a gap in it, and cells with large numbers intuitively represent possible viewpoints. The process is visualized in Figure 2.16.

2.7 Semantic mapping

Until now, we are only discussing how to use a mobile robot to build a coherent metric model of its surroundings, at the same time localize itself within this model, no matter the process is fully autonomous or rely on human-robot cooperation. Unfortunately, only simple metric information is not sufficient for real autonomy with regard to interacting with and reasoning about the environment. For example, comparing some metric based command like "Move to position $(1.0, 1.0, 1.0)$ with RPY angle $(0.0, 0.0, 0.0)$ in the global coordinate frame, and then pick up the object at position $(0.5, 0.0, 1.2)$ in your relative coordinate frame", some higher level command like "Move to the table in the kitchen and pick up the cup on it" is preferred. This, however, need a map between the semantic and spatial domain, which has been termed using keywords: semantic labeling, semantic categorization, semantic mapping, scene analysis, object detection and interpretation, etc.

Semantic mapping is usually learned from a set of features (in metric map) and a set of labels. This can only be done using *supervised learning* methods, where a human is required to annotate a sufficient number of training examples. The reason for this is that "semantic" labels can only be defined by humans but not robots. Different learning methods have been employed for object classification and labeling from 3D point clouds, here we give a short review of recent works.

In Nüchter and Hertzberg (2008), large meaningful structures in indoor scenarios (floor, wall, door, and ceiling) are interpreted first using a constraint network, which is built upon a common-knowledge of their spatial relations. Afterwards, objects are segmented by removing those points belonging to scene structures. The remaining

Figure 2.17: Semantic classification for an outdoor scenario. Left: ground-truth. Right: classification result. The figure is adapted from Paul et al. (2012)

points are projected onto a 2D image plane, and the contour of each object is extracted from a binarized image using a contour following algorithm. The resulting contours are described as rotation invariant features and put into a SVM classifier. The bottleneck of this approach is obvious: First, the approach is restricted to indoor applications due to the rely on structure interpretation. Second, because image processing techniques is employed for contour extraction and description, each object should be scanned with different perspectives in order to achieve high true positive rate, which is a time-consuming task.

In Eich et al. (2010), the point cloud is segmented using a region growing algorithm first. Then the concave shape of each segment, as polygon, is determined by the alpha-shape approach. Afterwards, a shape descriptor is constructed for each rectangular polygon, based on its area, its maximum extension , and the relationship with other entries. As a pre-condition of this approach, some independent entries should be defined (spatial axiom, such as the floor), which serves a base for the reasoning system. The spatial relationship is resolved iff a spatial axiom is reached. This approach has similar limitations as Nüchter and Hertzberg (2008), with an additional restriction to only objects with perfect rectangular appearances.

Conditional Random Fields (CRFs) have been employed in Rusu et al. (2009a) to label objects in a specified kitchen scenario, wherein a 2-level feature is constructed for each object. In order to separate the objects into segments, a major planar decomposition step and a region growing step is performed on the input scan. Then the horizontal and vertical planes are clustered together and a level-1 feature is built for each planar segment. Afterwards, similar to Nüchter and Hertzberg (2008), structural components of the environment are removed from the data. Thereafter, a level-2 feature is computed for each remaining planar region after detecting fixtures on it, where fixtures play a vital role. In the end, a sufficient number of manually labeled objects and their level-2 features are used to train the CRF classifier. However, this approach has been evaluated only in a kitchen scenario, it would be interesting to see how it can be extended to more general environments.

A promising approach has been reported recently in Paul et al. (2012), which is proposed for outdoor environments. In their work, each new scan is represented using triangular mesh first, which is later segmented into mesh regions. Afterwards, a high dimensional feature vector is computed for each segment based on spatial characteristics. These feature vectors and ground-truth class labels are then fed into the learning

algorithm of Gaussian Process (GP) multi-class classifier, wherein principal component analysis is employed for dimensionality reduction to decrease the computational complexity. The probabilistic formulation of GP classification allows it to provide an uncertainty estimate for the distribution over exist labels. As a result, a high uncertainty in the output can vote for a category not modeled during the learning phase, which can be reported to its supervisors for incremental learning. See Figure 2.17 for an classification example.

2.8 Public resources

There have already been a number of open-source codes and datasets related to 3D outdoor metric mapping, which have been listed in Table 2.1 and Table 2.2 respectively. Please note that the survey is probably incomplete.

2.9 Summary

Recent related work on different subtasks of outdoor metric mapping systems have been briefly reviewed in this chapter, namely data gathering, map representation, scan registration, loop closure detection, global relaxation, drivable area detection, viewpoint planning, as well as semantic labeling. Clearly, this thesis cannot cover all the key techniques in full.

Considering the rapid development of new 3D range sensors, gathering accurate range data in real-time won't be a problem in the near future. Therefore, we would like to build a system for range data with high accuracy regardless of the method for gathering datasets, which allows us concentrate more on data processing. As aforementioned, there have already been many alternatives for map representation, each has advantages and disadvantages for different applications. In order to develop a mapping framework for general purpose, we choose to use the very basic form of representation, i.e., point clouds. For the resulting map, the representation method can be selected properly from existing alternatives for a specific task. Additionally, fully autonomous exploration and semantic mapping is obviously the next step of a reliable human-robot cooperation mapping system.

Scan registration is voted as the main topic for this thesis among the three remaining research areas for the following reasons: First, it is the fundamental of global relaxation. Second, loop closure can be resolved heuristically based on an accurate registration algorithm as has been done in Sprickerhof et al. (2011). Third, promising results of global relaxation have been achieved using pose-graph, which is independent to the underlying registration scheme. Fourth, fast and accurate global registration is still an open question and a hot research topic.

On the one hand, a fast and accurate global registration algorithm for various environments is too aggressive, as they have quite different appearances and geometric characteristics. On the other hand, an algorithm for a too specific kind of environments is much easier, but results in a too narrow application area. As a compromise, we decide to focus on plane-rich environments based on the following observations: First, planar surfaces are abundant in urban environments where service robots can be employed

for many tasks, hence the approach can be utilized to liberate service robots from indoor environments which has a prominent impact on the problem of aging population. Second, corresponding planar surfaces can provide necessary geometric constraints for transformation computation. Third, correspondences between planar segments can be determined globally based on their attributes such as area, plane parameter, plane parameter uncertainty, 2D shape, etc.

The proposed registration approach is detailed in the following chapters, including point cloud segmentation in Chapter 4, planar segment area calculation in Chapter 5, and planar segment correspondence search in Chapter 6.

Table 2.1: Publicly available libraries. Overlaps can be found among them, for example, PCL, perception_oru, and OctoMap have their wrapper version in ROS.

software	link	description	publication(s)
Robot Operating System (ROS)	`http://www.ros.org`	A collection of libraries and tools for robot applications which is not restricted to mapping.	Corresponds to each specific package.
Point Cloud Library (PCL)	`http://pointclouds.org`	An project for point cloud processing which is not limited to mapping.	Corresponds to each specific package.
The 3D Toolkit (3DTK)	`http://slam6d.sourceforge.net`	ICP registration based 6D SLAM and other tools.	Nüchter et al. (2007b); Nüchter and Hertzberg (2008)
perception_oru	`http://code.google.com/p/oru-ros-pkg/`	Implementation of P2D and D2D 3D-NDT registration.	Magnusson et al. (2007); Stoyanov et al. (2012)
octocopter	`http://github.com/benadler/octocopter/`	Next best viewpoints planning.	Adler et al. (2012, 2013)
OctoMap	`http://octomap.github.com`	A probabilistic 3D mapping framework based on octrees	Wurm et al. (2010)
OpenSLAM	`http://www.openslam.org/`	A collection of robust SLAM algorithms.	Corresponds to each specific package.
Mobile Robot Programming Toolkit (MRPT)	`http://www.mrpt.org/`	A collection of libraries for robotic applications (not limited to mapping).	Corresponds to each specific package.

Table 2.2: Publicly available datasets.

software	link	description	publication(s)
Robotic 3D Scan Repository	`http://kos.informatik.uni-osnabrueck.de/3Dscans/`	A collection of point clouds from robotic experiments.	Corresponds to each specific dataset.
Barcelona Robot Lab Dataset	`http://www.iri.upc.edu/research/webprojects/pau/`	3D scans and on-board robot imagery, as well as imagery from a camera sensor network, gathered at the UPC Nord Campus in Barcelona.	Valencia et al. (2009)
PCL dataset repository	`http://svn.pointclouds.org/data/`	A collection of dataset in PCL.	Corresponds to each specific dataset.
ASL Datasets Repository	`http://projects.asl.ethz.ch/datasets/doku.php`	Aiming to evaluate scan registration algorithms in specific environments and conditions with provided ground-truth.	Pomerleau et al. (2012)
3D Scan and Map dataset for OctoMap	`http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/`	3D scans gathered by the Freiburg AIS group.	Wurm et al. (2010)
Jacobs Robotics Dataset	`http://robotics.jacobs-university.de/projects/3Dmap`	A collection of datasets gathered by the Jacobs Robotic Group.	Pathak et al. (2010b,a)
Canadian Planetary Emulation Terrain 3D Mapping Dataset	`http://asrl.utias.utoronto.ca/datasets/3dmap/`	A collection of 3D scans gathered at two unique planetary analogue rover test facilities in Canada.	Tong et al. (2012)
MRPT dataset repository	`http://www.mrpt.org/robotic_datasets`	A collection of datasets in MRPT.	Corresponds to each specific dataset.

Chapter 3

Experimental datasets

> Success depends upon previous preparation, and
> without such preparation there is sure to be failure.
>
> Confucius

In this chapter, we introduce four datasets which are employed to evaluate the proposed algorithms in later chapters: The first dataset, obtained in a scrap yard scenario with our custom-built platform, is presented in Section 3.1 where the platform is also introduced. Then a dataset gathered at the UPC Nord Campus in Barcelona is given in Section 3.2. Afterwards, a dataset acquired at Disaster City, Texas is detailed in Section 3.3. The fourth dataset, which contains a large urban scenario in the Bremen city center, is introduced in Section 3.4. Except the first dataset, the other three are publicly available. Slight preprocessing has been performed on the second and third dataset for their usage in this work, see corresponding sections for detail. To make the presentation self-consistent, the index of the first scan in each dataset has been set to 1. For gray scale range images, dark and light are used for representing short and far range, and white for invalid points.

3.1 The "Scrap Yard" dataset

The dataset was obtained using our custom-built 3D perception platform TOMAR (TAMS Outdoor MApping Robot) in a scrap yard scenario in Hamburg, Germany. TOMAR is introduced in Section 3.1.1, followed by the scenario in Section 3.1.2.

3.1.1 The platform TOMAR

TOMAR is built upon a Pioneer 3-AT robot (Adept MobileRobots, 2012) which is a four wheel drive robotic platform, can be operated on rough-terrain outdoor, and has an on-board computer. ROS is employed to control the robot. The platform is equipped with an aLRF for 3D perception, which is constructed by a FLIR PTU-D48E Pan-Tilt Unit (PTU) (FLIR, 2013) and a Hokuyo UTM-30LX LRF (Hokuyo, 2009), see Figure 3.1. UTM-30LX is a compact and accurate laser scanner designed for both

(a) (b)

Figure 3.1: (a): FLIR PTU-D48E; (b): Hokuyo UTM-30LX.

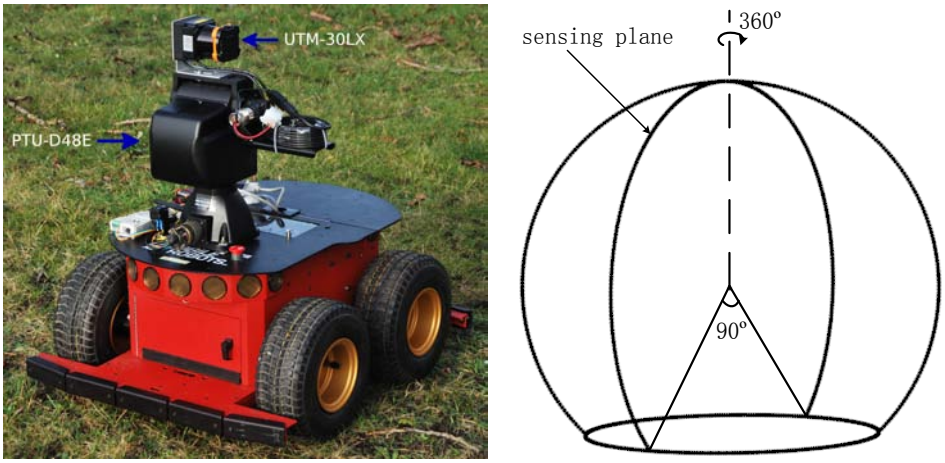
Table 3.1: PTU-D48E specifications.

Pan Position Resolution (°)	0.006
Tilt Position Resolution (°)	0.003
Min Pan Speed	0.006°/s
Max Pan Speed	100°/s
Min Tilt Speed	0.003°/s
Max Tilt Speed	50°/s
Pan Range	N×360°-continous
Tilt Range	-30°/+90°
Operating Voltage	12-30VDC

indoor and outdoor applications, it can detect objects within range from 0.1 to 30 m in a 270° FoV (up to 0.25° angular resolution). A relative low power consumption – 8.4 Watt (12 Volt 0.7 Ampere) – allows it to be used on mobile robotic platforms. In addition, it can obtain both depth and intensity information from the received laser pulse. The specifications of D48E are illustrated in Table 3.1, it was chosen to actuate the LRF due to the following three reasons:

1. It is designed for both indoor and outdoor applications.
2. High accuracy angle positioning, the pan and tilt position resolution are 0.006° and 0.003° respectively.
3. Most importantly, it features internal wiring with slip-ring for 360°-continuous pan, a single connector carries all pan-tilt and payload signals, providing simple and reliable installation and operation.

Different configurations of the PTU result different scan mechanisms, which have been named *pitching scan, rolling scan, yawing scan* and *yawing scan top* in Wulf and Wagner (2003). We choose to use the yawing scan top method, i.e., the UTM-30LX is installed horizontally on the top bracket of D48E, with its "sensor front" pointing upwards as shown in Figure 3.2(a). This results in a full horizontal FoV and a 135°



(a) (b)

Figure 3.2: (a): Our aLRF equipped platform. (b): Field of view of the aLRF.

vertical FoV, as shown in Figure 3.2(b). During each scan, all the 2D scan slices joint at one point, i.e., the sensor front. The *organized point cloud* (see Section 4.1) is constructed as follows: treat each half 2D scan slice (from the sensor front to the begin or end step) as a row, and adjacent half slices are stored as adjacent rows in the resulting image-like structure.

In general, the platform has the following advantages:

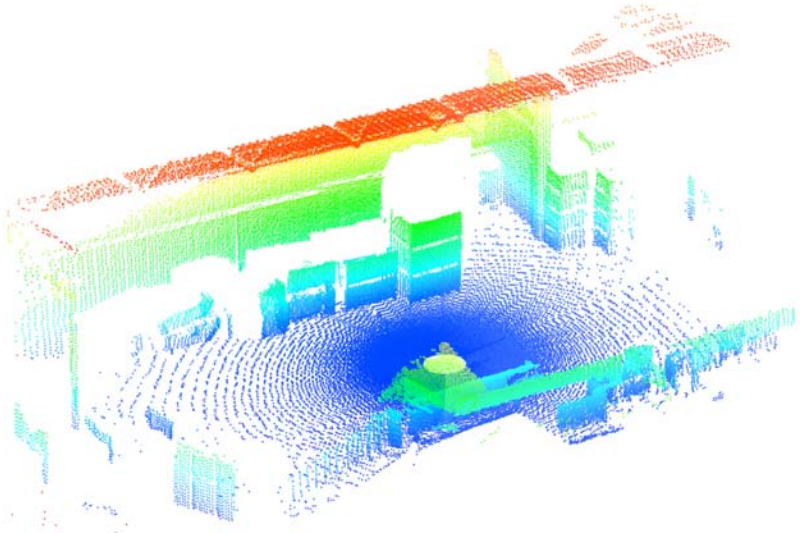
1. It delivers nearly omni-directional range information of the environment, which is good for scene understanding.
2. The point cloud is well organized, which provides good latitude and longitude neighborhood information. In other words, there is no intersection between the 2D scan slices except at the sensor front point.
3. It is flexible and quickly to pay attention to a given direction with a fine resolution scan without turning the robot.
4. It can also obtain corresponding omni-directional intensity images which has the texture information of the scanned scene.

3.1.2 The dataset

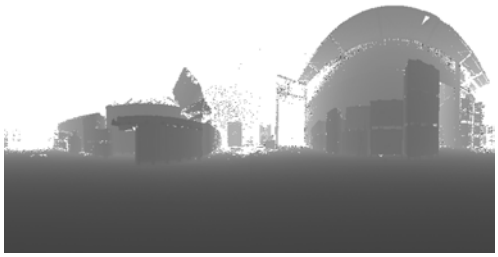
The dataset was acquired in a scrap yard situated in the north-west of Hamburg, Germany. Two photos of the field are shown in Figure 3.3. The scrap yard is chosen as a benchmark for mapping algorithms for the following reasons:

1. The environment features many unstructured geometries in the form of piles of scrap metal.
2. Most container's shapes have deviated from their original rectangular shapes after countless collisions with forklifts and scrap metals. Furthermore, most of their walls were built using trapezoidal sheet metal, making plane detection even more challenging.

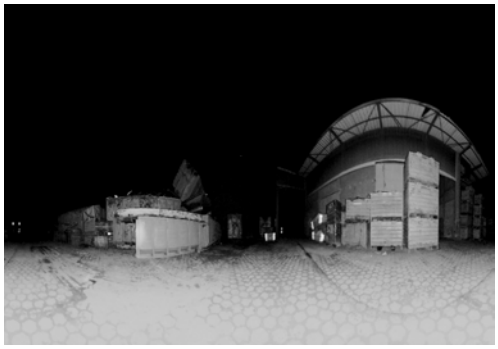
Figure 3.3: Two photos of the scrap yard field.



(a) scan 10, side view



(b) scan 10, range image



(c) scan 10, intensity image

Figure 3.4: One typical scan of the scrap yard dataset. (a): as point cloud colored by height; (c) as gray scale range image; (d): the corresponding gray scale intensity image.

3. Many forklifts, cranes and delivery cars were passing by during the experiment, adding noise to the scans.
4. Some kinds of metals exhibiting high reflectivity could not be sampled by the scanner.

Figure 3.5: The gray scale range image of the scans in the scrap yard dataset.

5. The data was gathered under direct sunlight outdoors, inducing bad measurement accuracy according the the sensor specifications.

The robot started in a storage room and then came out to an open area after 7 scans. In total, 19 useful scans were obtained. The pan resolution and the 2D scan resolution was set to 0.46° and 0.25° respectively, hence resulting organized point clouds with 778 rows and 540 columns (420, 120 points in total). Besides the depth information, the intensity information has been also provided in the dataset. A typical point cloud, together with its corresponding range and intensity images, has been visualized in Figure 3.4; note the similarity between the images. Such a dataset can not only be used for evaluating registration algorithms but also for fusing depth and intensity information. The gray scale range and intensity images of the dataset are shown in Figure 3.5 and Figure 3.6.

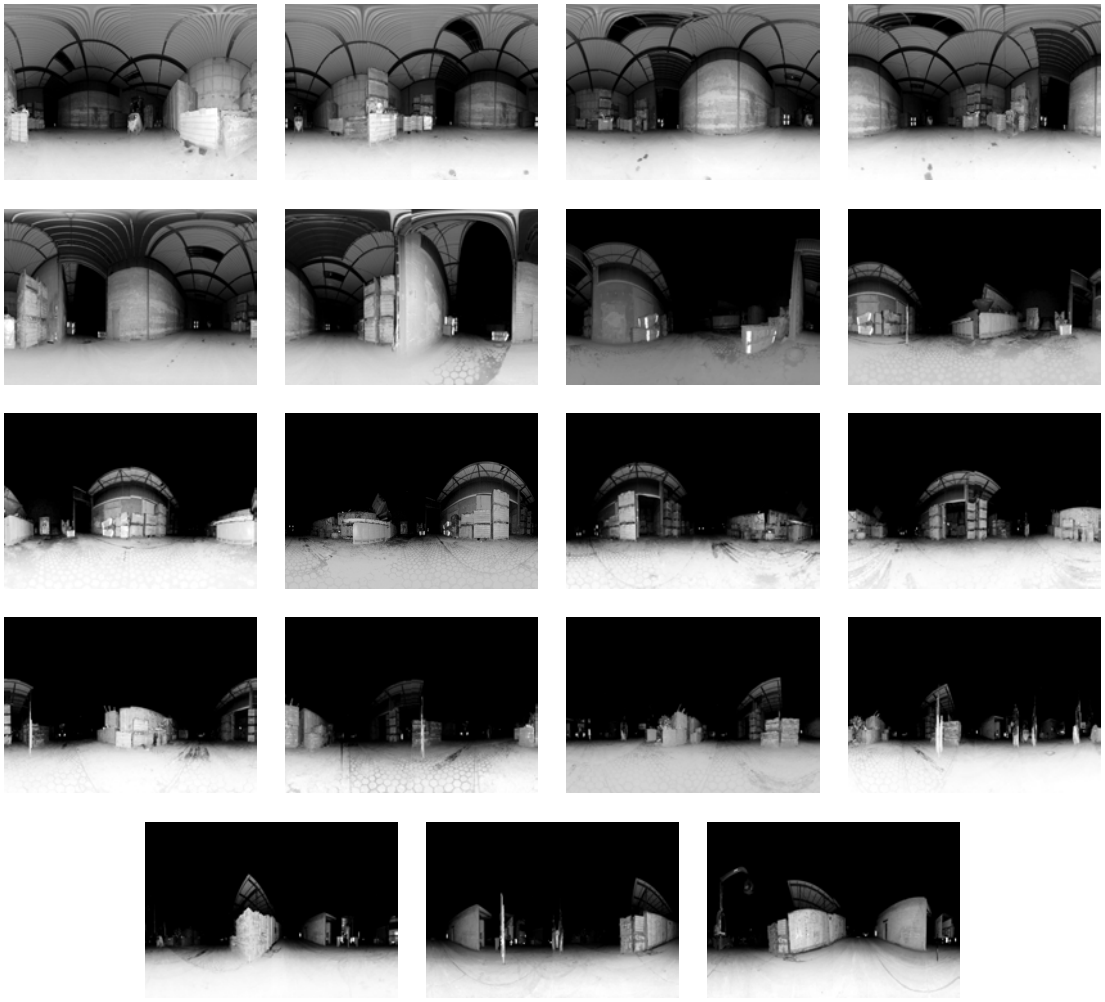
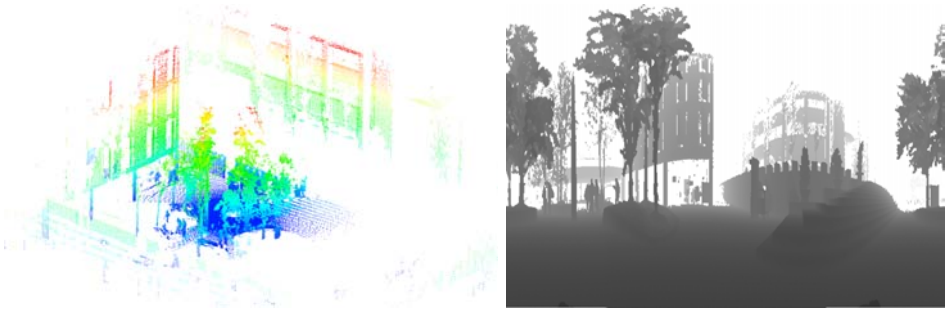


Figure 3.6: Gray scale intensity images of the scans in the scrap yard dataset.

3.2 The "Barcelona Robot Lab" dataset

The "Barcelona Robot Lab" dataset covers about 10000 m^2 of the UPC Nord Campus in Barcelona, which is intended for use in mobile robotics and computer vision research. It is available at `http://www.iri.upc.edu/research/webprojects/pau/datasets/BRL/php/dataset.php`. Multiple sensor channels have been provided in the dataset, including odometry information, compass data, unorganized point clouds, on-board robot imagery as well as imagery from a camera sensor network. The point clouds were acquired by an aLRF which is similar to that of TOMAR, i.e., each point cloud consists of a collection of 2D planar range scans. As a result, the points in each 3D point cloud can be reordered according to the order in which the points are measured by the sensor, and each resulting scan contains about $400,000$ points.

In order to apply algorithms which require organized point clouds, the points in each scan have been reordered using the above-mentioned method. One typical scan has been visualized in Figure 3.7 as point cloud and range image. Due to space limitations, we



(a) Scan 5, point cloud (b) Scan 5, range image

Figure 3.7: One typical scan of the "Barcelona Robot Lab" dataset. (a): as point cloud; (c) as range image in gray scale.

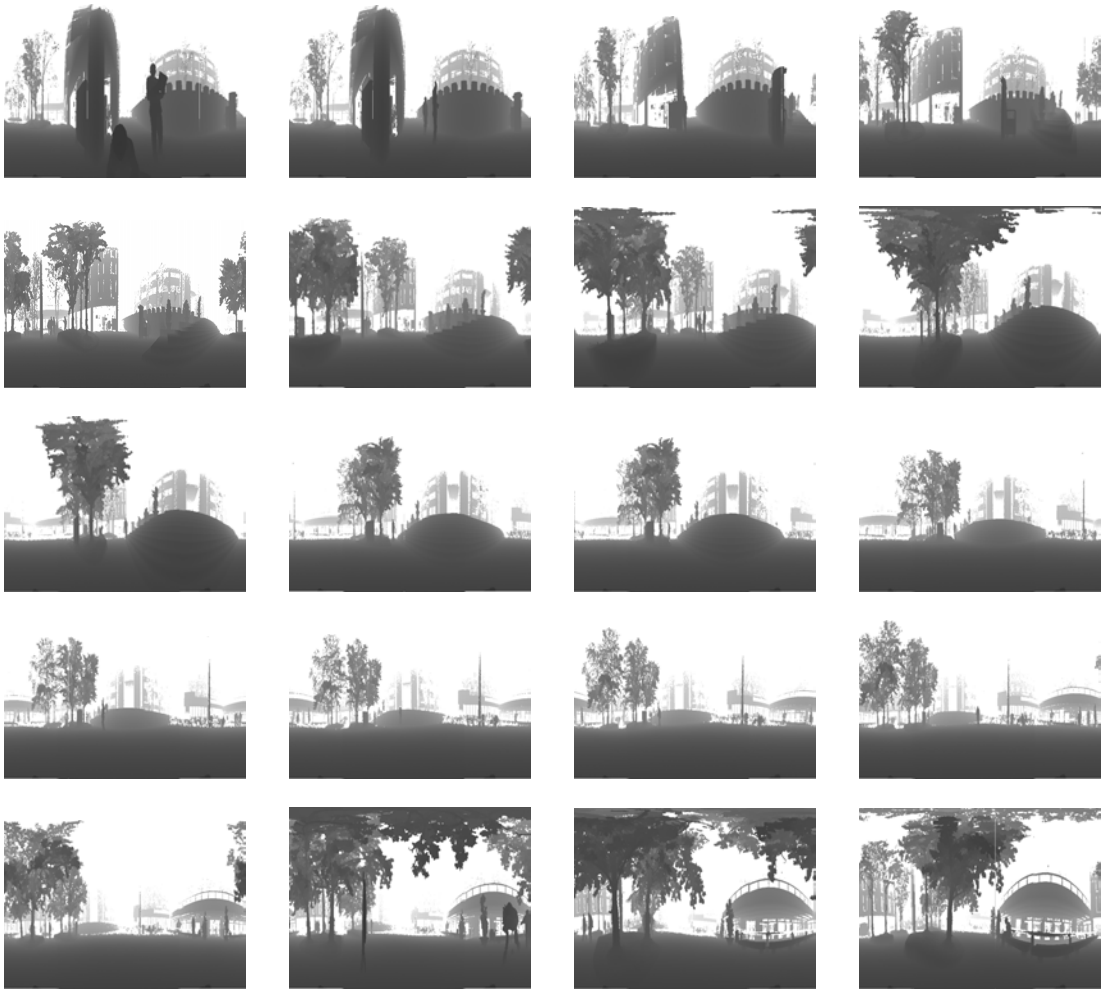


Figure 3.8: The gray scale range images of the first 20 scans in the "Barcelona Robot Lab" dataset.

Figure 3.9: Two photos of the "Collapsed Car Park" scenario, a front overview and a close-up view. The photos are adopted from Pathak et al. (2010a).

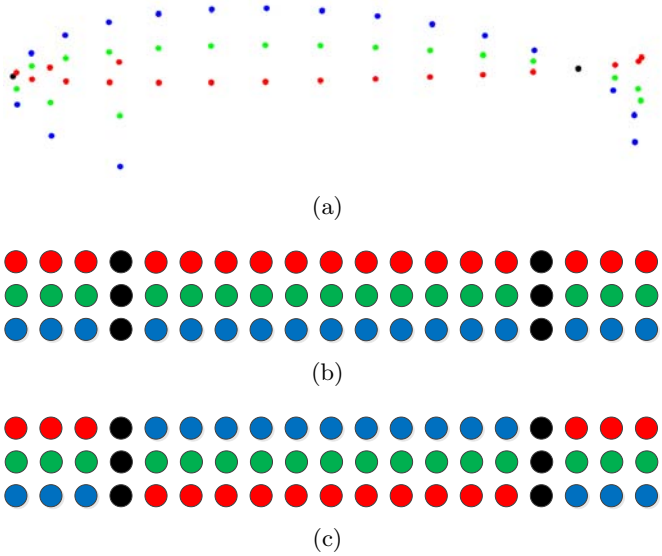


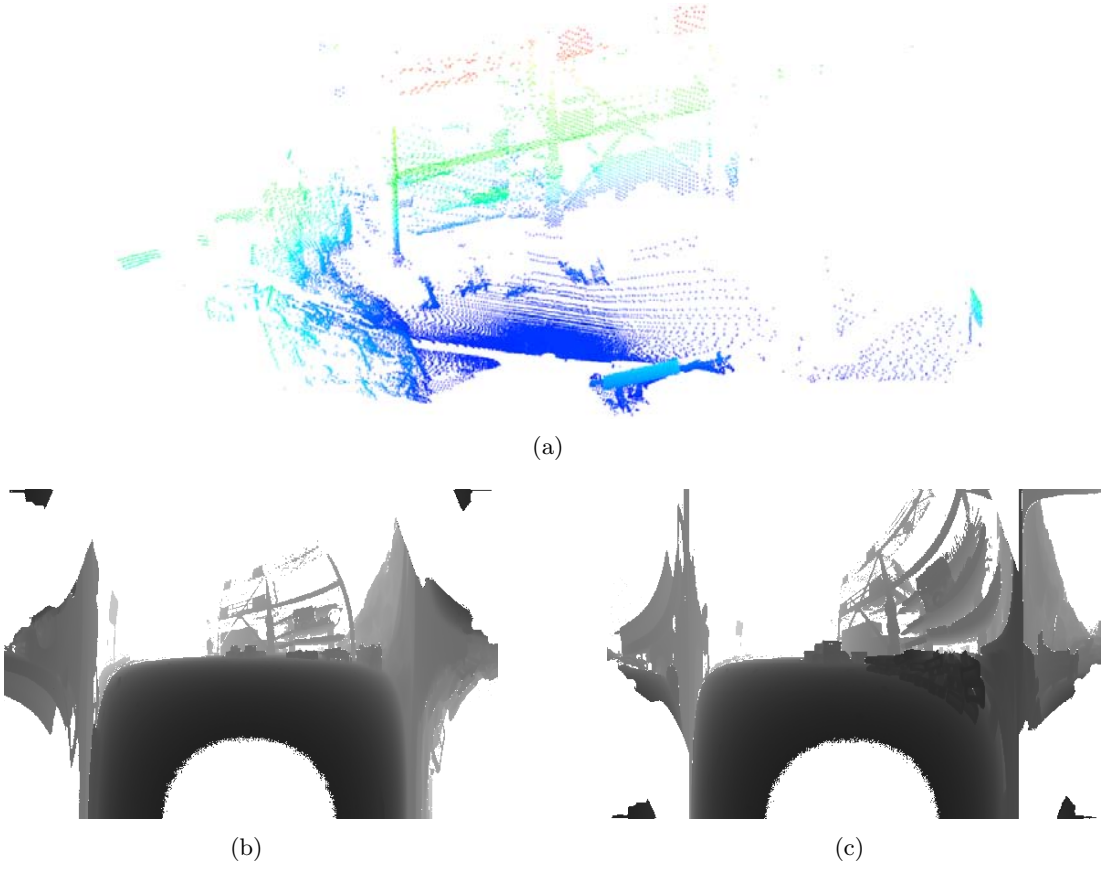
Figure 3.10: Illustration of the intersection in the range image between scan slices. (a): a scan consisting 3 scan slices which have been color coded; (b): range image by aligning each scan slice as a row, resulting the pixel-neighborhood information different from spatial; (c): reordered range image in order to make the pixel-neighborhood information to be consistent with spatial.

cannot present all range images here; instead, the range images of the first 20 scans are shown in Figure 3.8. For further range images, please refer to Appendix A.

3.3 The "Collapsed Car Park" dataset

This dataset was collected by Pathak et al. (2010a) during the 2008 NIST[1] Response Robot Evaluation Exercise at Disaster City, Texas. The scenario is depicted in Figure 3.9, and the raw data are available at `http://www.robotics.jacobs-university.de/datasets/RAW/RREE08/crashedCarPark/`. The data were gathered by a track robot

[1]National Institute of Standards and Technology



(a) (b) (c)

Figure 3.11: Scan 8 in the "Collapsed Car Park" dataset. (a): as point cloud which is colored by height; (b): as range image by aligning each scan slice as a row; (c): as range image after reordering the points. For the range images. Note the difference between (b) and (c).

equipped with an aLRF; the aLRF is based on a SICK S 300 which has a horizontal FoV of 270° of 541 beams. The sensor is pitched from -90° to +90° at a spacing of 0.5°, which leads to an organized point cloud of $541 \times 361 = 195,301$ points per sample. Compared to that of the other three datasets, this aLRF has a narrower FoV. Since the robot was operated under conditions with high amounts of rubble and dust, the odometry information is useless.

Range images, by aligning each scan slice as a row, have been provided in the dataset. However, when they are sampled, all scan slices in each point cloud intersect at two points on the rotation axis since the LRF's FoV is larger than 180°, see Figure 3.10(a). This results intersection when projecting spatial points to range image and vise versa, see Figure 3.10(b). In order to apply range image based algorithms to this dataset, we have reordered the range images, see Figure 3.10(c). A range image is shown in Figure 3.11 with both before and after being reordered, where the difference can be easily inspected. The range images of reordered point clouds are detailed in Figure 3.12.

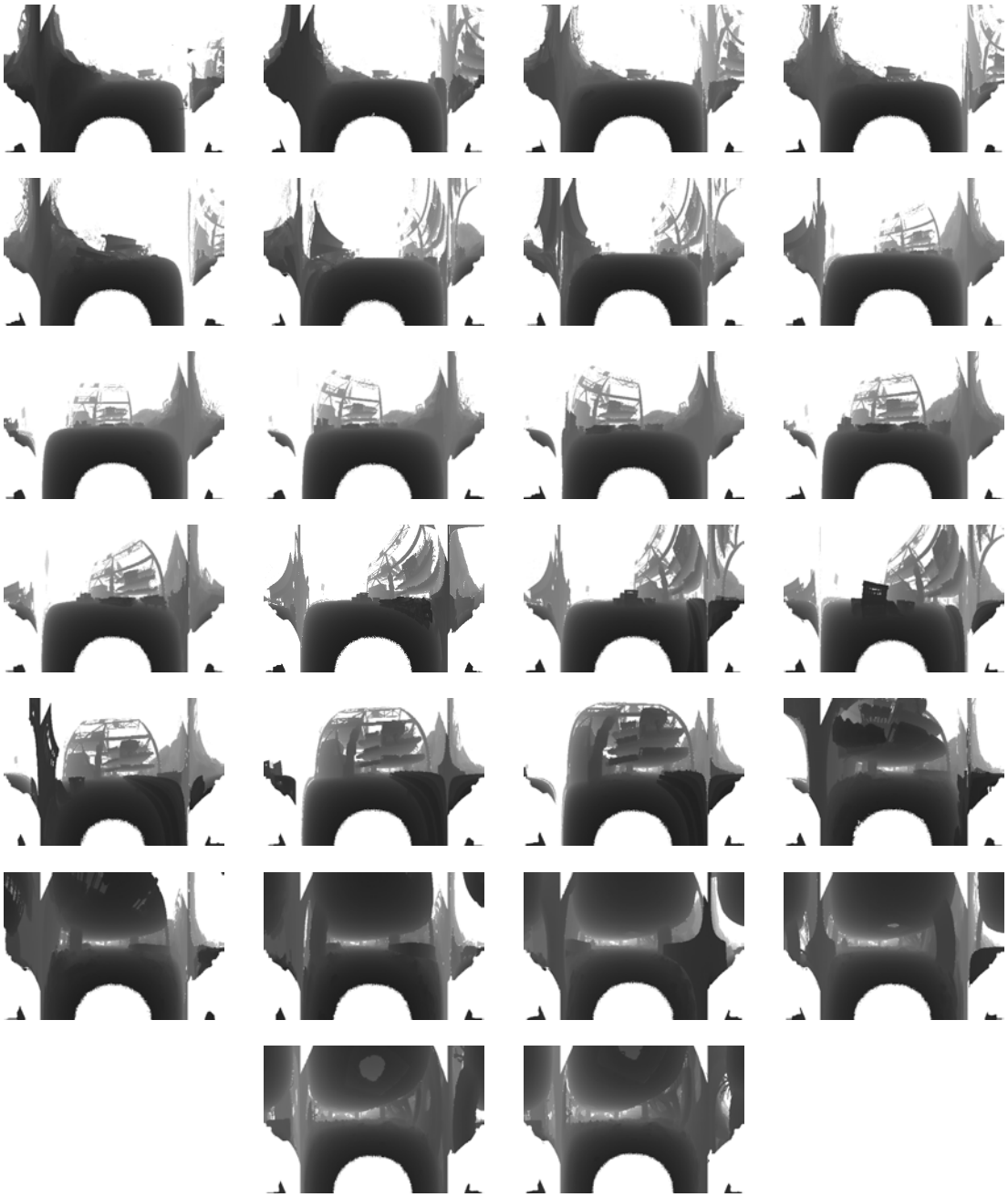


Figure 3.12: Gray scale range images of the "Collapsed Car Park" dataset.

Figure 3.13: A Bing map for the same area of the "Bremen City Center" scenario, which is from a bird's eye viewpoint.

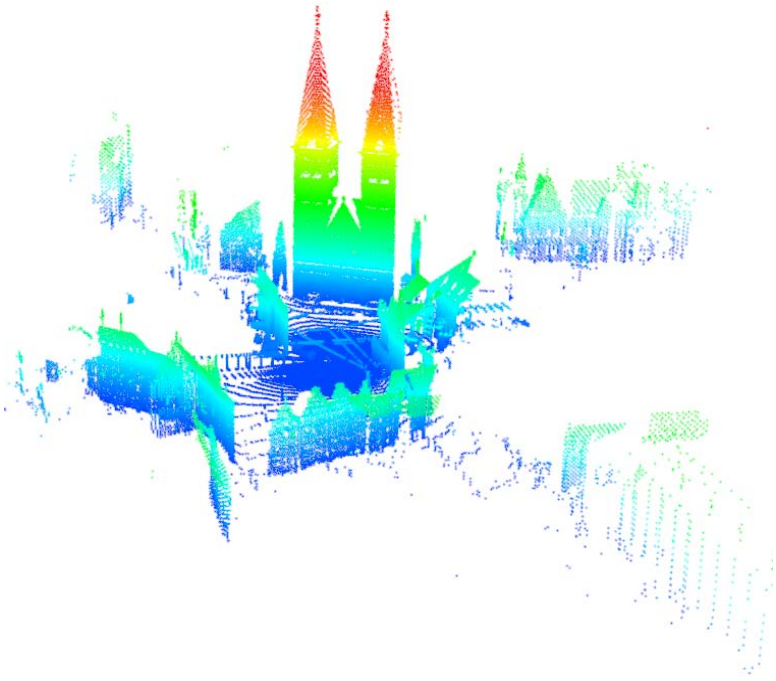


Figure 3.14: Scan 12 (as point cloud) in the "Bremen City Center" dataset, the points are colored by height.

3.4 The "Bremen City Center" dataset

This dataset consists of a big outdoor urban scenario and was gathered at the city-center of Bremen, Germany; a photo of the scenario is shown in Figure 3.13. It is publicly available at `http://kos.informatik.uni-osnabrueck.de/3Dscans/`. The RIGEL VZ-400 has been utilized for obtaining point clouds, which has a maximum

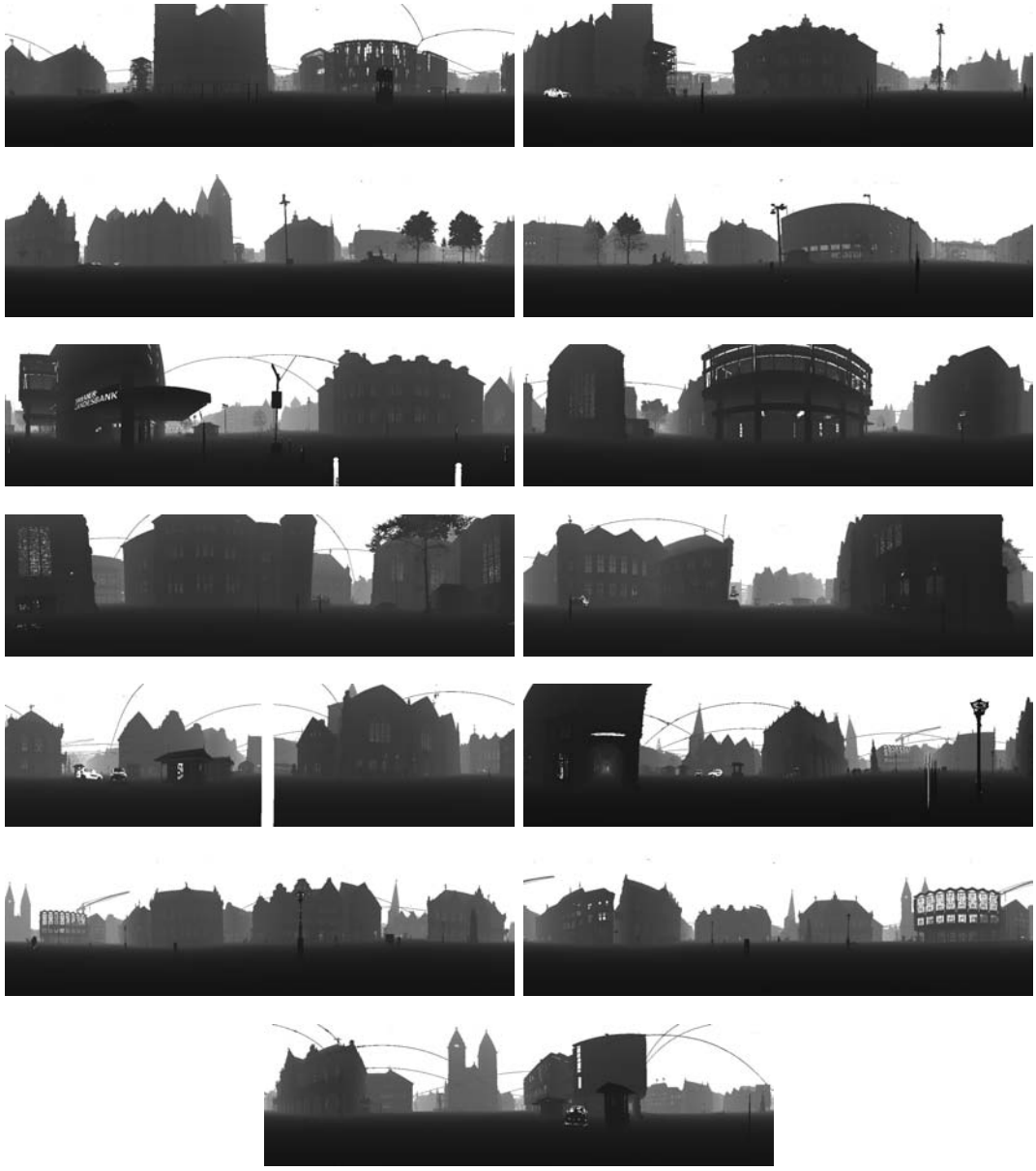


Figure 3.15: Gray scale range images of the 13 scans in the "Bremen City Center" dataset.

range of about half a kilometer, a horizontal FoV of 360°, and a vertical FoV of 100°. Sub-sampled point clouds have been used in this work, with an angular resolution of 0.25 deg. Therefore, there are $1441 \times 401 = 577,841$ points in each resulting scan. In contrast to aforementioned datasets, this dataset has a ground-truth which was obtained based on commercial reflective marker-based registration, see Pathak et al. (2010c) for detail.

This dataset has been employed to evaluate the robustness of registration and the predicted power of two overlap-metrics in Pathak et al. (2010c). It is challenging for registration due to a large distance between successive scan poses, see the ground-truth

in Section 6.4. As a result, this dataset features a high presence of occlusions and partial observations. A typical scan is shown in Figure 3.14, and the gray scale range images are illustrated in Figure 3.15.

3.5 Summary

Four datasets which will be employed in the following chapters have been introduced, where one is from our custom-built platform and the other three are publicly available. For the sake of making the presentation both simple and clear, we will denote the datasets using the city name where they are gathered as follows:

- "Scrap Yard" dataset – *Hamburg dataset*
- "Barcelona Robot Lab" dataset – *Barcelona dataset*
- "Collapsed Car Park" dataset – *Texas dataset*
- "Bremen City Center" dataset – *Bremen dataset*

Chapter 4

Point cloud plane segmentation

> Choose a job you love, and you will never have to work a day in your life.
>
> Confucius

THIS chapter focuses on the problem of point cloud plane segmentation, which has also been termed using keywords such as plane detection, plane extraction, and planar model segmentation in literatures. Being a complex task, it is still an open problem and an active research area in both the mobile robotics and the graphics community. This chapter is organized as follows: After reviewing related work in Section 4.1, the problem is formulated in Section 4.2. Then we present four complementary strategies for different point cloud formats and environments: First, a point based region growing approach is described in Section 4.3, which is proposed for *organized point clouds* from *structured/unstructured* environments. Second, a subwindow based region growing method is drawn in Section 4.4 for *organized point clouds* from *structured* environments. Third, Section 4.5 details a hybrid region growing algorithm, which is proposed for *organized point clouds* from *structured/unstructured* environments. Fourth, a cached octree region growing algorithm for *organized/unorganized point clouds* from *structured/unstructured* environments are introduced in Section 4.6. The experiments and results are given in Section 4.7, followed by a summary in Section 4.8.

4.1 Related work

Objects with planar surfaces are prevalent in urban and indoor environments, such as tables, floors, doors, walls, ceilings and roads. These can be segmented into polygons which provide a compressed representation against to the raw point clouds. The compression ratio is usually higher than 90% according to Vaskevicius et al. (2010) and Kaushik and Xiao (2012). The polygons can also be used for 3D model reconstruction (Vosselman et al., 2001; Bauer et al., 2003; Dorninger and Nothegger, 2007), scan registration (Pathak et al., 2010b) and SLAM (Weingarten and Siegwart, 2005; Viejo and Cazorla, 2007; Pathak et al., 2010a).

Table 4.1: Plane segmentation approach classification according to the temporal type and methodology.

Method/Type	Hough Transform	RANSAC	Region growing
Early generation	Borrmann et al. (2011); Dube and Zell (2011)	Weingarten and Siegwart (2005); Trevor et al. (2012)	Harati et al. (2007); Poppinga et al. (2008b); Hegde and Ye (2009); Xiao et al. (2011); Georgiev et al. (2011); Kaushik and Xiao (2012)
Late generation	Vosselman et al. (2001)	Bauer et al. (2003)	Dorninger and Nothegger (2007); Hänel et al. (2003)

Plane segmentation, which is still an open problem and an active research area in mobile robotics, is a fundamental issue for plane-based scan registration approaches. Although there are available algorithms in the graphics community (Bajaj et al., 1995; Amenta et al., 1998), they cannot be adopted into robotic systems directly due to relying on more precise depth information than what the robotic sensors can provide. Therefore, various algorithms to deal with noisy datasets on this topic have been proposed by researchers from the robotics community.

The classification of current plane segmentation algorithms has been depicted in Table 4.1. From a temporal point of view, plane detection approaches can be classified into two categories, namely early generation and late generation. Late generation happens after point clouds captured from different positions being registered to a common coordinate system, which aims to learn a smooth model of the environment. It is usually employed for polygon model construction which has a lower complexity compared to the raw data, see Vosselman et al. (2001); Hänel et al. (2003); Bauer et al. (2003); Dorninger and Nothegger (2007). The major goal of late generation is for model simplification and visualization, hence its speed is not critical. In contrast to the former, early generation happens before registration and is applied to each single point cloud, providing features for registration and plane-based SLAM systems, e.g., Weingarten and Siegwart (2005); Poppinga et al. (2008b); Hegde and Ye (2009); Kaushik and Xiao (2012). Early generation is usually performed on mobile robots where the computational resource is limited, so its computational complexity should be relatively low in order to be employed for localization and mapping.

From a methodological point of view, most plane detection approaches can be classified into three categories, i.e., algorithms based on Hough transform, RANSAC (Fischler and Bolles, 1981) and region growing. Hough transform is a classic feature extraction method in image processing for the detection of straight lines, circles or ellipses, etc. In order to use it for plane detection in 3D point cloud segmentation, Borrmann et al. (2011) have evaluated different variants of the Hough Transform. It was found that the main problem is the representation of the accumulator besides computational costs. To deal with this, they proposed the accumulator ball as an accumulator. The evaluation of different Hough methods recommends the Randomized Hough Transform for plane detection in 3D point clouds. However, it still has a problematic weakness; its

processing time increases with the number of planes but not the number of points in the point cloud. As it is reported in the paper, the segmentation time using the Randomized Hough Transform would be significantly larger than region growing when more than a number (15 in their experiments) of planes present in the data. Similarly, Dube and Zell (2011) have also proposed to use the Randomized Hough Transform for plane detection from depth images, wherein they concentrated on the Microsoft Kinect cameras and made use of the sensor noise model to find proper parameter metrics for the Randomized Hough Transform. The influence of local sampling has been evaluated and found to produce better segmentation results. However, their test environment is a clean corridor with only walls, windows, roof and ground which means about four or fewer planes presented in each frame. Furthermore, the highest detection rate of walls is 82.2% which is not satisfied for detailed map generation.

RANSAC is another route for estimating parameters of a model from a dataset which may have outliers. When being applied to 3D point cloud plane segmentation, dealing with multiple models in one dataset should be considered. In Weingarten and Siegwart (2005), the cloud is decomposed into equal sized 3D cubes, and then one model is fitted to each cube using RANSAC, thus avoiding the multiple model problems. Afterwards, small planar patches are merged using cube neighborhood information. However, the algorithm is still time consuming due to the iterative nature of RANSAC. Trevor et al. (2012) utilize the RANSAC algorithm for plane segmentation in another manner. In their work, they search the plane with the most inliers in the dataset each time, then all inliers for this plane are removed from the point cloud, and the RANSAC is performed again to find the next largest plane. This process terminates when no plane with a sufficient number of points could be found. However, the segmentation time has not been reported in these two papers.

The concept of *organized point cloud* (also known as range image or structured point cloud) should be mentioned before introducing the region growing approach. An organized point cloud resembles an organized image-like structure, where the data is split into rows and columns. Examples of such point clouds include data delivered by stereo cameras or ToF cameras. The advantage of such a point cloud is that the relationship between adjacent points (like pixels in an image) is known, making nearest-neighbor operations much more time-efficient. Some aLRFs can also produce organized point cloud datasets, one example built by us has been introduced in Section 3.1. The adjacency information will be denoted as *pixel-neighborhood information* later on in this thesis. Note that the nearest neighbor search (NNS) is an important issue for region growing, since it has to be performed at each step of the growth. One disadvantage of organized point clouds is that they need more storage space because they contain invalid points (their coordinates are usually stored as NaNs) in order to make the image-like structure.

octree is another data structure for NNS in 3D graphics. It is a hierarchical tree data structure where each internal node has exactly eight or no children, see Figure 4.1 for an explanation. It has a wide field of applications in point cloud processing such as sub-sampling, compression and visualization besides NNS. Compared to looking for nearest neighbors in organized point clouds by selecting adjacent pixels, neighbor search in octree is much slower. Because the former algorithm returns neighbors from

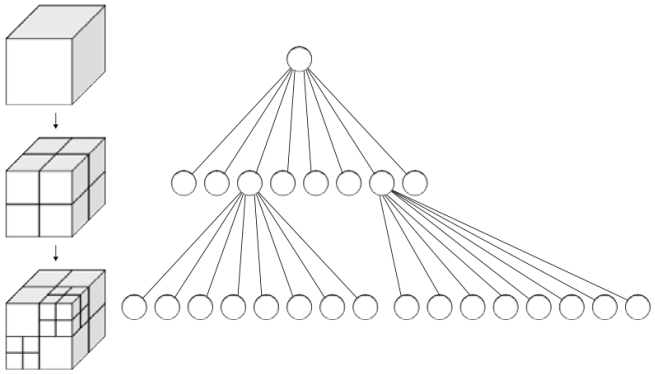


Figure 4.1: Left: Recursive subdivision of a cube into octants. Right: The corresponding octree. The figure is adopted from `http://en.wikipedia.org/wiki/Octree`.

the closest rays, forming the projection from the object to the sensor, while the latter returns true geometric neighbors with regard to Euclidean distance.

Region growing was proposed for image segmentation based on color information between neighbor pixels. It was extended to plane segmentation in Hänel et al. (2003). In their work, planar segments are used to smooth the resulting map, but not as features in the SLAM phase, which means the segmentation speed is not a critical point. To embed the approach into plane-based mapping systems, Poppinga et al. (2008b) presented two optimizations to it, by which the speed was accelerated significantly: one optimization is to use pixel-neighborhood information for NNS; the other one is an incremental version for plane parameters update and plane fitting error computation.

In the above algorithms, only one point is added to the region in each growing step. As alternatives, the growth unit can also be a line segment or a subwindow. In Harati et al. (2007), the so-called bearing angle is computed for each point as a measure of the flatness of its local area, using pixel-neighborhood information. Based on this measure, a line-based region growing algorithm is proposed. However, since bearing angle is the incident angle between the laser beam and edges of the scanned polygon in the selected direction, it can not be properly calculated in cluttered environments. Georgiev et al. (2011) started by extracting 2D line segments from each 2D scan slice (each row or column in organized point cloud), where connected line segments represent candidate sets of coplanar segments. Then, a region growing algorithm is utilized to find planar segments and their least squares fitted (infinite) plane. In Kaushik et al. (2010) the point cloud is divided into small subwindows, and plane parameters are computed for each subwindow. Then the subwindows are clustered into big surfaces by a Breadth-First search algorithm. One drawback of this approach is that there are some subwindows whose appearance could not be approximated by a plane. Then it was extended and published in Kaushik and Xiao (2012), where the planar patches and non-planar patches are distinguished. However, the plane parameters are not recomputed when new data points are added; instead, the plane parameters of the selected seed patch are treated as the plane model. Therefore the resulting plane parameter, a fundamental issue for plane-based SLAM, would be inaccurate.

4.2 Problem formulation

The plane segmentation problem is stated as follows: given an organized or unorganized point cloud Ψ which contains 3D points $\mathbf{p}_i, i = 1, 2, ...N$, the goal is to find connected coplanar points and their optimally fitted infinite planes. Invalid points may exist if the cloud is organized; their coordinates are typically fixed to infinity and are represented as NaNs. Different forms of plane equations exist in the literature. A comparison of them can be found in Weingarten (2006). The Hessian form equation is chosen to represent planes in this dissertation. It is described as

$$\hat{\mathbf{n}} \cdot \mathbf{p} = d, \tag{4.1}$$

where $\hat{\mathbf{n}}$ is the unit normal vector of the plane, $\mathbf{p}$ is an arbitrary point on the plane and d is the distance from the origin to the plane. When a set of points are found to be coplanar, least squares is employed to find the optimal plane in the sense of minimizing the sum of squared residuals, i.e.,

$$\langle \hat{\mathbf{n}}, d \rangle = \arg\min_{\hat{\mathbf{n}},d} \sum_{j=0}^{M} (\hat{\mathbf{n}} \cdot \mathbf{p} - d)^2. \tag{4.2}$$

With the assumption that the geometric center $\mathbf{m}$ is on the plane, i.e.,

$$\hat{\mathbf{n}} \cdot \mathbf{m} = d, \tag{4.3}$$

Eq. (4.2) can be rewritten as

$$\hat{\mathbf{n}} = \arg\min_{\hat{\mathbf{n}}} \hat{\mathbf{n}}^{\mathrm{T}} \mathbf{S} \hat{\mathbf{n}}, \tag{4.4}$$

where

$$\mathbf{S} = \sum_{j=1}^{M} (\mathbf{p}_j - \mathbf{m})(\mathbf{p}_j - \mathbf{m})^{\mathrm{T}} \tag{4.5}$$

is the scatter matrix. $\mathbf{S}$ is a positive defined matrix which has three positive eigenvalues. In this case, Eq. (4.4) can be solved by principal component analysis, which gives the solution

$$\begin{cases} \hat{\mathbf{n}} = \hat{\mathbf{e}}_{min}(\mathbf{S}) \\ d = \hat{\mathbf{n}} \cdot \mathbf{m} \end{cases}, \tag{4.6}$$

where $\hat{\mathbf{e}}_{min}$ stands for the eigenvector which corresponding to the minimum eigenvalue of $\mathbf{S}$. It could be observed that $\mathcal{P}(\hat{\mathbf{n}}, d) \equiv \mathcal{P}(-\hat{\mathbf{n}}, -d)$, for the sake of sign consistency, only the solution with $d > 0$ will be used. Different algorithms to find coplanar points have been reviewed in last section, and we will present four strategies in the following sections.

4.3 Point based region growing

As introduced in Section 4.1, region growing algorithm for plane segmentation was originally proposed in Hänel et al. (2003), which has been depicted in Algorithm 4.1.

Algorithm 4.1: The plane extraction algorithm proposed in Hänel et al. (2003).

```
1 select point tuple v_1, v_2 ;
2 Π := {v_1, v_2} ;
3 while new point can be found do
4 |   select point v' with pointDist(Π, v') < δ;
5 |   if error(Π ∪ {v'} < ε && ‖(Π ∪ {v'}, v')‖ < γ) then
6 |   |   Π := Π ∪ {v'};
7 |   end
8 end
```

One major problem with the approach is the high time complexity. As reported in the paper, a naïve implementation needs 10 hours on a standard PC for a typical dataset with 200,000 surfaces (triangles). Then Poppinga et al. (2008b) revised it in order to embed it in a mapping system, where the plane parameters and plane fitting Mean Square Error (MSE) is calculated incrementally and the pixel-neighbor information in the organized point cloud is employed for efficient neighbor searching. It is accelerated without losing any precision in this work by the following two improvements:

1. A seed selection procedure is introduced to avoid blind region growth, only qualified seeds will be extended.
2. An efficient way for computing MSE is presented, which makes use of the relation between the scatter matrix and MSE.

4.3.1 Point based region growing algorithm

Our improved algorithm proceeds as follows: one point $\mathbf{p}_s$ is selected from the point cloud data Ψ, and its qualification as a new seed is examined (Algorithm 4.2, line 3 – 4). The point $\mathbf{p}_s$ will be regarded as a new seed if it meets the following two criteria:

1. There are six or more unidentified points within distance δ among its nearest eight neighbors in the image-like structure.
2. The local appearance of $\mathbf{p}_s$ is planar.

Except the threshold δ, the procedure is the same as the shape classification step in the subwindow based region growing algorithm (Section 4.4). If $\mathbf{p}_s$ is a new seed, it together with its nearest neighbors within distance δ is marked as a new growing region $\mathbf{G}$, and the nearest neighbors of $\mathbf{G}$ are put into a First-In-Last-Out (FILO) queue $\mathbf{Q}$ (Algorithm 4.2, line 5). Then $\mathbf{G}$ is extended by considering its neighbors within distance δ. Suppose the considering point is $\mathbf{p}_c$, it will be added to $\mathbf{G}$ iff (Algorithm 4.2, line 8)

1. The distance from $\mathbf{p}_c$ to the optimal plane of $\mathbf{G} \cup \mathbf{p}_c$ is less than γ.
2. The plane fitting error MSE of $\mathbf{G} \cup \mathbf{p}_c$ is less than ϵ.

The growth will continue until no more points can be added to $\mathbf{G}$ (Algorithm 4.2, line 6 – 12). Afterwards, if $\mathbf{G}$ contains more than θ points, it will be assigned to be a new planar segment and added to the regions set $\mathcal{R}$. Otherwise, it is added to the uncertain area $\mathcal{R}'$ (Algorithm 4.2, line 14 – 18). The segmentation ends when each

Algorithm 4.2: Point based Region Growing Plane Segmentation

Input: Ψ: an organized point cloud
Output: $\mathcal{R}$: planar segments, $\mathcal{R}'$: uncertain points

1 $\mathcal{R} \leftarrow \emptyset$, $\mathcal{R}' \leftarrow \emptyset, \mathbf{G} \leftarrow \emptyset, \mathbf{Q} \leftarrow \emptyset$;
2 **while** $(\Psi \setminus (\mathcal{R} \cup \mathcal{R}') \neq \emptyset)$ **do**
3 select $\mathbf{p}_s$ which has the minimum MSE in $\Psi \setminus (\mathcal{R} \cup \mathcal{R}')$;
4 **if** `newSeed(`$\mathbf{p}_s$`)` $==$ `true` **then**
5 $(\mathbf{G}, \mathbf{Q})$ = `initializeSeed` $(\mathbf{p}_s)$;
6 **while** $\mathbf{Q} \neq \emptyset$ **do**
7 $\mathbf{p}_c = \mathbf{Q}$.`pop()` ;
8 **if** `mse(`$\mathbf{G} \cup \mathbf{p}_c$`)`$< \epsilon$ && `dis(plane(`$\mathbf{G} \cup \mathbf{p}_c$`),`$\mathbf{p}_c$`)`$< \gamma$ **then**
9 $\mathbf{G} \leftarrow \mathbf{G} \cup \mathbf{p}_c$;
10 $\mathbf{Q} \leftarrow \mathbf{Q} \cup$ `NN(`$\mathbf{p}_c$, δ`)` ;
11 **end**
12 **end**
13 **end**
14 **if** `size(`$\mathbf{G}$`)`$\geq \theta$ **then**
15 $\mathcal{R} \leftarrow \mathcal{R} \cup \mathbf{G}$;
16 **else**
17 $\mathcal{R}' \leftarrow \mathcal{R}' \cup \mathbf{G}$;
18 **end**
19 **end**

point is identified to either $\mathcal{R}$ or $\mathcal{R}'$. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are pre-defined thresholds which need to be tuned.

4.3.2 Efficient MSE calculation

To compute plane parameter and fitting MSE efficiently is essential in Algorithm 4.2, since they are performed whenever a new point is investigated, no matter whether the point will be added (Algorithm 4.2, line 8). Incremental computation is intuitively a good solution for such a problem, accordingly, Poppinga et al. (2008b) proposed a method which could compute the plane parameters and MSE incrementally. However, the relationship between MSE and the scatter matrix can be employed to simplify the computation. The mathematical derivation of the relationship is drawn below.

Suppose there are K points in the segment, with coordinates $\mathbf{p}_i = (x_i, y_i, z_i)^{\mathrm{T}}, i = 1, ...K$. After least-squares plane fitting, MSE can be computed as:

$$MSE = \frac{1}{K} \sum_{i=1}^{K} (\hat{\mathbf{n}} \cdot \mathbf{p}_i - d)^2 \tag{4.7}$$

Substituting Eq. (4.3) into (4.7), it can be rewritten as

$$MSE = \frac{1}{K} \sum_{i=1}^{K} (\hat{\mathbf{n}} \cdot \mathbf{p}_i - \hat{\mathbf{n}} \cdot \mathbf{m})^2 \tag{4.8}$$

The expansion of Eq. (4.8) is

$$MSE = \frac{1}{K}\sum_{i=1}^{K} \hat{\mathbf{n}}^{\mathrm{T}}(\mathbf{p}_i - \mathbf{m})(\mathbf{p}_i - \mathbf{m})^{\mathrm{T}}\hat{\mathbf{n}} \tag{4.9}$$

Then Eq. (4.9) is rewritten as

$$MSE = (\hat{\mathbf{n}}^{\mathrm{T}}\mathbf{S}\hat{\mathbf{n}})/K \tag{4.10}$$

after some linear algebra. Combining Eq. (4.6) and (4.10), we get

$$MSE = \frac{1}{K}\lambda_{min}(\mathbf{S}) \tag{4.11}$$

4.4 subwindow based region growing

From the application point of view, the advantage of the point-based region growing algorithm is that it can be used in both structured and unstructured environments. Planar detection in structured environments could be faster if utilizing a larger growth unit such as subwindows in the image-like structure. Compared to Kaushik et al. (2010); Kaushik and Xiao (2012), our algorithm computes the plane parameters for the growing region whenever a new subwindow is added. This is advantageous for plane-based registration and SLAM which cannot be performed without plane parameters.

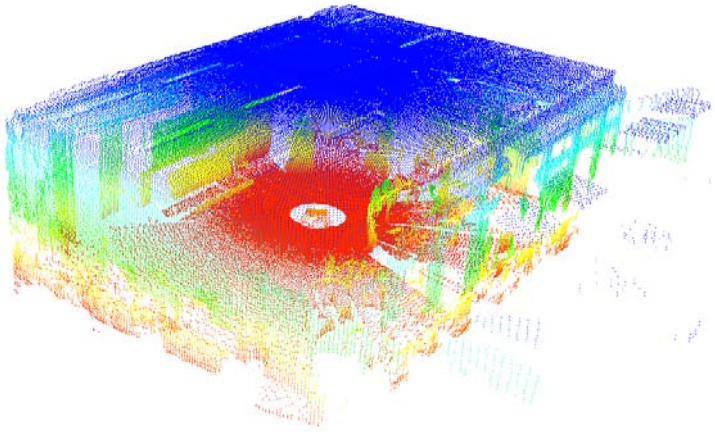
4.4.1 Feasibility of subwindow based region growing

The subwindow is suitable for plane segmentation if the following two assumptions are fulfilled: First, most of the subwindows which locate on planar surfaces have a planar appearance. Second, the subwindows from the same physical surface have similar plane parameters. To confirm the first assumption, the point cloud is decomposed into small subwindows first, then the subwindows are classified into two categories based on their shape appearances, namely planar or non-planar.

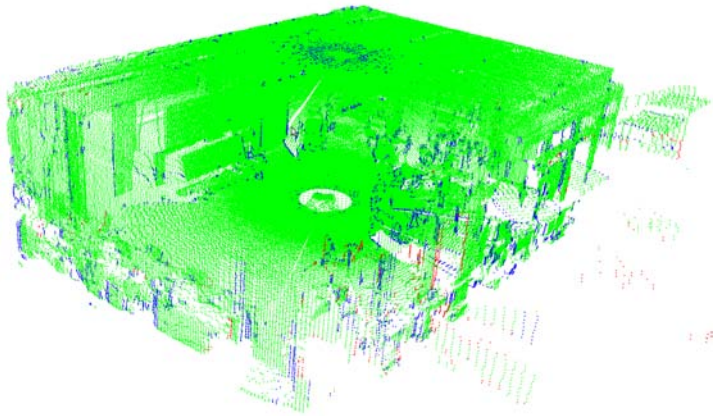
To determine the appearance of a subwindow ω, the scatter matrix $\mathbf{S}$ can be computed as in Eq. (4.5) where $\mathbf{p}_0, \mathbf{p}_1, ...\mathbf{p}_{sw}$ are the valid point in ω. Note that invalid points may also appear in ω, for example when there is no object in certain laser beam directions. Given sorted eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$ of $\mathbf{S}$, the shape of ω is determined by the below criteria:

$$\omega \in \begin{cases} \text{sparse}, & \text{if sw} < \mu\text{size}(\omega) \\ \text{planar}, & \text{if } \lambda_1 \leq \eta\lambda_2 \\ \text{non-planar}, & \text{otherwise} \end{cases}, \tag{4.12}$$

where $\mu, \eta \in (0, 1)$, and size(ω) is the total number of valid and invalid points. The subwindow is marked as sparse when the number of valid points is smaller than a given threshold. A parameter tuning step is needed for η in order to yield satisfactory classification results. From the experiments, it is only related to the employed range sensor, i.e., the value of η is fixed for each sensor.



(a)



(b)

Figure 4.2: (a):A typical point cloud gathered by our custom-built aLRF. The data was sampled in our robot laboratory and the points are colored by height. (b): The classification result of subwindows, where the subwindow side length was set to 3; sparse, planar and non-planar subwindows are colored in red, green and blue.

In our work, the subwindow is set to be square, with side length bigger than 2, as 4 points are not adequate for shape analysis. For better understanding, an example is illustrated in Figure 4.2. The side length of the subwindow and η have been set to 3 and 0.3 respectively. For visualization, sparse, planar and non-planar subwindows are colored in red, green and blue. It is clear that most subwindows which are scanned from planar surfaces have planar appearance. Similar results were found for other scans, thus the first assumption has been confirmed.

Now we deal with the second assumption, it means that the normal of a subwindow is a good estimation of the surface. Hänel et al. (2003) pointed out that local surface normals from a planar surface in the real world are almost uniformly distributed. However, two orthogonal 2D laser scanners on a mobile robot (see Figure 2.3(b)) were used for data collection in their robotic system (the system has been introduced in Section 2.1). Clearly, both localization error and sensor noise exist in their 3D point clouds.

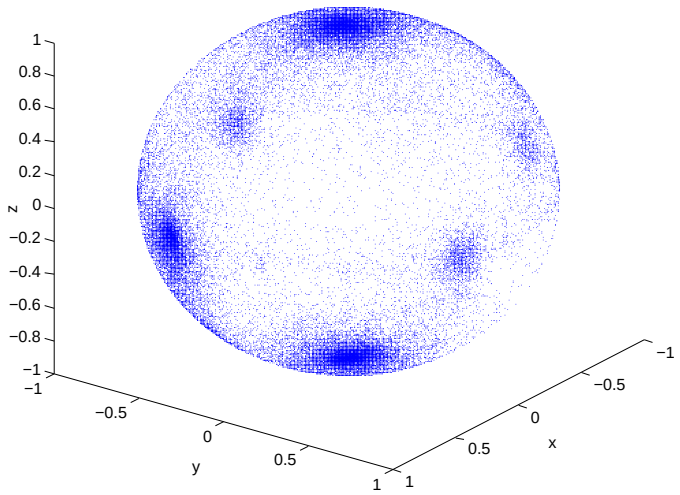


Figure 4.3: Distribution of unit normals for all planar subwindows in Figure 4.2, with a different perspective to the point cloud. Although normals are present almost everywhere on the sphere, it is still easy to find six dense clusters corresponding to the six big planar surfaces in the point cloud.

The noise level should be higher than that of a point cloud sampled by an aLRF, which only contains sensor noise. Considering the planar subwindows of Figure 4.2, their unit normals have been visualized in Figure 4.3. Although some random normals are still present, several dense clusters are apparent according to the planar surfaces in Figure 4.2, thus the second assumption has been verified. Therefore, it can be concluded that subwindows can be used in plane segmentation for certain sensor configuration.

4.4.2 subwindow based region growing algorithm

The proposed algorithm proceeds as follows. For an organized point cloud Ψ, it is divided into subwindows first based on the image-like structure. The subwindows are then classified into planar and non-planar and only planar subwindows will be kept for plane segmentation (Algorithm 4.3, line 2). At the same time, local plane parameters for each planar subwindow are computed as well as plane fitting MSE. Afterwards, the subwindow ω with the minimum MSE among all unidentified subwindows is chosen as a new seed (Algorithm 4.3, line 4). A growing region $\mathbf{G}$ is initialized by ω and its unidentified neighbors are put into a FILO queue $\mathbf{Q}$ which keeps $\mathbf{G}$'s neighbors (Algorithm 4.3, line 5). Then, $\mathbf{G}$ is extended by investigating its neighbors in $\mathbf{Q}$. Suppose that ω_c is the neighbor subwindow being considered, it is assigned to $\mathbf{G}$ iff it meets the following criteria (Algorithm 1, Line 8 – 11).

1. The dot product between the normal vectors of ω_c and $\mathbf{G}$ is greater than δ. Actually, $\arccos(\hat{\mathbf{n}}_G \cdot \hat{\mathbf{n}}_{\omega_c})$ is the angle between G and ω_c, so this criterion is to ensure that the investigated subwindow has a similar surface normal direction to $\mathbf{G}$.

Algorithm 4.3: subwindow based region growing plane segmentation

Input: Ψ: an organized point cloud
Output: $\mathcal{R}$: planar segments, $\mathcal{R}'$: uncertain points

```
1  R ← ∅, R' ← ∅, G ← ∅, Q ← ∅, Ω ← ∅ ;
2  Ω=planarSubwindows(Ψ) ;
3  while Ω \ (R) ∪ R') ≠ ∅ do
4  |   select ω with the minimum MSE in Ω \ (R ∪ R') ;
5  |   G ← ω, Q ←NN(ω) ;
6  |   while Q ≠ ∅ do
7  |   |   ω_c = Q.pop() ;
8  |   |   if n̂_G · n̂_ω_c > δ && |n̂_G · (m_G − m_ω_c)| < γ && mse(G ∪ ω_c) < ε then
9  |   |   |   G ← G ∪ ω_c ;
10 |   |   |   Q ← Q ∪ NN(ω_c) ;
11 |   |   end
12 |   end
13 |   if size(G)≥ θ then
14 |   |   R ← R ∪ G ;
15 |   else
16 |   |   R' ← R' ∪ G ;
17 |   end
18 end
```

2. To avoid adding a subwindow which is parallel but not coplanar to G, the distance from the mass center of ω_c to the optimal plane of G should be less than γ.
3. To guarantee acceptable flatness of the resulting segment, the plane fitting MSE of $G \cup \omega_c$ should be less than ϵ.

This process ends when no more neighbors can be added to G, i.e., when Q is empty (Algorithm 4.3, Line 6 – 12). Since our goal is to extract large planes in the scene, only G with more than θ subwindows is regarded as a planar segment and added to the plane set $\mathcal{R}$, otherwise it is added to the uncertain region set $\mathcal{R}'$ (Algorithm 4.3, Line 13 – 17). The algorithm terminates when every planar subwindow is assigned either to $\mathcal{R}$ or $\mathcal{R}'$. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are pre-defined thresholds which need to be tuned.

4.4.3 Incremental plane parameter and MSE calculation

As seen from the algorithm, a plane should be fitted to the growing region whenever a new subwindow is added, the time cost of it is critical for a plane-based registration. In order to bound its cost, the consequent incremental method is proposed. When shape classification is finished, the following quantity is tracked for each planar subwindow besides plane parameter, MSE, $\mathbf{S}$ and $\mathbf{m}$:

$$\mathbf{J} = \sum_{i=1}^{K} \mathbf{p}_i \mathbf{p}_i^{\mathrm{T}}, \tag{4.13}$$

which is the second order moment about the origin. All above quantities are also tracked for the growing region, they are used to derive the new plane parameters together with that of the subwindow to be added. At the beginning of the region growing, i.e., when a new seed is selected, the quantities of the growing region is simply assigned to that of the seed. Then the quantities of the growing region is updated incrementally as follows. Suppose there are K_G points in the current growing region; the subwindow ω has passed the coplanar tests and is going to be added, it contains K_ω points. In Eq. (4.14), $\mathbf{G}$ and ω are used as subscripts to denote the growing region and the investigating subwindow respectively, while there is no subscript for the combined region. Note that $\mathbf{G}$ may contain just one subwindow, i.e., when only the seed subwindow has been added. Eq. (4.14) is developed to calculate the plane parameters of the combined region using the above tracked quantities. Obviously, it can also be used for computing the planar parameters when merging two coplanar segments.

$$\begin{cases} \mathbf{M} &= \mathbf{m}_G K_G + \mathbf{m}_\omega K_\omega \\ \mathbf{m} &= \mathbf{M}/(K_{GR} + K_\omega) \\ \mathbf{J} &= \mathbf{J}_G + \mathbf{J}_\omega \\ \mathbf{S} &= \frac{1}{K_G+K_\omega} \sum_{i=1}^{K_G+K_\omega} (\mathbf{p}_i - \mathbf{m})(\mathbf{p}_i - \mathbf{m})^{\mathrm{T}} \\ \hat{\mathbf{n}} &= \hat{\mathbf{e}}_{min}(\mathbf{S}) \\ d &= \hat{\mathbf{n}} \cdot \mathbf{m} \\ \mathrm{MSE} &= \frac{\lambda_{min}(\mathbf{S})}{K_G+K_\omega} \end{cases} \tag{4.14}$$

In Eq. (4.14), all the other equations need constant time except the computation of $\mathbf{S}$. The time complexity to find the eigenvalues of a $n \times n$ matrix is $O(n^3)$, so the eigenvalue decomposition of $\mathbf{S}$ is the most time consuming part when K_G is small at the starting stage of a growing region. Then K_G increases as the region grows, and the calculation of $\mathbf{S}$ becomes the most time consuming part when there are many points ($K_G > n^3, n = 3$) in the growing region. In order to make the algorithm fast, we calculate $\mathbf{S}$ using other tracked quantities. After some algebra, it can be simplified as

$$\mathbf{S} = \mathbf{J} - \mathbf{M}\mathbf{m}^{\mathrm{T}} \tag{4.15}$$

Eq. (4.14) and (4.15) yield an incremental version for computing the plane parameters in the algorithm.

4.4.4 Computational complexity analysis

Suppose an organized point cloud from structured environment contains n points, which is to be segmented using the algorithm. The subwindow size is set to k, which means there are $m = \lfloor n/k \rfloor$ subwindows. Shape classification needs constant time for each subwindow with a time complexity $O(m)$. In addition, the neighbor search executes with a time complexity at most $O(m \log m)$, for all subwindows belonging to one segment. Computing the plane parameter when a new subwindow is added to the region

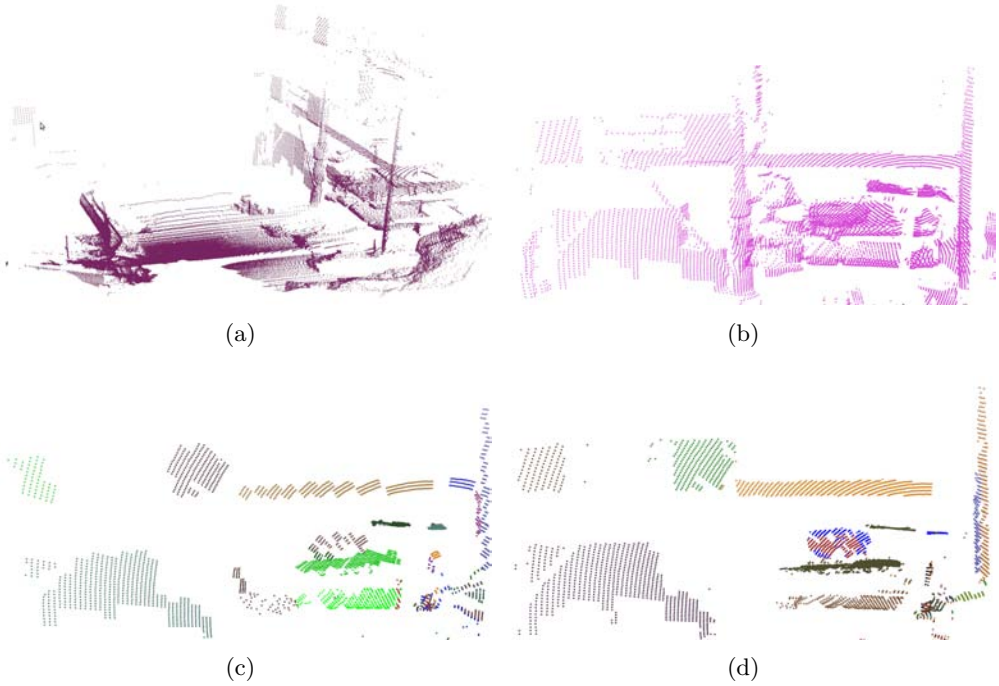


Figure 4.4: A segmentation quality comparison between the subwindow based and hybrid region growing when applied to scan from unstructured environments. (a): One scan from the Texas dataset. (b): A close-up view of one part in (a) which has abundant planar surfaces. (c) and (d): Close-up views of the plane segmentation result from subwindow based and hybrid region growing, respectively.

also needs constant time with time complexity of $O(m)$. To sum up, the overall time complexity of the algorithm is $O(m \log m)$. At most seven variables are tracked for each subwindow and segment, which yields the memory complexity $O(m)$.

4.5 Hybrid region growing

The subwindow based region growing algorithm has good performance in structured environments with higher speed compared to the point based region growing algorithm. However, its result is unsatisfactory when applied to unstructured environments, see Figure 4.4 as an example. The segmentation result from pure subwindow based region growing is not satisfied since the plane edges are classified into non-planar or sparse. This also happens when there are trees in front of a wall, as part of the wall will be occluded by the leaves, making the corresponding subwindow non-planar. The hybrid region growing algorithm is proposed to cope with this situation by considering both the planar subwindows and non-planar subwindows in the region growing process.

4.5.1 Hybrid region growing algorithm

The procedure used in the hybrid region growing algorithm is quite similar to that of the subwindow based region growing algorithm. For a point cloud Ψ, it is decomposed

Algorithm 4.4: Hybrid region growing plane segmentation

Input: Ψ: an organized point cloud
Output: $\mathcal{R}$: planar segments, $\mathcal{R}'$: uncertain points

1 $\mathcal{R} \leftarrow \emptyset$, $\mathcal{R}' \leftarrow \emptyset, \mathbf{G} \leftarrow \emptyset, \mathbf{Q} \leftarrow \emptyset, \Omega \leftarrow \emptyset, \Theta \leftarrow \emptyset, \mathbf{Q} \leftarrow \emptyset$;
2 (Ω, Θ)=`subwindows(`Ψ`)` ;
3 **while** $\Omega \setminus (\mathbf{R}) \cup \mathbf{R}') \neq \emptyset$ **do**
4 select ω with minimum MSE in $\Omega \setminus (\mathbf{R} \cup \mathbf{R}')$;
5 $\mathbf{G} \leftarrow \omega$, $\mathbf{Q} \leftarrow$`NN(`ω`)` ;
6 **while** $\mathbf{Q} \neq \emptyset$ **do**
7 $\omega = \mathbf{Q}$.`pop()` ;
8 **if** `isPlanar(`ω`)==true` **then**
9 **if** $\hat{\mathbf{n}}_G \cdot \hat{\mathbf{n}}_{\omega_c} > \delta$ && $|\hat{\mathbf{n}}_G \cdot (\mathbf{m}_G - \mathbf{m}_{\omega_c})| < \gamma$ && `mse(`$\mathbf{G} \cup \omega_c$`)` $< \epsilon$ **then**
10 $\mathbf{G} \leftarrow (\mathbf{G} \cup \omega_c)$;
11 $\mathbf{Q} \leftarrow (\mathbf{Q} \cup$`NN(`$\omega_c$`)`$)$;
12 **end**
13 **else**
14 **for** *each point* $\mathbf{p}_c$ *in* ω **do**
15 **if** $\hat{\mathbf{n}}_G \cdot (\mathbf{m}_G - \mathbf{p}_c) < \gamma$ && `mse(`$\mathbf{G} \cup \mathbf{p}_c$`)` $< \epsilon$ **then**
16 $\mathbf{G} \leftarrow \mathbf{G} \cup \mathbf{p}_c$;
17 $\mathbf{Q} \leftarrow \mathbf{Q} \cup$ `NN(`ω_c`)` ;
18 **end**
19 **end**
20 **end**
21 **end**
22 **if** `size(`$\mathbf{G}$`)` $\geq \theta$ **then**
23 $\mathcal{R} \leftarrow \mathcal{R} \cup \mathbf{G}$;
24 **else**
25 $\mathcal{R}' \leftarrow \mathcal{R}' \cup \mathbf{G}$;
26 **end**
27 **end**

into subwindows first and each subwindow is classified as planar or non-planar based on the method presented in Section 4.4, then the planar subwindows are put into a list Ω and the non-planar subwindows are put into another list Θ (Algorithm 4.4, line 2). When there are still unidentified planar subwindows, the subwindow ω with the minimum MSE among all unidentified planar subwindows is chosen as a new seed. A growing region $\mathbf{G}$ is initialized by ω, and its unidentified neighbors are put into a FILO queue $\mathbf{Q}$ (Algorithm 4.4, line 5). Then $\mathbf{G}$ is extended by investigating its neighbors. If the considering subwindow is planar, the criteria for determining whether to add it into $\mathbf{G}$ are the same as those in subwindow based region growing (Algorithm 4.4, line 9 – 12). If the subwindow is non-planar, each of its points will be investigated separately, and a point $\mathbf{p}_c$ will be added to $\mathbf{G}$ iff it passes the following tests (Algorithm 4.4, line 14 – 19).

1. The distance from the point to the optimal plane fitted to $\mathbf{G}$ is smaller than γ,

making sure $\mathbf{p}_c$ is a coplanar point of $\mathbf{G}$.

2. The plane fitting error of $\mathbf{G} \cup \mathbf{p}_c$ should be less than ϵ, ensuring the flatness of the segment acceptable.

This process will continue until no new neighbor of $\mathbf{G}$ can be found. Afterwards, if $\mathbf{G}$ has more points than a threshold θ, it will be viewed as a planar segment; otherwise it will be marked as uncertain area (Algorithm 4.4, line 22 – 26). The algorithm ends when every point has been assigned to either $\mathcal{R}$ or $\mathcal{R}'$. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are pre-defined thresholds. One parameter tuning step is needed for one specific range sensor.

4.5.2 Incremental plane parameter and MSE calculation

When adding a subwindow to $\mathbf{G}$, its plane parameter is computed as the same as that in the subwindow based region growing, see Section 4.4. What's more, adding a single point is just a special case of adding a subwindow, i.e., when there is only one point in the subwindow. As a result, the plane parameter can be updated as follows when a point $\mathbf{p}$ is added.

$$\begin{cases} \mathbf{M} &= \mathbf{m}_G K_G + \mathbf{p} \\ \mathbf{m} &= \mathbf{M}/(K_G + 1) \\ \mathbf{J} &= \mathbf{J}_G + \mathbf{p}\mathbf{p}^{\mathrm{T}} \\ \mathbf{S} &= \mathbf{J} - \mathbf{M}\mathbf{m}^{\mathrm{T}} \\ \hat{\mathbf{n}} &= \hat{\mathbf{e}}_{min}(\mathbf{S}) \\ d &= \hat{\mathbf{n}} \cdot \mathbf{m} \\ \mathrm{MSE} &= \frac{\lambda_{min}(\mathbf{S})}{K_G+1} \end{cases} \tag{4.16}$$

4.5.3 Computational complexity analysis

There are two growth unit in the algorithm, namely a single point or a planar subwindow. Therefore, the time complexity of hybrid region growing is between the point based and subwindow based region growing algorithm. Its time complexity is equal to the subwindow based algorithm when there are no non-planar subwindows, and is equal to the point based algorithm when there are no planar subwindows in the given point cloud. For a point cloud with n points, the time complexity of subwindow based algorithm is $O(\frac{n}{k} \log \frac{n}{k})$, where k is the subwindow size. The time complexity of point based algorithm is $O(n \log n)$. Consequently, the time complexity of the hybrid region growing algorithm is in the range $[O(\frac{n}{k} \log \frac{n}{k}), O(n \log n)]$, it increases with the clutter level of the environment and vice versa.

4.6 Cached octree region growing

One severe drawback of current fast region growing algorithms (see (Poppinga et al., 2008b; Georgiev et al., 2011)) and the three proposed algorithms in above sections is

that they are restricted to organized point clouds, since the pixel-neighborhood information is required in all of them for efficiency. To eliminate the dependency on pixel-neighborhood information, we propose an cached octree region growing algorithm in this section, which can deal with both organized and unorganized point clouds. Compared to aforementioned algorithms, it has potential applications such as map segmentation, plane-based map merging, and plane-based scan to map registration (localization).

4.6.1 Cached octree region growing algorithm

As mentioned in Section 4.1, efficient nearest neighbor search is important for region growing approaches. Hence, the pixel-neighborhood information is employed in existing fast plane segmentation algorithms, as well as the other three algorithms proposed in this chapter. This, however, requires image-like 2.5D range images, restricting the application area of the above approaches. On the one hand, octree based segmentation algorithms can operate on unorganized point clouds (known as full 3D point clouds), which can also provide the basis for map merging and scan to map registration. On the other hand, searching neighbors in octree is slower than in the image-like structure, which makes the octree based segmentation algorithms much slower than their counterparts using pixel-neighborhood information. They are so slow because the nearest neighbor search has to be performed multiple times for many points in the region growing procedure.

The main idea of cached octree region growing is to accelerate the algorithm by requiring a single nearest neighbor search trial for each point in the octree. As a result, a compromise is made between memory and speed by caching the indices of the nearest neighbors searched for each point. Three routines are available in octree: "K nearest neighbors (k-NN) search", "neighbors within voxel search" and "neighbors within radius search"; for the last two methods, the number of nearest neighbors is very different from point to point due to uneven sampling, inducing difficulties when allocating dynamic amounts of memory for the cached octree. On the contrary, it is easy to allocate memory for caching a constant number of neighbors at each point, so the k-NN search routine is chosen for constructing the cached octree. The additional memory required to hold the cached points is $4 \times K \times N$ bytes, where K is the number of neighbors to cache and N is the size of the point cloud. For a cloud with 400,000 points, about 38 MB of memory are needed for $K = 25$, which is a manageable size for modern computers.

The proposed algorithm proceeds as follows: if the point cloud is organized, only the valid points are taken into account (Algorithm 4.5, line 2 – 4). Afterwards, an octree is constructed for the valid points (Algorithm 4.5, line 6). For each 3D point, search its K nearest neighbors in the octree and cache their indices in an array CKNN($\mathbf{p}_i$) (Algorithm 4.5, line 7 – 9). Then, fit a local plane to each point with its cached neighbors using least squares with respect to MSE (Algorithm 4.5, line 10 – 12). While unidentified points exist (Algorithm 4.5, line 13), choose the point $\mathbf{p}_s$ which has minimum MSE among all unidentified points as a new seed (Algorithm 4.5, line 14). A new growing region $\mathbf{G}$ and its nearest neighbors $\mathbf{Q}$ (a FIFO queue) are initialized based on $\mathbf{p}_s$ and its cached neighbors (Algorithm 4.5, line 15). Thereafter, $\mathbf{G}$ is extended by considering

Algorithm 4.5: Cached octree region growing plane segmentation

Input: Ψ: a point cloud
Output: $\mathcal{R}$: planar segments, $\mathcal{R}'$: uncertain points

```
1  R ← ∅, R' ← ∅, G ← ∅, Q ← ∅ ;
2  if organized(Ψ) == true then
3  |  remove invalid points from Ψ ;
4  end
5  N=size(Ψ) ;
6  tree = octree(Ψ) ;
7  for i = 1 → N do
8  |  search KNN for p_i and cache their indices in CKNN(p_i) ;
9  end
10 for i = 1 → N do
11 |  local plane fitting for p_i ∪ CKNN(p_i);
12 end
13 while (Ψ \ (R ∪ R')) ≠ ∅) do
14 |  select p_s which has minimum MSE in Ψ \ (R ∪ R') ;
15 |  (G, Q) = initializeSeed(p_s,CKNN(p_i)) ;
16 |  while Q ≠ ∅ do
17 |  |  p_c = Q.pop() ;
18 |  |  if dis(plane(G∪p_c))< γ && mse(G∪p_c)< ε then
19 |  |  |  G ← G∪ p_c ;
20 |  |  |  Q ← Q ∪ NN(CKNN(p_c), δ) ;
21 |  |  end
22 |  end
23 |  if size(G)≥ θ then
24 |  |  R ← R ∪ G ;
25 |  else
26 |  |  R' ← R' ∪ G ;
27 |  end
28 end
```

its neighbors in $\mathbf{Q}$. Suppose the considering point is $\mathbf{p}_c$, it is added to $\mathbf{G}$ if it meets the following criteria (Algorithm 4.5, line 18).

1. The distance from the point to the optimal plane fitted to $\mathbf{G} \cup \mathbf{p}_c$ is smaller than γ, making sure $\mathbf{p}_c$ is a coplanar point of $\mathbf{G}$.
2. The plane fitting error of $\mathbf{G} \cup \mathbf{p}_c$ should be less than ϵ, this ensures the flatness of the segment is acceptable.

After $\mathbf{p}_c$ is added to $\mathbf{G}$, its cached neighbors within distance δ are added to $\mathbf{Q}$ (Algorithm 4.5, line 20). The growth will continue until no more points can be added to $\mathbf{G}$ (Algorithm 4.5, line 16 – 22). Afterwards, if $\mathbf{G}$ contains more than θ points, it will be assigned to be a new segment and added to the regions set $\mathcal{R}$. Otherwise, it is added to the uncertain area $\mathcal{R}'$ (Algorithm 4.5, line 23 – 26). The algorithm ends when

each point is assigned either to $\mathcal{R}$ or $\mathcal{R}'$. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are pre-defined thresholds which need to be tuned at the beginning.

4.6.2 Computational complexity analysis

The plane parameter and MSE computation is the same as that in the point based region growing algorithm. Given a point cloud of size N, generally, building the octree is of $O(N \log N)$ complexity. k-NN search in the constructed octree is at least $O(\log N)$. Since one k-NN search trial is needed for each point, the caching process has a complexity of $O(N \log N)$. After caching the octree, constant time is needed for fitting a local plane to each point, which in total is of $O(N)$ complexity. The main loop exhibits similar behavior since constant time is needed for each iteration. To sum up, the time complexity is $O(N \log N)$.

4.7 Experiments and results

All presented algorithms have been implemented in C++; the code has been published online, and please refer to Appendix A for access to it. Experiments have been carried out on standard desktop computers. The comparison between speeds in one figure has been run on the same computer, but not all experiments have been carried out on the same computer. The efficiency of linear algebra is crucial in the approaches, especially for calculating the eigenvalues and eigenvectors of a square matrix, because it has to be performed whenever a point or a subwindow is investigated during the region growing phase. Therefore the linear algebra library Eigen (Guennebaud et al., 2010), a C++ template library for linear algebra, has been employed. The Point Cloud Library (PCL) (Rusu and Cousins, 2011) has been utilized for point cloud reading and writing, nearest-neighbor search in octree, and 3D visualization. For figures illustrating segmentation results, the segments are colored randomly; therefore, one color may be patched to multiple segments. Due to space limitation, we cannot present all the segmentation results. Instead, some typical results have been selected for explanation.

To make the presentation simple and clear, the following abbreviations are used for the proposed algorithms in the following parts of this section:

- PBRG: point based region growing;
- SWRG: subwindow based region growing;
- HRG : hybrid region growing;
- CORG: cached octree region growing.

Furthermore, Kaustubh Pathak has kindly provided us the access to their code used in Poppinga et al. (2008b). Their algorithm serves as a baseline for comparison and is denoted as JIRRG. To save our time for parameter tuning, JIRRG has only been tuned for the Barcelona dataset. As a result, only algorithms have been applied to this dataset are compared to JIRRG directly, i.e., PBRG, HRG and CORG. SWRG, instead has been compared to PBRG. It should be noted that plane parameter uncertainties have also been computed during the region growing procedure in JIRRG, which has not been considered in the proposed algorithms. Therefore, the comparison is not fully impartial; however, the uncertainty computation does not induce much time complexity

Table 4.2: Tuned parameters for PBRG when being applied to the Barcelona dataset.

Parameter	δ	γ	ϵ	θ
Value	0.40	0.05	0.006	50

compared to the coplanar points detection, see JIRRG for detail. We have not compare the algorithms to the Hough Transform method (Borrmann et al., 2011), although their code is publicly available; it is because that each point cloud in our experiment contains more than 15 planar surfaces, and the Hough Transform has already been found to be much slower than PBRG in such situations.

In the experiments, only segmentation speed has been compared quantitatively while the segmentation result itself has only been qualitatively inspected manually. To the best of our knowledge, there is still no metric proposed for measuring plane segmentation quality in the literature. It would be very interesting to compare the segmentation results to the ground-truth, but there are difficulties to do this: on the one hand, ground-truth can be easily determined in simulation datasets, where the algorithms can produce satisfying results; on the other hand, it is hard to label ground-truth manually for real sensor data, particularly in complex outdoor environments. A compromise could be labeling a pseudo ground-truth using a "good" or the "best" algorithm, and other algorithms can be compared to this pseudo ground-truth with the aim to find the next "best" algorithm.

4.7.1 Organized point cloud segmentation

This section focuses on organized point cloud segmentation. As aforementioned, except CORG, the other three algorithms are limited to coping with organized point clouds. To benchmark the performance of PBRG, it has been applied to all 400 point clouds in the Barcelona dataset. The experimental tuned parameters have been detailed in Table 4.2. A typical result is depicted in Figure 4.5. As stated in Section 4.3, it is proposed to accelerate JIRRG without affecting the segmentation result. To evaluate the presented improvements, PBRG and JIRRG are benchmarked against each other with regard to segmentation speed as shown in Figure 4.6. It can be seen that both algorithms have a nice linear time complexity, and PBRG is faster than JIRRG; the mean time of them are 0.61 s and 0.86 s, respectively.

In particular, SWRG can only deal with structured environments; note that here structured means little or without occlusion, a typical result of it being applied to the Hamburg dataset has been depicted in Figure 4.7. Its problem when being applied unstructured environments has already been shown in Figure 4.4. HRG is proposed to deal with this problem while still maintain the advantage of having subwindow as a growth unit: the whole segmentation result of Figure 4.4(a) using HRG is illustrated in Figure 4.8.

To analyze the speed of SWRG and HRG, they have been compared with that of PBRG. Since the Hamburg dataset does not contain enough scans for statistical analysis, the indoor dataset in Kaushik et al. (2010) has been utilized. The tuned parameters for them are illustrated in Table 4.3. The segmentation time for each point

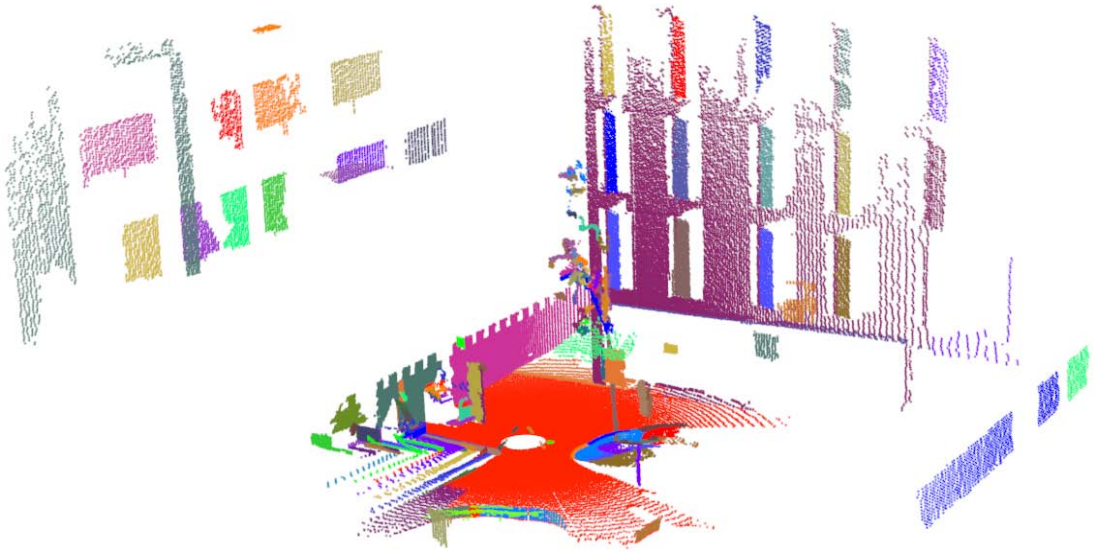


Figure 4.5: A typical segmentation result of PBRG; scan 4 in the Barcelona dataset has been used.

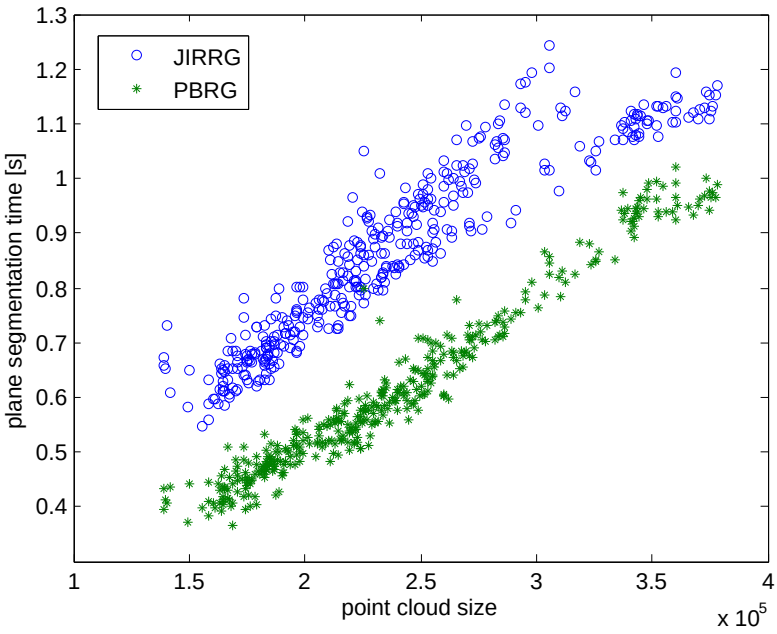


Figure 4.6: Plane segmentation time comparison between PBRG and JIRRG using the Barcelona dataset.

cloud with different algorithms is depicted in Figure 4.9, where the point cloud size corresponds to the valid point in each point cloud. Since most subwindows have a planar appearance, SWRG and HRG have approximately the same speed. On average, SWRG is about 4.5 times faster than PBRG.

According to its theoretical analysis, the time complexity of HRG varies along with the ratio of planar subwindows. To verify this property, it has been also applied

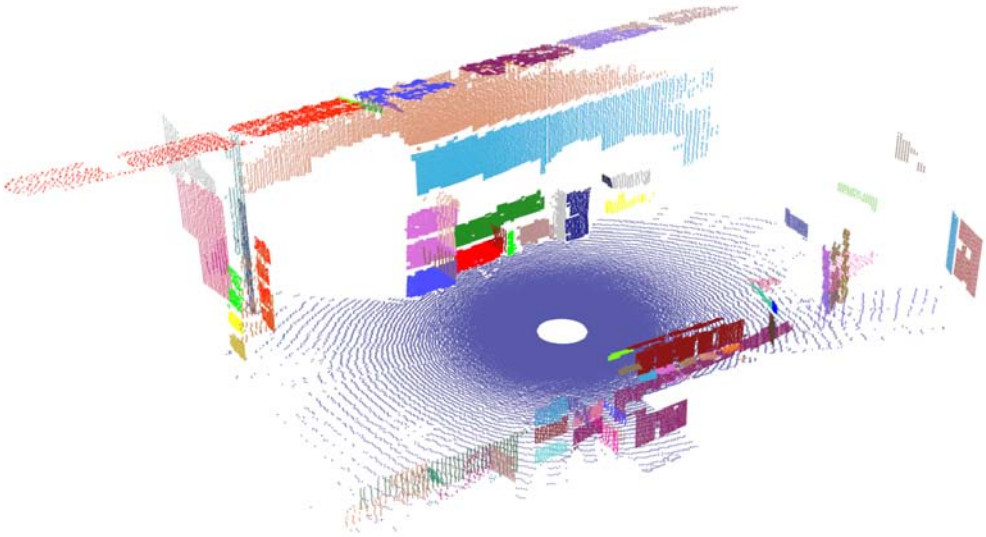


Figure 4.7: A typical segmentation result of SBRG. Scan 10 in the Hamburg dataset has been used.

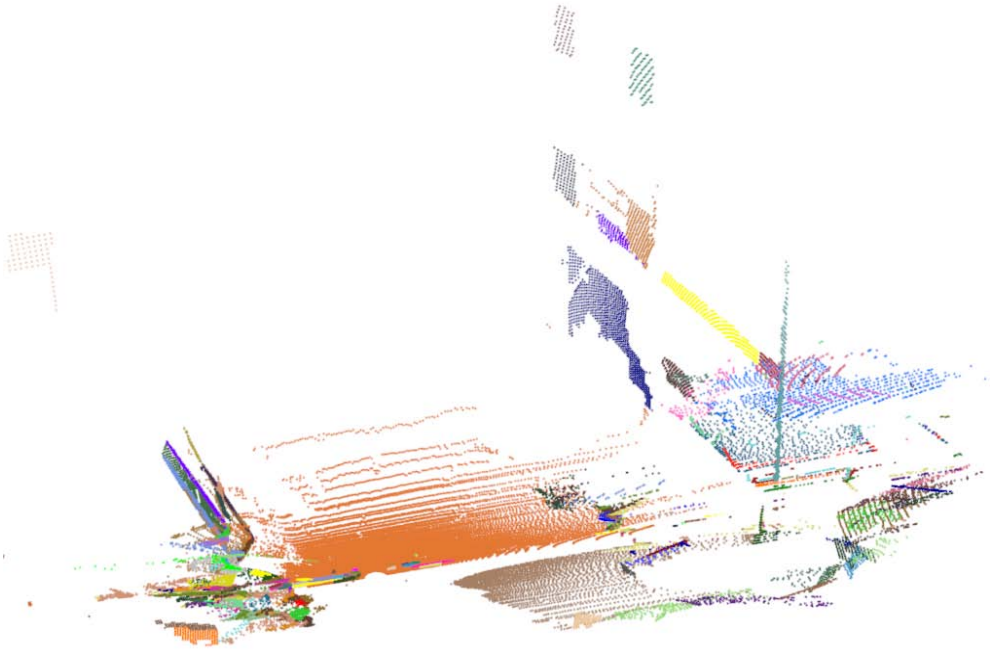


Figure 4.8: A typical segmentation result of HRG. Scan 6 in the Texas dataset has been used.

Table 4.3: Tuned parameters for SWRG and HRG when being applied to the indoor dataset, where the subwindow length has been fixed to 3.

Parameter	δ	γ	ϵ	θ
Value	0.95	0.035	0.001	50

Table 4.4: Tuned parameters for CORG when being to the Barcelona dataset.

Parameter	δ	γ	ϵ	θ
Value	0.4	0.05	0.001	50

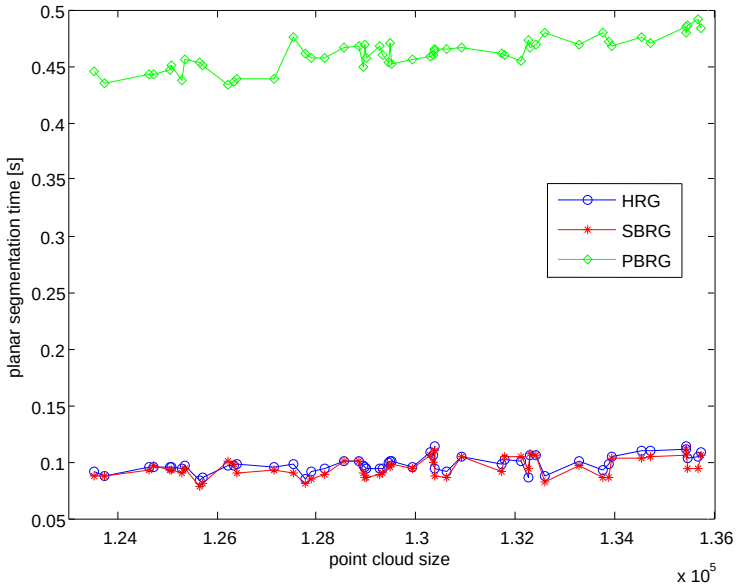


Figure 4.9: Benchmarking plane segmentation speeds of PBRG, SWRG, and HRG in an indoor scenario. Since most subwindows have a planar appearance, the speed of HRG is almost the same as SWRG.

to the Barcelona dataset and compared to PBRG again. The performances of them are detailed in Figure 4.10, where the subwindow side length 3 and 4 are investigated. Even larger subwindow side length has also been tested, which makes the narrow planar surfaces unidentified. Note the linear relation between the processing time and the point cloud sizes. The linearity of HRG is worse than that of the PBRG due to two kinds of growth units are used in it. When the subwindow size is increased, the segmentation speed is faster.

4.7.2 Unorganized point clouds segmentation

In CORG, K (the number of nearest neighbors whose indices are cached) should be selected properly because it is an important parameter affecting the speed. If K is too large, it can cause neighbor caching and local plane fitting to consume excess time. Moreover, outliers can be induced for local plane fitting. On the other hand, if K is too small, fitting a local plane to each point and its cached neighbors does not yield meaningful results. Different K values have been studied in the experiments; the smallest K we used is 15 in order to make the local plane fittings meaningful.

The tuned parameters for CORG have been shown in Table 4.4. A typical segmentation result is depicted in Figure 4.11. To benchmark the speed of CORG, its octree caching step is analyzed separately at different K values as shown in Figure 4.12. An

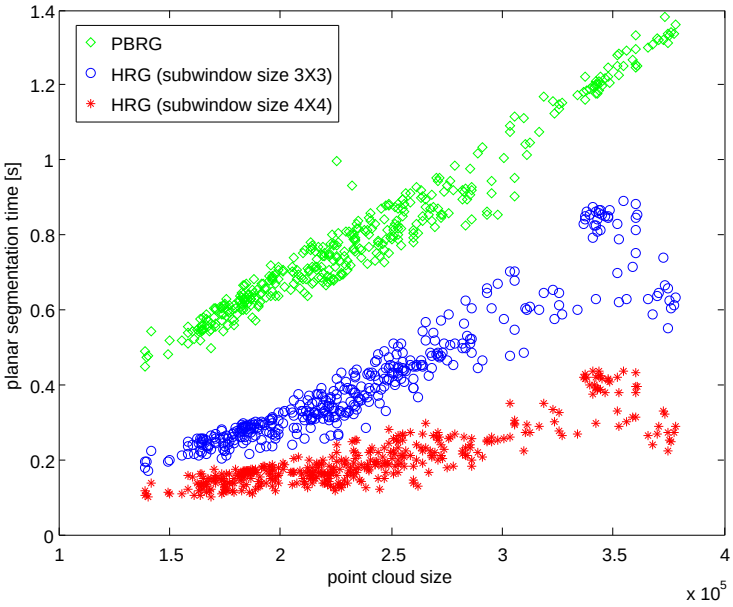


Figure 4.10: Benchmarking plane segmentation speed of HRG using the Barcelona dataset, two difference subwindow sizes have been used in the hybrid region growing algorithm. The PBRG has been employed as the baseline for comparison.

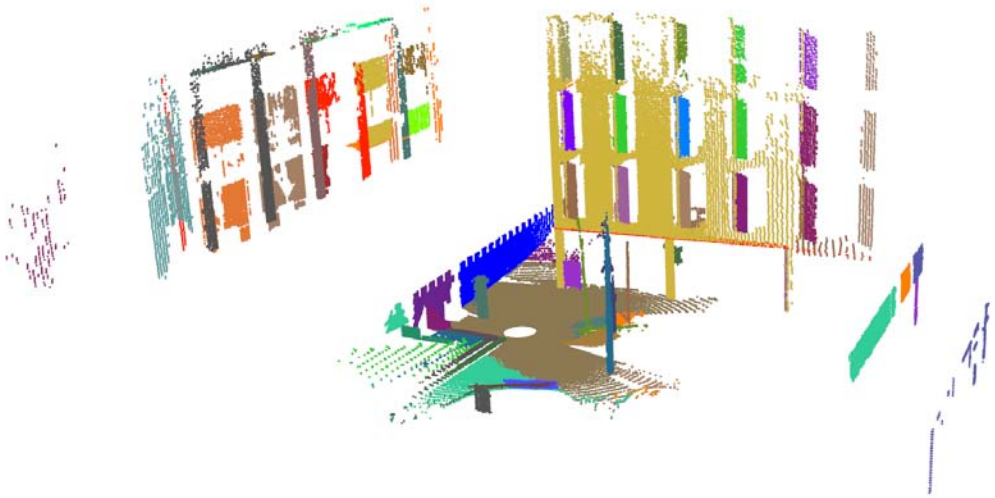


Figure 4.11: A typical segmentation result of CORG, scan 4 in the Barcelona dataset has been employed.

approximate linear relation between the size of the point clouds and the octree caching time can be observed [time complexity of $O(N \log N)$]. The elapsed time for extracting a segment depending on the number of associated points is depicted in Figure 4.13, which is independent of K. This linear relation verifies the theoretical time complexity analysis experimentally.

Again, JIRRG has been employed for segmentation speed comparison (JIRRG has been applied to the corresponding organized point clouds). Figure 4.14 shows the plane

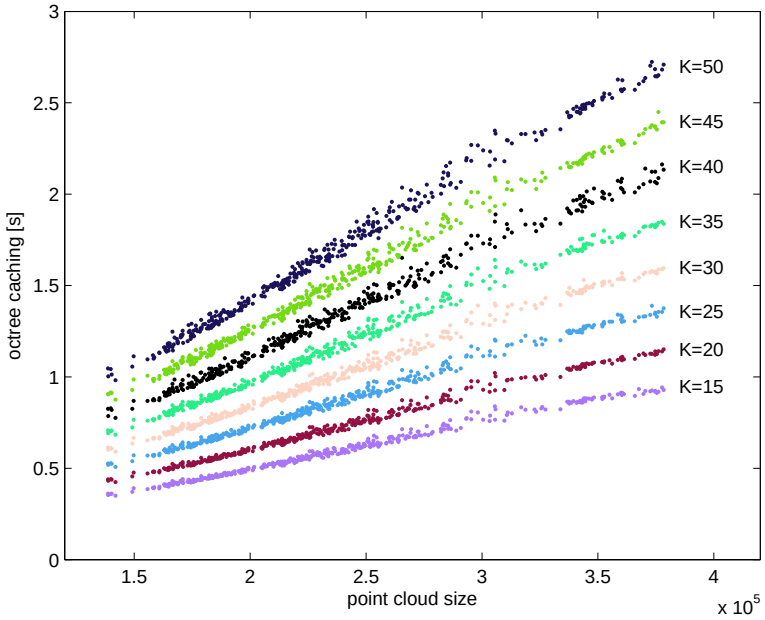


Figure 4.12: Octree caching time in CORG for different K values.

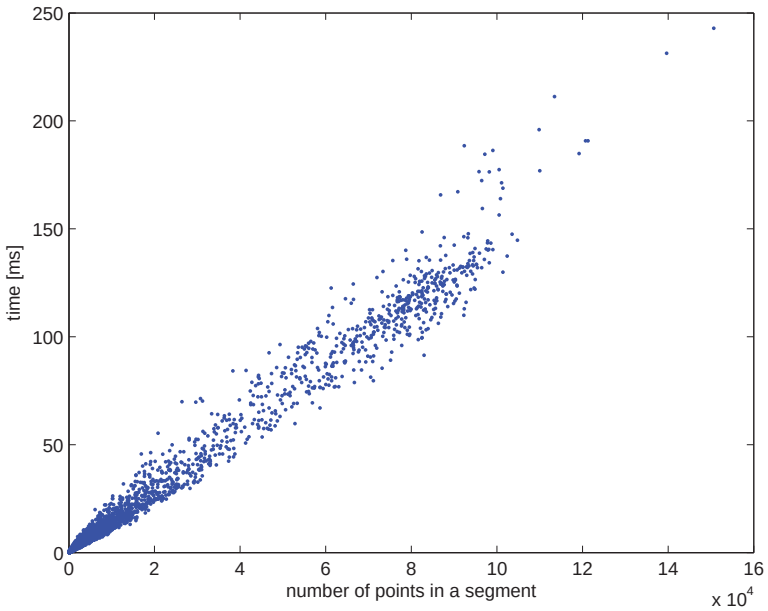


Figure 4.13: Elapsed time for extracting a segment according to the number of points on it.

segmentation time of both algorithms. Comparing it to Figure 4.12, K introduces a time shift resulting from octree caching. Again, this verifies that K has no effect on the second step of CORG. Furthermore, the speed performance of CORG with $K = 15$ is similar to that of JIRRG, with CORG being slightly slower.

As stated in Section 4.6, CORG does not require an image-like structure for segmentation, which is an advantage compared to JIRRG and the other three proposed

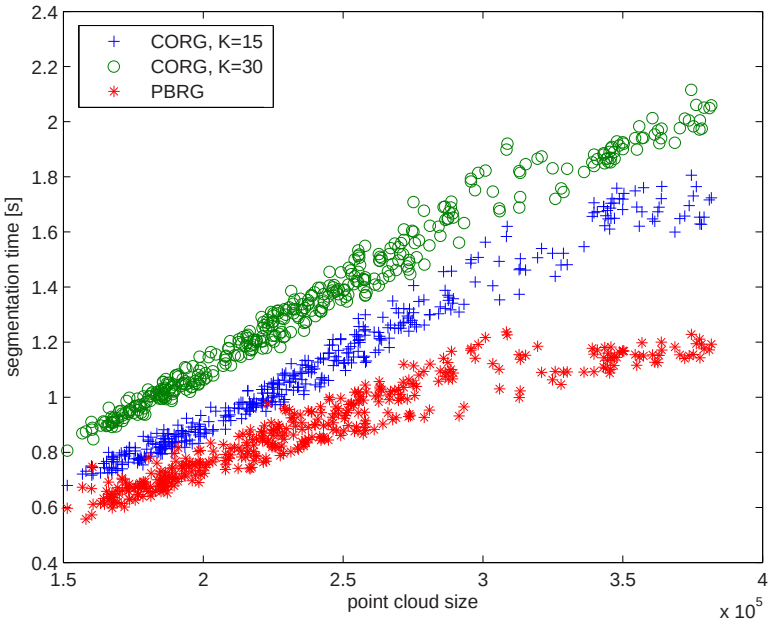


Figure 4.14: The JIRRG and CORG plane segmentation time for 3D point clouds. CORG was evaluated with regard to caching different amounts (K) of nearest-neighbor indexes.

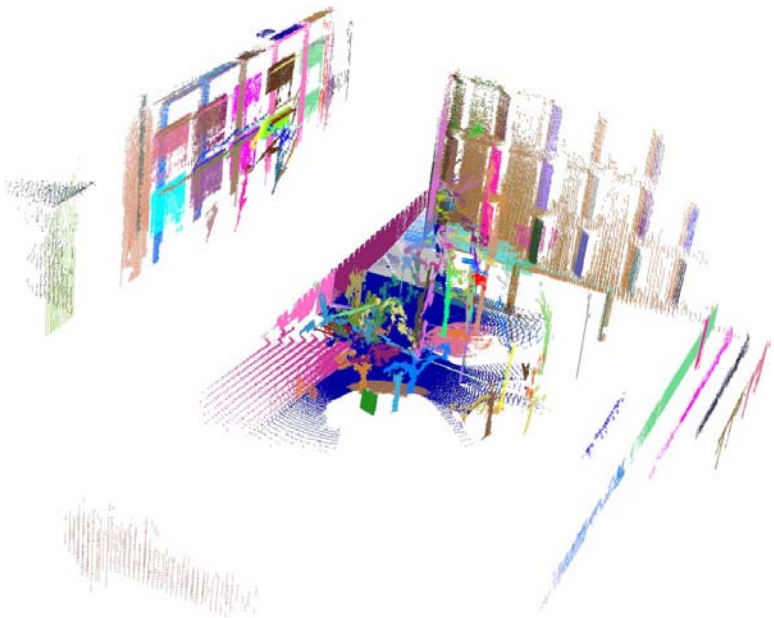


Figure 4.15: CORG plane segmentation on the incremental registration result from scan 2 to 6 in the Barcelona dataset.

algorithms. But this advantage cannot be inferred from the above experiment, since range images can be easily constructed by projection for such point clouds, i.e., each scan is obtained at a stationary viewpoint. However, for a point cloud sampled from multiple viewpoints, such as the registration result of multiple scans, a range image is impossible if occlusions occurs among the viewpoints. In such situations, CORG

Table 4.5: Comparison of the proposed algorithms.

	PBRG	SWRG	HRG	CORG
organized point cloud	√	√	√	√
unorganized point cloud	×	×	×	√
environment clutter level	++	+	++	++
segmentation speed	++	++++	+++	+

can still efficiently segment the point cloud into planar surfaces. To demonstrate this ability, we have applied CORG to the incremental registration result from scans 2 to 6 in the Barcelona dataset, as shown in Figure 4.15. This can enlarge the application area of plane-based mapping techniques.

4.7.3 Discussion

A comparison between the presented algorithms is depicted in Table 4.5. According to the evaluations, all algorithms can be applied to organized point clouds. Among them, SWRG can only be employed in structured environments where HRG has a similar speed; PBRG and CORG are much slower in such situation. However, HRG can also be used in more cluttered environments at a faster speed than that of PBRG. As a result, HRG is recommended for organized point cloud plane segmentation. For unorganized point clouds, CORG is able to detect planar surfaces at an acceptable speed compared to that of PBRG.

4.8 Summary

Four complementary region growing strategies have been presented in this chapter, namely a point based region growing algorithm in Section 4.3, a subwindow based region growing algorithm in Section 4.4, a hybrid region growing algorithm in Section 4.5, and a cached octree region growing algorithm in Section 4.6. They are then evaluated in Section 4.7, with regard to applicability and speed. It is found that the hybrid region growing algorithm is most suitable for dealing with organized point clouds; and the cached octree algorithm provides an efficient solution for unorganized point cloud segmentation, which has potential application areas such as plane based map merging and scan to map registration.

Chapter 5

Planar segment area calculation

> To study and not think is a waste. To think and not study is dangerous.
>
> Confucius

USING the techniques presented in Chapter 4, each point cloud can be segmented into a set of planar segments. This chapter concentrates on determining the covered area of each segment. Again, planar segments from *organized* and *unorganized point clouds* are distinguished and treated with different methods. No matter in which method, the first step is to decompose each segment into small pieces, and the second step is to sum up the areas of small pieces. If the segment is from an organized point cloud, the underlying image-like structure is employed to find small triangles and quadrilaterals; otherwise, its points are projected onto the optimum fitting plane first and available triangulation algorithms are used to divide it into triangles. As is known, finding segment correspondences is a core problem in plane-based scan registration approaches. The resulting area will be used as a similarity metric for determining segment correspondences in Chapter 6.

This chapter is organized as follows: The related work will be given in Section 5.1. Then, the problem statement and data preprocessing is introduced in Section 5.2. Afterwards, we will present three alternative approaches, i.e., a Delaunay triangulation based method in Section 5.3, an alpha-shape based method in Section 5.4, and a method for organized point clouds in Section 5.5. The methods will be benchmarked in Section 5.6 with regard to accuracy and speed. Afterwards, a discussion about the distinction between the area and point number associated to a segment is given in Section 5.7. In the end, Section 5.8 concludes this chapter.

5.1 Related work

For a given set of 2D points (a planar segment in this dissertation), many problems have received considerable attention in computational geometry, such as triangulation (Yvinec, 2012), convex hulls (Hert and Schirra, 2012), and shape reconstruction (Da,

2012). Determining the area covered by a set of points, on the other hand, has not been widely considered.

In Dold (2005), all scan points associated to a planar segment are projected onto the optimum fitted plane, then the convex hull of those points is determined. Afterwards, the area of the corresponding segment is calculated from the boundary points. Unfortunately, the employment of convex hull induces a main drawback for real-word datasets. The convex hull algorithms (see Hert and Schirra (2012)) are designed for convex subsets, their output is always a convex polygon regardless of whether the coplanar points form a concave or convex shape. However, the shape of a planar segment from point clouds is not limited to convex, which restricts the algorithm's application area. In Kohlhepp et al. (2004), a split-and-merge procedure is employed for planar segment detection, wherein the outer and inner 3D boundary polygons are also extracted. Then the area of each patch is calculated based on its outer and inner boundaries, but no detail of the method has been given.

In He et al. (2005), the area of each segment is calculated by $s = Ndxdy/\cos\theta$, where θ is the angle between the plane normal and the z-axis, dx and dy correspond to the average intervals between projected points along the x-axis and y-axis respectively, and N is the number of area elements. Fischer and Kohlhepp (2000) just stated that the surface area equals the "number of square units covered by the patch, in the unit of measurement of range", which is similar to the former approach. This idea is termed *number of square units* in the following parts of this chapter; a setback of this concept is its bad accuracy when it comes to area calculation, see Section 5.6 for a quantitative analysis.

5.2 Data preprocessing

The problem can be stated as follows: Given a planar patch $\mathcal{P}$ which is segmented from a point cloud, calculate the area covered by the points associated to it. Ideally, those points associated to the segment should be exactly coplanar. However, they are just approximately on the plane in reality because many surfaces in the environments are not ideally plane and robotic sensors are noise-prone. As a result, the points should be projected on their least squares fitting plane, for the sake of using 2D computational geometry algorithms.

There are two kinds of projection, namely orthogonal projection and beam-along projection, which have been clarified in Figure 5.1. Suppose the plane equation of the segment is $\hat{\mathbf{n}} \cdot \mathbf{p} = d$, the orthogonal projection $\dot{\mathbf{v}}$ and the beam-along projection $\ddot{\mathbf{v}}$ of point $\mathbf{p}$ can be calculated as

$$\dot{\mathbf{v}} = \mathbf{p} + (d - \hat{\mathbf{n}} \cdot \mathbf{p})\hat{\mathbf{n}} \tag{5.1}$$

and

$$\ddot{\mathbf{v}} = \frac{d}{\hat{\mathbf{n}} \cdot \mathbf{p}}\mathbf{p}. \tag{5.2}$$

The orthogonal projection is chosen for area calculation in this work since beam-along projection has a severe disadvantage. This can be found in Figure 5.1; the beam-along projection will create significant distortion when the angle between the laser

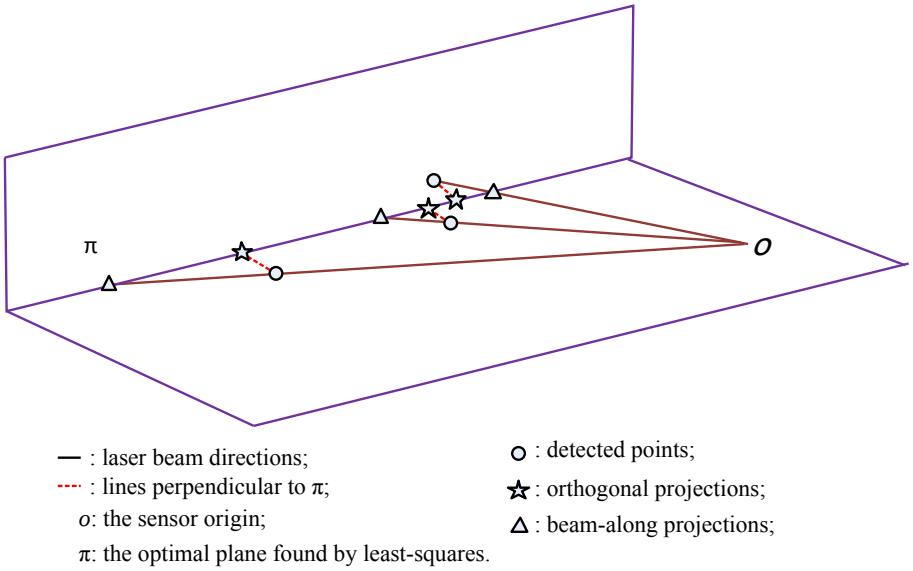


Figure 5.1: Projection of the points from a segment (resulted from plane segmentation) to their least-square fitted optimum plane using two methods: orthogonal projection and beam-along projection.

beam and the scanned surface is very small. This can happen because the orthogonal distance has been utilized during the segmentation process (see Chapter 4), which cannot guarantee an upper limit for the distance between each point and its beam-along projection. The distortion can also be analyzed theoretically in Eq. (5.2), $\|\ddot{\mathbf{v}}\|$ will be much larger than $\|\mathbf{p}\|$ if $\hat{\mathbf{n}} \cdot \mathbf{p} \to 0$.

Note that those points are still in 3D space after projection although they can only span a 2D subspace. As a result, they can be transformed into a 2D coordinate system by rotating the plane in order to align its normal along with an axis (z-axis in this thesis). After rotation, all points will have the same z value which can be eliminated. The rotation matrix is resolved by the eigendecomposition of the points' scatter matrix $\mathbf{S}$ ($\mathbf{S}$ is computed in 4.5). Suppose the three eigenpairs of the decomposition are $\langle \lambda_i, \hat{\mathbf{x}}_i \rangle, i = 1, 2, 3$, wherein $\lambda_1 \geq \lambda_2 \geq \lambda_3$, the rotation matrix $\mathbf{R}$ is constructed as

$$\mathbf{R} = [\hat{\mathbf{x}}_1 \ \hat{\mathbf{x}}_2 \ \hat{\mathbf{x}}_3]^{\mathrm{T}}. \tag{5.3}$$

Now we have both 2D and 3D coordinates of the coplanar points, therefore the area covered by these points can be resolved in either 2D or 3D. From the top-down point of view, the area covered by them can be calculated if their shape (usually as polygon) can be determined. Available algorithms for reconstructing the covered shape of coplanar points can be found in computational geometry, such as 2D alpha-shape (Da, 2012).

From a bottom-up point of view, if the segment can be divided into small simple components whose area can be calculated, the area of a segment can be calculated by summing up these small areas. The ideas of aforementioned related works He et al. (2005); Fischer and Kohlhepp (2000) fall into this methodology. There are also available algorithms for the subdivision task, i.e., 2D Delaunay triangulation (Yvinec, 2012). However, if the segment is from an organized point cloud, the image-like structure can

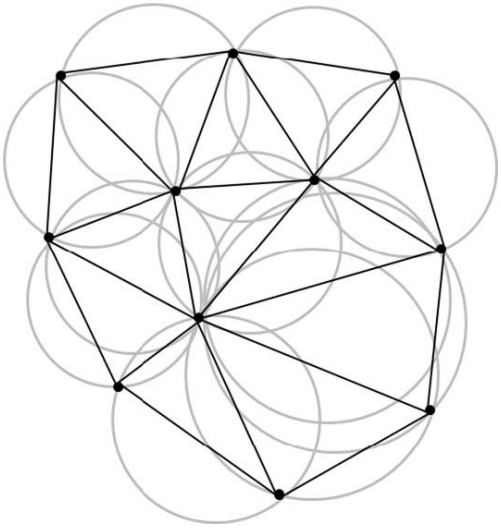


Figure 5.2: An example of Delaunay triangulation. Note that none of the points is inside the circumcircle of other triangles.

Algorithm 5.1: Delaunay triangulation based segment area calculation.

Input: a planar segment
Output: area covered by the segment
1 project the points to their optimum fitting plane ;
2 transform the points into 2D space ;
3 decompose the points into triangles using Delaunay triangulation ;
4 summing up the areas of triangles ;

be employed to divide the segment. Based on this observation, we will propose a novel algorithm for this specific condition termed area by summing up faces from range image (ASUFRI). In particular, this method resolves the area directly in 3D. We will present three different methods for area calculation in the following sections.

5.3 2D Delaunay triangulation based area calculation

In computational geometry, a Delaunay triangulation for a set of coplanar points is a triangulation such that no point locates inside the circumcircle of any resulting triangle. Theoretically, Delaunay triangulations try to maximize the minimum angle of all angles of the triangles in the triangulation in order to avoid skinny triangles. It is named after Boris Delaunay for his work on this topic. One example of Delaunay triangulation has been illustrated in Figure 5.2, where the circumcircle of each triangle has also been shown.

The Delaunay triangulation based area calculation is detailed in Algorithm 5.1, the first two steps (line 1,2) of which have been introduced in Section 5.2. The drawback of Delaunay triangulation when being applied to calculate the covered area of a planar segment can be found in Figure 5.3(b), i.e., it does not distinguish whether a triangle is from a hole or a concave part; the corresponding triangles have been filled using

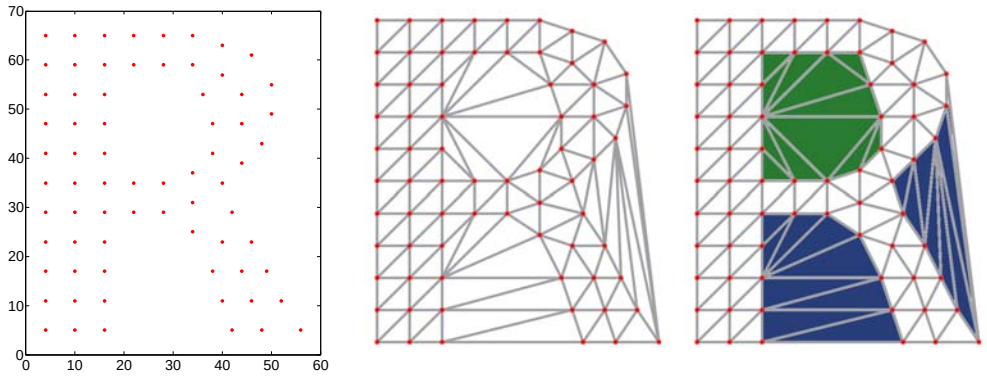


Figure 5.3: An example of 2D Delaunay triangulation. (a): the dataset; (b): the Delaunay triangulation with edges in light gray; (c): the Delaunay triangulation, triangles belong to the hole and concave parts are filled with green and blue, respectively.

different colors in Figure 5.3(c). As a result, the area accuracy of this approach is not guaranteed when dealing with real-world datasets. Its accuracy and speed will be benchmarked in Section 5.6, with comparison to other methods.

5.4 2D Alpha Shapes based area calculation

Given a set $\mathcal{S}$ of points in 2D, the alpha-shape (or α-shape) of $\mathcal{S}$ forms a discrete family of piecewise linear simple curves, wherein each is a linear approximation of the original shape according to a specific α ($0 \leq \alpha \leq \infty$). They were first defined by Edelsbrunner et al. (1983). The alpha-shape of a set of points is a generalization of the concept of the convex hull, i.e. every convex hull is an alpha-shape but not every alpha-shape is a convex hull. In contrast to the Delaunay triangulation, alpha-shape can be used to reconstruct the surface of an object with correct topology, see Stelldinger (2008).

As mentioned in Edelsbrunner and Mücke (1994), one can intuitively think of an alpha-shape as the following:

> Think of $\mathbb{R}^3$ filled with Styrofoam and the points of $\mathcal{S}$ made of more solid material, such as rock. Now imagine a spherical eraser with radius α. It is omnipresent in the sense that it carves out Styrofoam at all positions where it does not enclose any of the sprinkled rocks, that is, points of $\mathcal{S}$. The resulting object will be called the α-hull.

In 2D, the eraser is just a circle, an explanation of this process is illustrated in Figure 5.4. Clearly, alpha-shape depends on a parameter α. On the one hand, a very small value will allow us to carve out all Styrofoam, hence the alpha shape degenerates to the point set. On the other hand, a very huge value will prevent us from carving the inside of the convex hull of $\mathcal{S}$, hence the alpha-shape for $\alpha \to \infty$ is the convex hull of $\mathcal{S}$. The effect of changing α has been illustrated in Figure 5.5.

The alpha-shape of $\mathcal{S}$ is closely related to alpha complexes, which are sub-complexes of its Delaunay triangulation. In particular, each edge or triangle of the triangulation

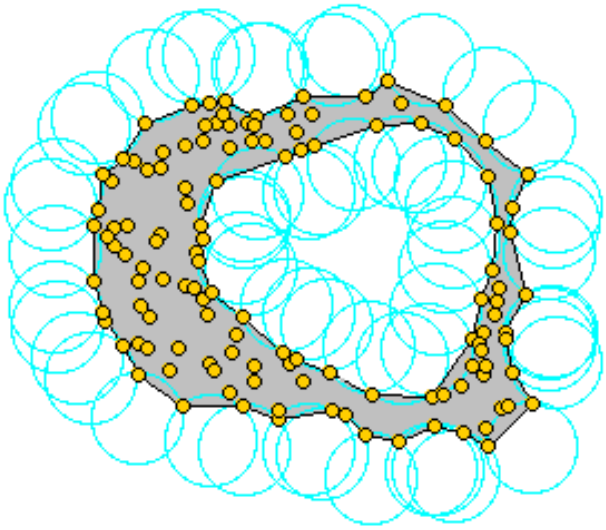
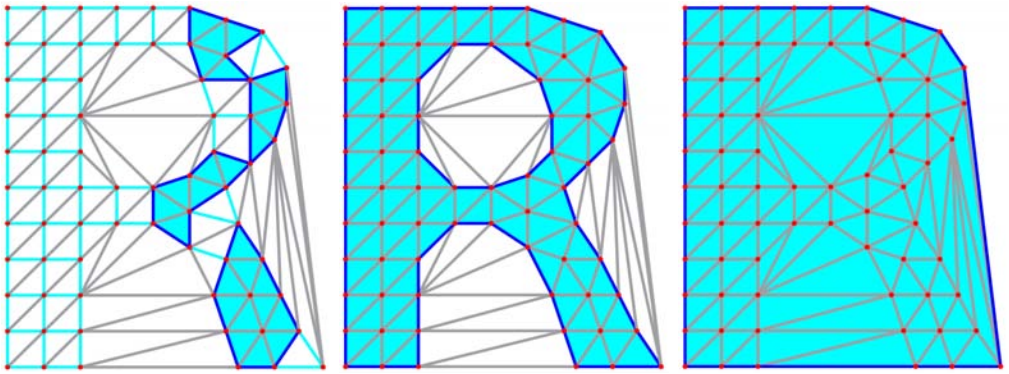


Figure 5.4: Explanation of the alpha-shape idea, note that the alpha-shape algorithm can deal with dataset with holes and concave shape. The figure is reprinted from Da (2012).



(a) (b) (c)

Figure 5.5: Illustration of the relation between the α value and the resulting α-shape. Red points represent the given dataset, light gray lines show the corresponding Delaunay triangulation edges, while the singular and regular α-edges appear as light cyan and blue lines, respectively. The α-edges form a subset of Delaunay triangulation edges, whereas the blue area represents the resulting α-shape. The coordinates of the dataset is referred to Figure 5.3. (a): The resulting α-shape at α=17.6; (b): The optimum α-shape at α=22.8; (c): The α-shape at $\alpha \to \infty$, which equals the convex hull of the dataset.

is given a characteristic radius, i.e., the radius of the smallest empty circle containing the edge or triangle. For each α value, the α-complex of $\mathcal{S}$ is the complex formed by the set of edges and triangles whose radii are at most $1/\alpha$. The union of the cells in the alpha complex is called α-shape. Relying on this fact, the α-shape can be computed efficiently and relatively easily. As a result, the α-shape of $\mathcal{S}$ forms a discrete family, although they are defined for all real numbers ($0 \leq \alpha \leq \infty$), because there is a finite set of triangles for a finite dataset.

Algorithm 5.2: Alpha-shape based segment area calculation.

Input: a planar segment
Output: area covered by the segment
1 project the points to their optimum fitting plane ;
2 transform the points into 2D space ;
3 reconstruct the shape (polygon) formed by the points using alpha-shape;
4 calculate the area of the polygon ;

Now the problem is how to determine the α value which results in a proper approximation of the shape for the given dataset. According to the literature, the optimization criteria have been defined as to finding a value which generates a shape containing all data points and having less than a given number of connected components. Although strategies for searching optimum α values do exist, they cannot always guarantee human-satisfying results. As a compromise, finding the ideal α is usually an interactive process.

However, since we are trying to embed it into an autonomous robotic system, it won't allow us to choose an α value for each planar segment manually. Even it is allowed, it will be time consuming considering there are tens of segments in each scan. Therefore, the optimum α value for each segment is searched by finding the shape including all data points and having a single connected component, using the implementation of Da (2012). The process of alpha-shape based area calculation is depicted in Algorithm 5.2, which is easy to be understood. For the last step (Algorithm 5.2, line 4), i.e., polygon area calculation, the underlying triangulation is employed. In other words, the polygon area is resolved by summing up the area of triangles in the alpha-complex. This approach is benchmarked with regard to accuracy and speed using real-world datasets in Section 5.6, as well as comparison with other approaches.

5.5 Area by summing up faces from range image

In this section, we propose a method for calculating the area of segments from organized point clouds. The idea is also to use the sum of small sub-surfaces to approximate the segment's area. Compared to the Delaunay triangulation based method, it has the following three differences: First, it does not require the projection from raw data to the fitted plane. Second, the image-like structure is utilized for segment decomposition. Third, the sub-surface is not limited to being triangle. Clearly, there are two underlying research questions, namely how to acquire sub-surfaces and how to calculate the local surfaces' area in 3D. Note that, the first sub-problem should be distinguished from getting triangles from a triangulation of the dataset, because triangulation methods do not care about whether a triangle locating inside of the segment. The first problem is solved in Section 5.5.1, followed by answering the second question in Section 5.5.2.

5.5.1 Local faces from a 2.5D range image

The first sub-problem is solved by the pixel-neighborhood information in the organized point cloud. The idea to use the pixel-neighborhood information here differs from its

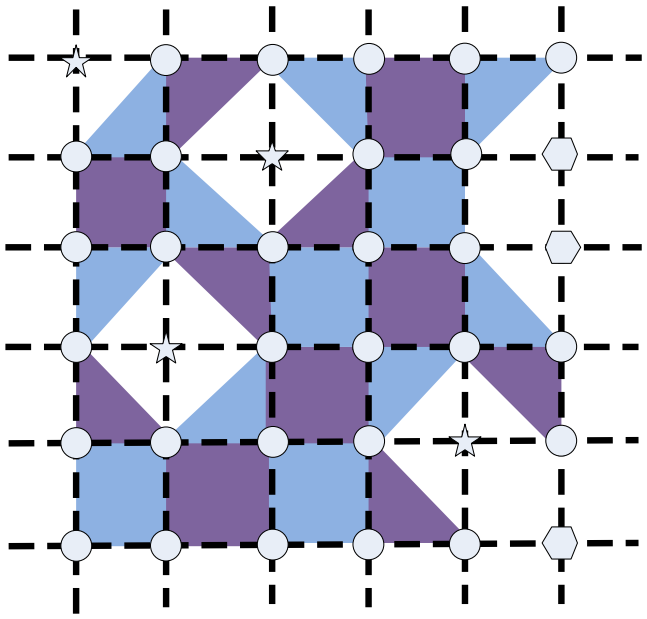


Figure 5.6: Divide the segment into local triangles and rectangles (shadowed area in the figure, two colors have been utilized for visualization) using the image-like structure; only part of the point cloud related to the planar patch is drawn here. The meaning of other items in the figure: vertical and horizontal dashed lines denote rows and columns of the image-like structure; points associated to the planar patch, valid points that are not on the planar patch, and invalid points are represented by circles, hexagons, and pentagrams.

usage in region growing (see Chapter 4). In this stage, it is already known which points are coplanar and on the same surface, the pixels are not used for neighbor search but for finding small triangles and rectangles for segments which are defined on the image-like structure. The idea is illustrated in Figure 5.6. Additionally, the triangles remain triangles when projecting them from pixel back to 3D space, while the rectangles become general quadrilaterals.

5.5.2 3D polygon area calculation

This section deals with how to calculate the area of sub-surfaces generated from Section 5.5.1. As mentioned, a rectangle in the pixel-space becomes a general quadrilateral in the spatial space; in other words, it could be convex or concave. A straightforward idea to calculate its area is to divide it into two triangles and sum up their areas. There are obviously two ways to divide a quadrilateral into two triangles, independent of whether it is convex or concave. As shown in Figure 5.7, there is no problem when it is convex. However, if the quadrilateral is concave, simply summing up the areas of two triangles may be incorrect, as shown in Figure 5.7(d). This problem can be overcome with an additional check to choose the subdivision with a smaller area, which is not desirable. Instead, we present the following method which is more efficient.

We begin with a more general case, i.e., determining the area of a planar polygon in 3D. As illustrated in Figure 5.8, a planar polygon Ω is defined on the infinite plane π in 3D space with unit surface normal $\hat{\mathbf{n}}$, and it has ω vertices $\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_{\omega-1}, \mathbf{v}_\omega$,

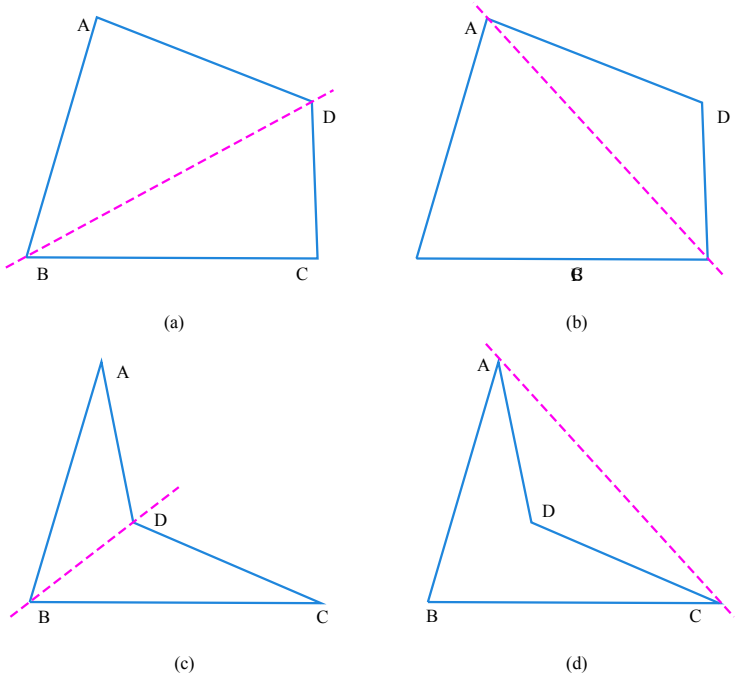


Figure 5.7: Dividing a quadrilateral into two triangles. (a) and (b): two situations for a convex; (c) and (d): two situations for a concave.

where $\mathbf{v}_\omega = \mathbf{v}_0$. The polygon is simple[1] but not limited to being convex. The problem is to determine the size of the area covered by Ω. In order to acquire the area, first fix an arbitrary point $\mathbf{q}$, which is not necessarily on π. Then $\mathbf{q}$, $\mathbf{v}_0$ and $\mathbf{v}_1$ will construct a triangle $\mathbf{q}\mathbf{v}_0\mathbf{v}_1$, other adjacent vertices will similarly construct necessary triangles with $\mathbf{q}$. Suppose $\mathbf{q}\mathbf{q}'$ is perpendicular to π, and $\mathbf{q}'$ is on π, then $\mathbf{q}'\mathbf{v}_0\mathbf{v}_1$ will form a triangle which is the projection of $\mathbf{q}\mathbf{v}_0\mathbf{v}_1$ on π. Other adjacent vertices obey the same rule. Clearly, the summed area of $\mathbf{q}'\mathbf{v}_0\mathbf{v}_1, \mathbf{q}'\mathbf{v}_1\mathbf{v}_2, ...\mathbf{q}'\mathbf{v}_{\omega-1}\mathbf{v}_\omega$ is equal to that of Ω. Now the problem becomes how to calculate the area of each triangle $\mathbf{q}'\mathbf{v}_{i-1}\mathbf{v}_i$.

Taking $\mathbf{q}'\mathbf{v}_0\mathbf{v}_1$ for an example, draw a line perpendicular to $\mathbf{v}_0\mathbf{v}_1$ from $\mathbf{q}'$, intersecting $\mathbf{v}_0\mathbf{v}_1$ at point $\mathbf{o}$, suppose the angle between π and $\mathbf{q}\mathbf{v}_0\mathbf{v}_1$ is θ. The area of $\mathbf{q}'\mathbf{v}_0\mathbf{v}_1$ can be calculated as $s_{01} = \frac{1}{2}\|\mathbf{v}_0\mathbf{v}_1\|\|\mathbf{q}'\mathbf{o}\| = \frac{1}{2}\|\mathbf{q}\mathbf{o}\|\|\mathbf{v}_0\mathbf{v}_1\|\cos(\theta)$. Since $(\mathbf{q}\mathbf{v}_0 \times \mathbf{q}\mathbf{v}_1) \perp \mathbf{q}\mathbf{v}_0\mathbf{v}_1$ and $\hat{\mathbf{n}} \perp \pi$, the angle between $\mathbf{q}\mathbf{v}_0 \times \mathbf{q}\mathbf{v}_1$ and $\hat{\mathbf{n}}$ equals to θ. As it is known, $\|\mathbf{q}\mathbf{o}\|\|\mathbf{v}_0\mathbf{v}_1\| = \|\mathbf{q}\mathbf{v}_0 \times \mathbf{q}\mathbf{v}_1\|$, this yields $s_{01} = \frac{1}{2}\|\mathbf{q}\mathbf{v}_0 \times \mathbf{q}\mathbf{v}_1\|\cos(\theta) = \frac{1}{2}(\mathbf{q}\mathbf{v}_0 \times \mathbf{q}\mathbf{v}_1) \cdot \hat{\mathbf{n}}$. Note that, s_{01} will be negative if θ is an obtuse angle.

Applying the above process to all the triangles in π, the area of Ω can be calculated as

$$s(\Omega) = \frac{1}{2}\hat{\mathbf{n}} \cdot \sum_{i=1}^{\omega-1} \mathbf{q}\mathbf{v}_i \times \mathbf{q}\mathbf{v}_{i+1}. \tag{5.4}$$

[1]In computational geometry, a simple polygon is defined as a flat shape consisting of straight, non-intersecting line segments that are joined pair-wise to form a closed path.

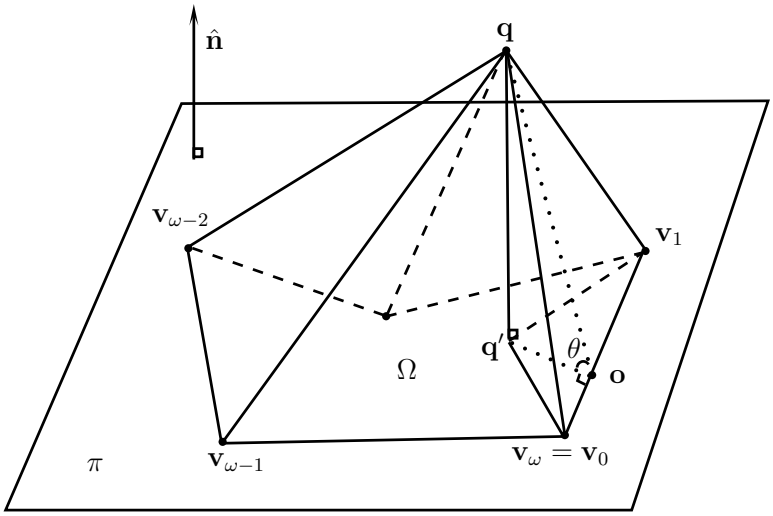


Figure 5.8: Explaining the idea of calculating the area of a 3D planar polygon Ω when the surface normal is known. The polygon should be simple, however, it is not limited to being convex but can also be concave.

If $\mathbf{q}$ is chosen as the coordinate origin, Eq. (5.4) is simplified to

$$s(\Omega) = \frac{1}{2}\hat{\mathbf{n}} \cdot \sum_{i=1}^{\omega-1} \mathbf{v}_i \times \mathbf{v}_{i+1}. \tag{5.5}$$

The triangular and planar quadrilateral areas are two special cases of a 3D planar polygon when ω is 3 or 4. However, Eq. (5.5) holds true only when all the vertices are ideally coplanar, which is not the case in the result of point cloud plane segmentation.

As mentioned in Section 5.2, orthogonal projection has been chosen in this research. Now we draw the mathematical derivation for area calculation using orthogonal projection. Substituting Eq. (5.1) into (5.5), we get

$$s(\Omega) = \frac{1}{2}\hat{\mathbf{n}} \cdot \sum_{i=1}^{\omega-1} (\mathbf{p}_i + \tau_i\hat{\mathbf{n}}) \times (\mathbf{p}_{i+1} + \tau_{i+1}\hat{\mathbf{n}}), \tag{5.6}$$

where $\tau_i = d - \hat{\mathbf{n}} \cdot \mathbf{p}_i$. The expansion of Eq. (5.6) is

$$s(\Omega) = \frac{1}{2}\hat{\mathbf{n}} \cdot \sum_{i=1}^{\omega-1} (\mathbf{p}_i \times \mathbf{p}_{i+1} + \tau_i\hat{\mathbf{n}} \times \mathbf{p}_{i+1} + \mathbf{p}_i \times \tau_{i+1}\hat{\mathbf{n}} + \tau_i\tau_{i+1}\hat{\mathbf{n}} \times \hat{\mathbf{n}}). \tag{5.7}$$

Each of the latter three terms at the right hand side of Eq. (5.7) equals zero, which yields

$$s(\Omega) = \frac{1}{2}\hat{\mathbf{n}} \cdot \sum_{i=1}^{\omega-1} \mathbf{p}_i \times \mathbf{p}_{i+1}. \tag{5.8}$$

In other words, we do not need to project the raw sensor points onto the optimum plane for calculating the segments' area. This is an advantage compared to other methods introduced in Section 5.3 and Section 5.4.

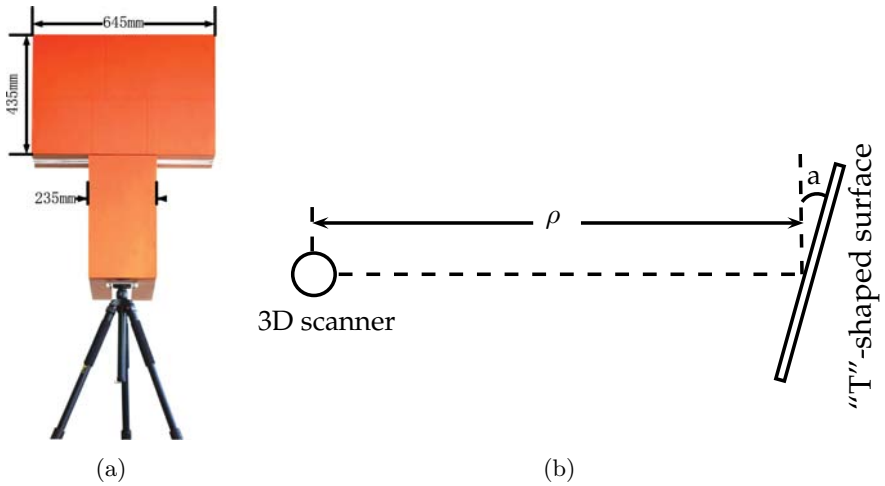


Figure 5.9: (a): the T-shaped surface constructed by two boxes mounted on a tripod, the scale of each edge has been labeled; (b): top view of the area benchmarking scenario, different ρ and a are used for evaluation.

5.5.3 Computational complexity, limitations and extendibility

Suppose we have an organized point cloud with size N which has been segmented into planar patches. Finding local triangles and rectangles in the image-like structure needs constant time, which has time complexity $O(N)$. Area computation for each local surface also needs constant time which has complexity $O(N)$. As a result, the algorithm has time complexity $O(N)$.

An inherent limitation of this method is the requirements of a 2.5D image-like structure, which cannot always be fulfilled by point clouds. But this is not an unsolvable problem for dense point clouds for which at least one perspective can view all surfaces without occlusion. To deal with those point clouds, only an additional down-sampling step is needed, i.e., a lower resolution range image should be constructed from the investigating scan using spherical coordinates. The down-sampling has a computational complexity of $O(N)$, and it will not induce a notable area difference if the resolution is selected properly. However, it is impossible to construct a range image for some point clouds, e.g., a sub-map which contains several aligned scans, or a point cloud which is gathered by a moving sensor.

Although the algorithm is proposed for coplanar point sets, it can be applied to calculate the area of arbitrary surfaces which have no self-occlusion. In other words, it has the ability to deal with more general segments than planar surfaces. The only difference between calculating the area of a planar surface and a non-planar surface is the normal vector in Eq. (5.8), where a local normal should be used instead of the global normal. It would be interesting to see whether the performance will be improved by integrating the area metric to the segment correspondences search procedure of Douillard et al. (2012).

5.6 Experiments and results

The algorithms have been implemented in C++ and all the experiments were carried out on a standard PC. For the Delaunay triangulation and alpha-shape, the implementations in CGAL have been used. The code has been published online, please refer to Appendix A for access to it. The presented approaches are benchmarked against each other with regard to accuracy and speed first, followed by typical results for different datasets. In order to benchmark the speed, the number of valid points is used as the size of each point cloud.

Since it is difficult to acquire the ground-truth for outdoor planar surfaces, an indoor scenario was used to benchmark the area calculation accuracy. Two boxes were employed to construct a "T"-shaped planar surface which is shown in Figure 5.9(a), where the scale of each edge has been labeled. Accordingly, the ground-truth area is 0.3828 m^2.

To evaluate the accuracy of the methods, the "T"-shaped surface was scanned by our custom-built aLRF (see Section 3.1) at different distances with three different orientations. The setup is illustrated in Figure 5.9(b). The relative error e was employed as the accuracy measure metric, which is calculated as

$$e = \frac{|s - \bar{s}|}{\bar{s}} \tag{5.9}$$

where s is the computed area and $\bar{s}$ is the ground-truth. Four methods have been employed to compute the area for comparison, namely the presented algorithms and number of square units. The relative errors of each method have been depicted in Figure 5.10. Clearly, the alpha-shape based method (mean error 8.91%) and ASUFRI (mean error 7.45%) have a better accuracy than the other two methods, while ASUFRI has the best performance.

Theoretically, the error was caused not only by data processing but also by the scanning mechanism. In the experiment, the pan resolution was set to 0.5 deg, which results in a gap approximately 78.5 mm between adjacent scan slices at 9 m. Therefore, the edge part of the surface could be missed in the scan, which makes the surface area smaller than the ground-truth. This error is inherent in the system and is the main reason why the relative error sometimes exceeds 10%. The relative error caused by the scanning mechanism, however, decreases with larger surfaces.

Considering that the Barcelona dataset contains enough point clouds for statistic analysis, it has been employed to benchmark the speed of area calculation. We only compared the speed of ASUFRI to the alpha-shape based method, since these two methods outperform their counterparts with respect to accuracy. The area calculation time as a function of the point cloud size has been drawn in Figure 5.11. ASUFRI (3.54 ms in average) is clearly significantly faster than the alpha-shape based method (1.02 s in average), because it does not require the projection from raw sensor data to the optimum fitted plane, transforming the projected points to 2D, and the triangulation related operations. Please also note the linear relation between the point cloud size and computation time for ASUFRI, which confirms the linear computational complexity of the algorithm from an experimental point of view. For all the planar surfaces segmented

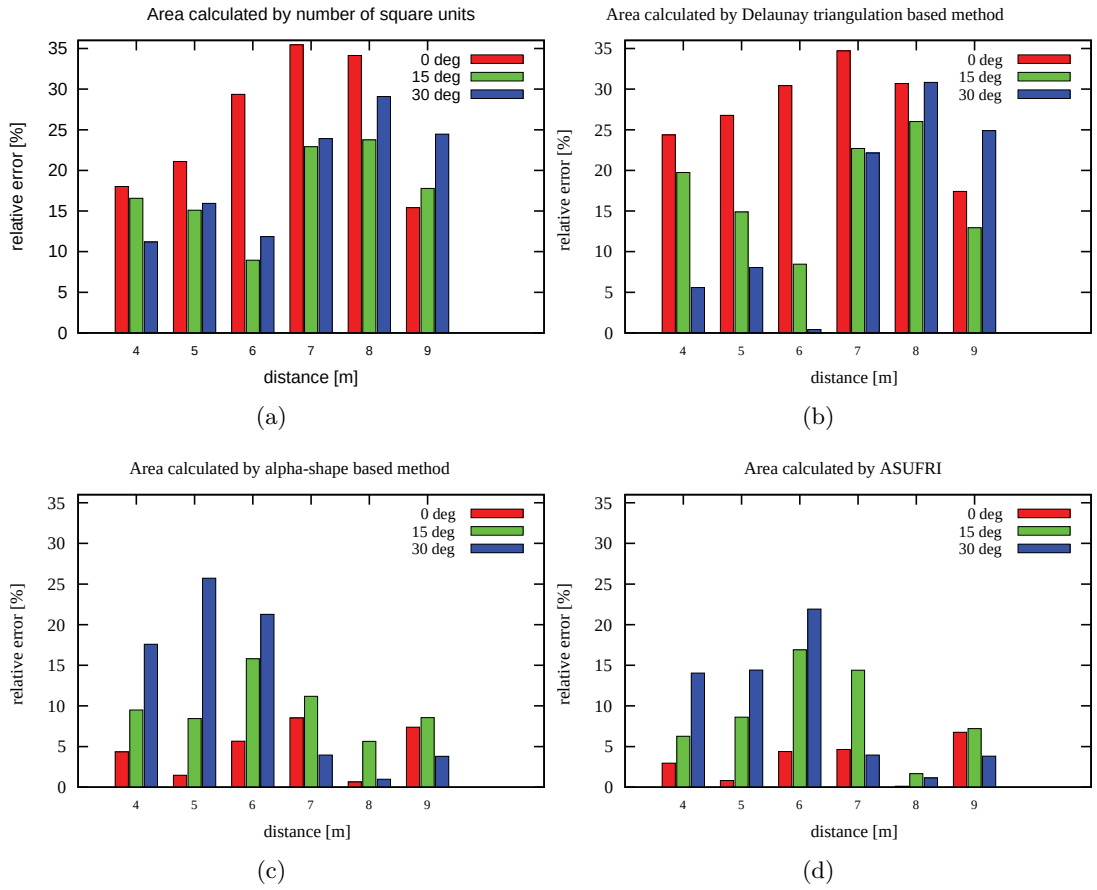


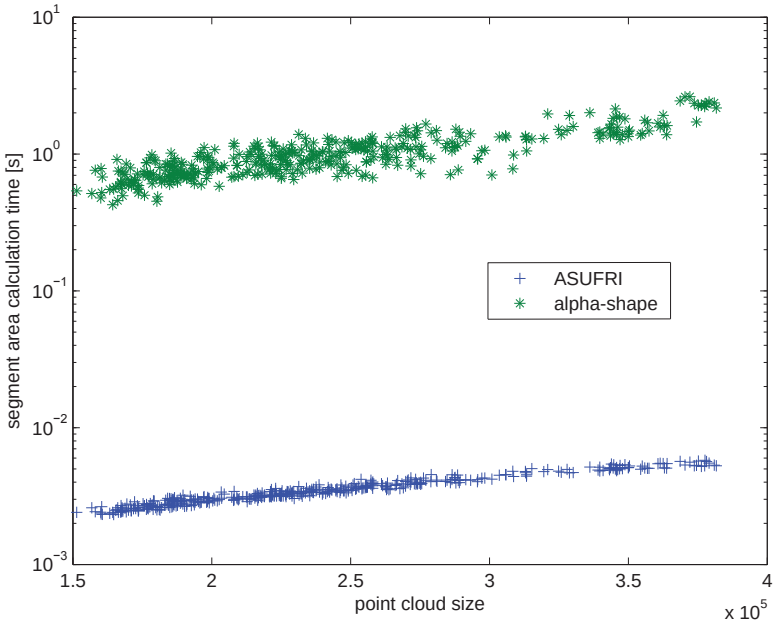
Figure 5.10: Illustrating the relative error compared to ground-truth for segment area calculation using four methods, x-axis represents different distances between the scanner and rotation axis of the surface, and three colors have been employed to denote orientations. (a): area computed by the method number of square units; (b): area computed by the Delaunay triangulation based method; (c): area computed by the alpha-shape based method; (d) area computed by ASUFRI.

from a point cloud with about $400,000$ points, less than 6 ms are needed for determining their area.

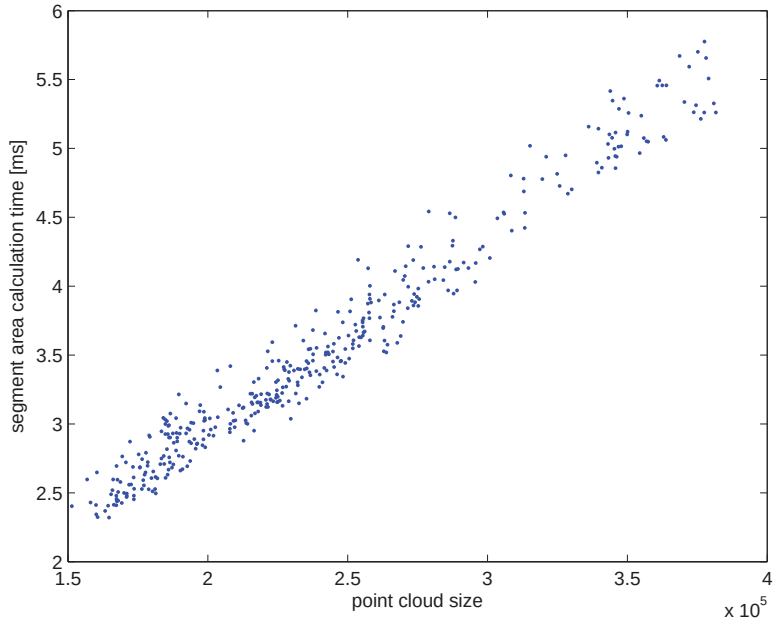
Benchmarking remark

The area calculation approaches have been benchmarked against each other, with regard to accuracy and speed. According to accuracy, the alpha-shape based method and ASUFRI have a better performance than their counterparts, while ASUFRI has the highest accuracy. Actually, the suitability of the area for segment association has also been verified through the accuracy benchmarking experiment, i.e., its area has a good repeatability when a surface is scanned at different distances from the sensor with different orientations. Regarding to speed, ASUFRI is approximately 100 times faster than the alpha-shape based method.

From the application point of view, the alpha-shape based method can deal with full

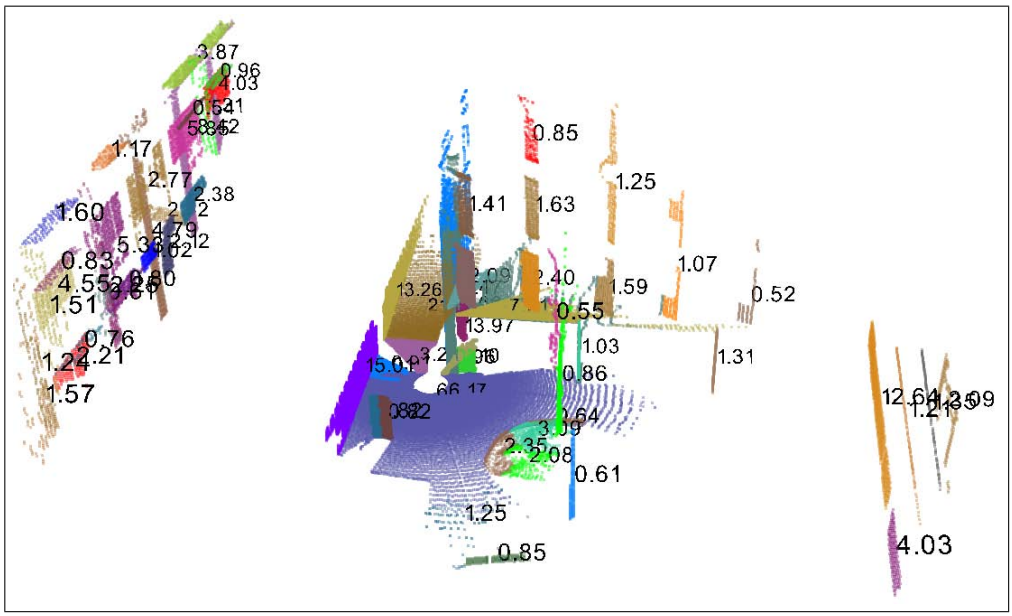


(a)

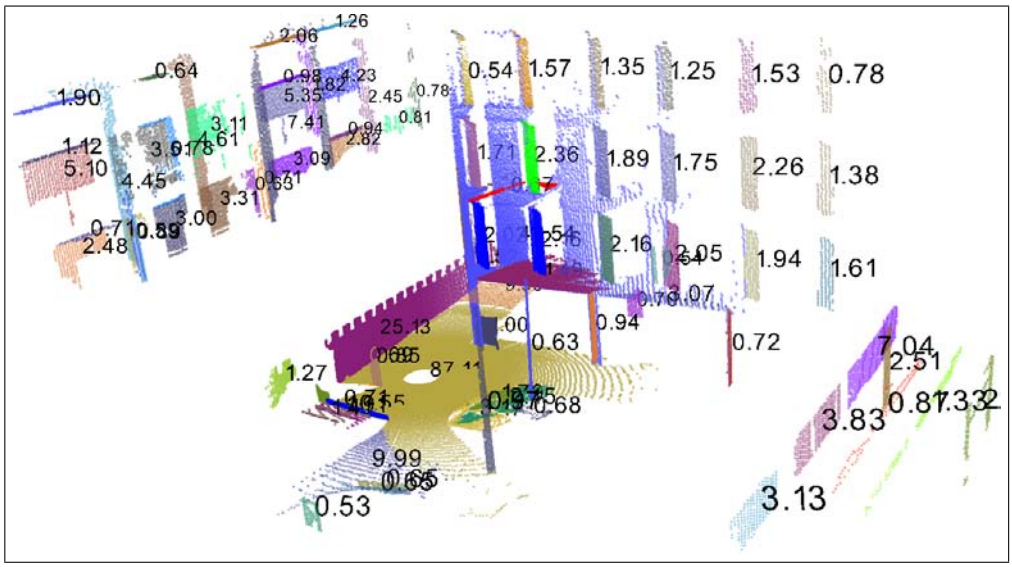


(b)

Figure 5.11: The area calculation time as a function of point cloud sizes. (a): comparison of ASUFRI and the alpha-shape based method, time in logarithmic scale. (b): area calculation time using ASUFRI in milliseconds. All 400 point clouds in the Barcelona dataset have been employed for benchmarking.



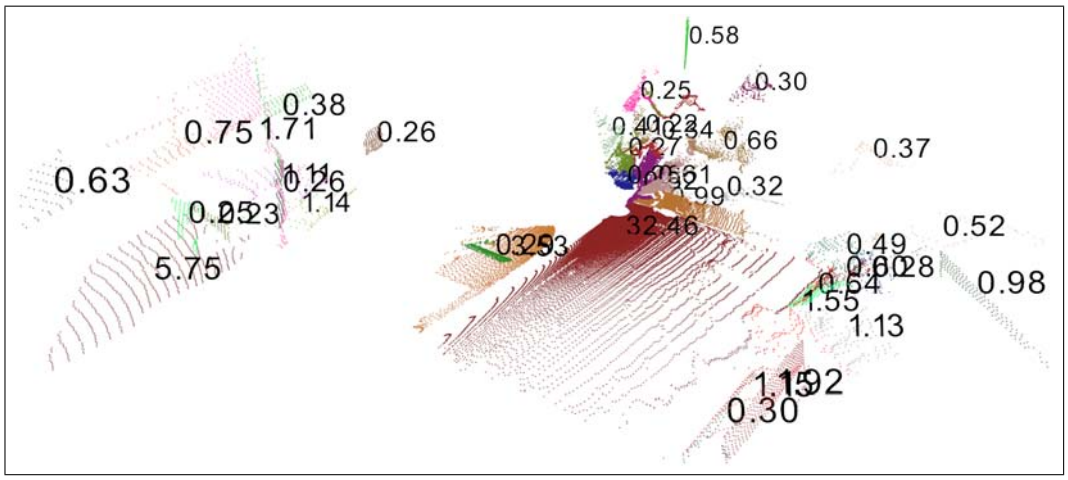
(a)



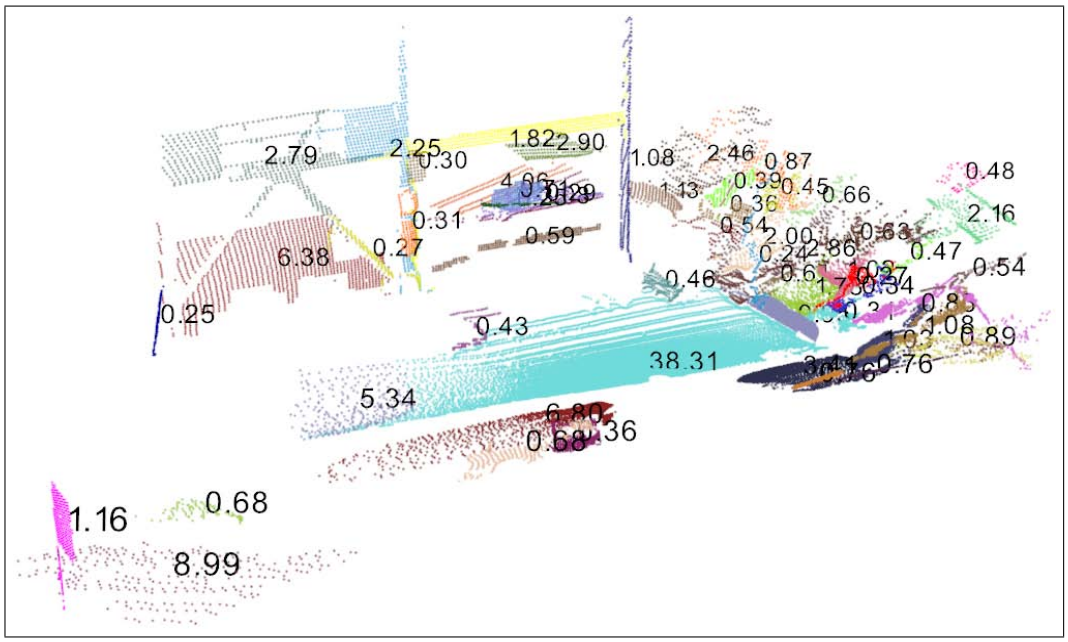
(b)

Figure 5.12: Depicting the area of each segment, wherein the cached octree region growing algorithm is employed for segmentation and the ASUFRI method is used for area calculation. (a) and (b): scan 2 and scan 3 in the Barcelona dataset. Segments whose area is smaller than 0.5 m^2 have been filtered out.

3D scans while ASUFRI can only be applied to organized scans. However, as already mentioned in Section 5.5, this does not hinder ASUFRI from dense point sets sampled at a fixed point of view. In conclusion, if the given point cloud is obtained with a single point of view, ASUFRI should be employed for segment area calculation; otherwise, the 2D alpha-shape based method is recommended.



(a)



(b)

Figure 5.13: Depicting the area of each segment, wherein the cached octree region growing algorithm is employed for segmentation and the ASUFRI method is used for area calculation. (a) and (b): scan 1 and scan 9 in the Texas dataset. Segments whose area is smaller than 0.2 m^2 have been filtered out.

Typical results

Due to space limitations, we cannot present all the planar segment area calculation results here. Instead, some typical results of different datasets have been selected for illustration, for further results please refer to Section A. For visualization, the segments have been colored randomly, and areas (in square meters) are positioned to the mass center of each corresponding planar patch. Viewpoints have been selected manually to show as many as segments. Segments whose area is smaller than a threshold have been

filtered out, please refer to each figure for the specific threshold. Figure 5.12 and Figure 5.13 detail four results from the ASUFRI method, two from the Barcelona dataset and another two from the Texas dataset.

5.7 Compared with point number as a similarity metric

The resulting area of this chapter is intended to be employed as a similarity metric between planar segments for correspondences determination. It would be interesting to see whether the area is a better metric than the number of points associated to each planar segment, which has been employed for search space pruning (Pathak et al., 2010b), EGI construction (Makadia et al., 2006), and overlapping ratio computation (Pathak et al., 2010c). Obviously, it will be gilding the lily if they are not distinct from each other, since the point number is a co-product of the segmentation process.

Intuitively, point number as a similarity metric is ill-defined for long-range sensors due to uneven sampling. For example, a fully-scanned surface will reflect back quite different numbers of beams at different distances or with different orientations, but its area will not change. Theoretically, the distinction between them can be measured by their covariance matrix.

All the four datasets have been employed to calculate the covariance matrix between the area and the point number of each segment. The area as a function of point number has been illustrated in Figure 5.14, wherein the cached octree region growing algorithm and ASUFRI have been employed for segmentation and area calculation, respectively. The Pearson product-moment correlation coefficient is chosen to measure the dependence between area and point number, i.e., the correlation coefficient ρ is calculated as

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{5.10}$$

where X and Y are two variables (area and point number here), $cov()$ means covariance and σ denotes standard deviation. Please see Figure 5.14 for the correlation coefficients between the area and point number for all segments in each dataset. It can be seen that the Barcelona dataset has the highest correlation coefficient (0.85), which can also be observed from the point number/area distributions in the figure. However, the correlation coefficients are relatively small in other datasets, which means the number of points associated to segments is not a proper similarity metric for segment correspondences search.

5.8 Summary

Three approaches for calculating the area of planar segments have been presented in this chapter, namely a 2D Delaunay triangulation based method in Section 5.3, a 2D alpha-shape based method in Section 5.4, a method termed ASUFRI in Section 5.5. The three approaches together with another related work have been benchmarked against each other with regards to accuracy and speed in Section 5.6, where ASUFRI and the alpha-shape based method are found to possess higher accuracy than their counterparts, and ASUFRI is much faster than the alpha-shape based method. From an application point

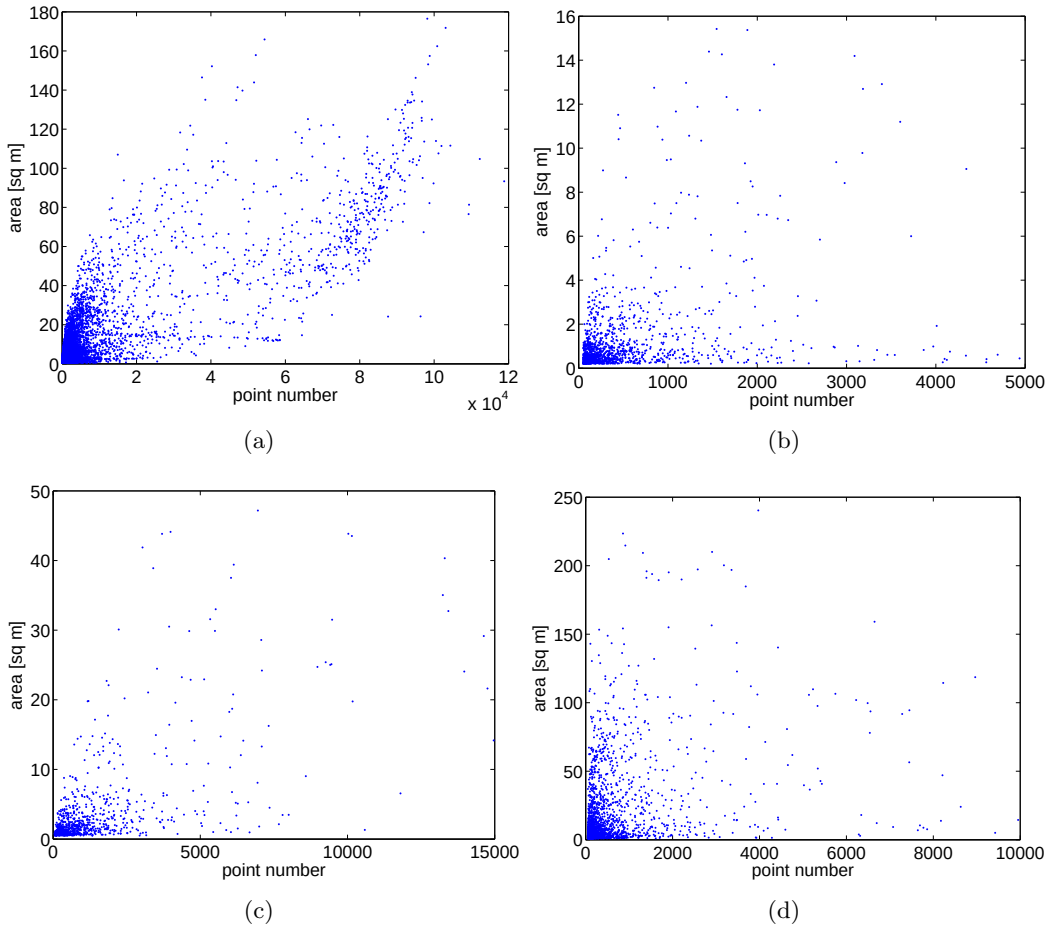


Figure 5.14: The area as a function of point number for all segments in each dataset. (a): for the Barcelona dataset, $\rho = 0.85$; (b): for the Texas dataset, $\rho = 0.37$; (c): for the Hamburg dataset, $\rho = 0.64$; (d): for the Bremen dataset, $\rho = 0.34$. ρ is the correlation coefficient between the area and point number for all segments in each dataset.

of view, ASUFRI can only be applied to organized point clouds, while the alpha-shape based method can deal with unorganized point sets. In Section 5.7, we also clarified that the area is a better similarity metric for segment correspondences search than the number of points associated to each segment.

Chapter 6

Planar segments based registration

> By three methods we may learn wisdom:
> First, by reflection, which is noblest; second,
> by imitation, which is easiest; and third by
> experience, which is the bitterest.
>
> Confucius

POINT cloud plane segmentation and planar segment area calculation have been introduced in Chapter 4 and Chapter 5, respectively. Armed with those techniques, a scan can be represented as a set of area attributed planar segments. This chapter focuses on pairwise registration based on these resulting segments. The main idea is to determine segment correspondences with a constrained search algorithm, which are then used to resolve the transformation. The correspondences are searched globally to maximize a spherical-correlation-like metric, wherein the search space is pruned by both self-similarity and interrelations (geometric constraints). The novelty of the approach is that only the area and plane parameters of each segment are employed, while no prior pose estimation from other sensors such as odometers and IMUs is required.

This chapter is organized as follows: The related work is introduced in Section 6.1, then we present the global search algorithm in Section 6.2. Afterwards, a closed-form transformation refine method is given in Section 6.3. The experiments are then drawn in Section 6.4, where the approach is compared to ICP and Minimum Uncertainty Maximum Consensus (MUMC), and benchmarked with regard to both accuracy and speed. Afterwards, it is extended to the domain of map merging in Section 6.5. In the end, Section 6.6 concludes the chapter.

6.1 Related work

3D scan registration has attracted increasing interest from the mobile robotic community during the past few years due to the rapid development of range sensors and their application in robotic systems. Most existing approaches need range sensor readings as well as a prior pose estimation which is usually obtained from odometer or IMU, such as ICP and 3D-NDT. In other words, local registration algorithms rely on a proper

initial pose guess; even with initial guess, they are susceptible to local minima. Please refer to Section 2.3 for a review on those methodologies.

Unfortunately, the prior pose estimation is not reliable in unstructured environments where the ground surface is complex, since the contact points between the robot and terrain are not stable and the robot may experience sideways sliding during movement. To deal with this problem, several odometry-free registration (also known as encoder-free registration or global registration) methodologies have been proposed. Since no prior pose estimation is employed in these approaches, the transformation is usually found by feature correspondences, where both global and local features can be utilized. As a geometric feature, planar segments have been explored for point cloud registration in recent years, which will be reviewed in this section. A short review on other global registration algorithms can be found in Section 2.3.

A comprehensive discussion on finding correspondences between two sets of planar or quadratic segments using attributed surface graphs can be found in Fischer and Kohlhepp (2000). In order to find corresponding surfaces, similarity metrics are formulated based on a number of attributes such as area ratio, shape factor, and curvature histogram. Based on these metrics, a set of Neuro-Fuzzy rules are designed which are employed for correspondences determination in a bounded tree search. When the correspondences are resolved, an evolutionary algorithm is used to refine the transformation. The approach has two main flaws: First, the Neuro-Fuzzy rules should be trained beforehand using supervised learning which needs a sufficient number of manually labeled true positives. Second, the evolutionary algorithm for transformation refine is time consuming.

In He et al. (2005), the correspondence between *complete plane patches* are determined by an interpretation tree wherein the search space is pruned by area, normal direction and centroids. However, this approach is valid only for complete planar surfaces due to the following reasons: First, the centroid is employed as a constraint which is variant for partially observed surfaces. Second, as is reported in the paper, only two correspondences are enough for transformation computation, again, this is only true when centroids are considered for transformation. Furthermore, even with the assumption a sufficient number of complete planar surfaces exist in each range image, their criterion for judging whether a surface is complete is only heuristic, which has not been proven either theoretically or experimentally with a number of datasets (only one pair of range image has been tested in the paper).

The orthogonal relation between planes in indoor environments has been intensively utilized in previous researches. In Kohlhepp et al. (2006), an orthogonal surface assignment (OSA) algorithm is proposed based on the building coordinate system (BCS). In order to establish a unique BCS for each range image, a so-called invariant direction (ground or ceiling) is tracked first. Another two axis alignments and the translation are then determined by maximizing the total overlap between corresponding features, where only surfaces parallel or orthogonal to the invariant direction are considered. An interesting sensor configuration has been employed in Nguyen et al. (2007), with two inclined laser mounted at 45° to the forward and backward direction. With the constraint moving on a planar surface, only those planes are perpendicular to the floor are considered for mapping. A similar approach was proposed in Harati and Siegwart (2007),

where the so-called right angle is utilized to determine plane correspondences. Nevertheless, the above algorithms are limited to structured indoor environments because the orthogonality between surfaces can not be fulfilled in more general environments.

In Viejo and Cazorla (2007), the rotation is found by a modified ICP algorithm over two normal vector sets obtained from the planes, with the correspondence criterion being angle between normal vectors. Each match in the correspondences is weighted by the angle between the two matched patches' normal vectors and the distance between the two planes. Afterwards, another modified ICP algorithm is employed to compute the translation by considering both the normal vector and the centroid of each planar segment. The drawback of this approach is that only planar movement has been considered and has not been evaluated with 6D pose registration.

To the best of our knowledge, using geometric constraints for plane correspondence determination was original presented in Rabbani and van den Heuvel (2005) for industrial environment reconstruction. As most industrial environments contain planes, cylinders, and spheres, these objects are segmented from each scan first. Then a model is fitted to each object using least squares. Afterwards, object correspondences are searched by geometric constraints, the constraints for planes are similar to that of Section 6.2. The main flaws of their work are:

1. Instead of picking up two nonparallel plane pairs, they randomly pick up one pair and search for the second, which is inefficient.
2. During correspondence search, the translation is computed using only one plane pair, wherein three corresponding pairs should be employed.
3. No global geometric consistent metric has been addressed, only one false positive in the correspondences will lead to incorrect pose registration.
4. Except plane parameters, no other similarity metrics have been employed, such as shape factor and area ratio.
5. The correspondences are determined with a naïve RANSAC implementation, where no heuristic information has been used.

A more recent approach is found in Pathak et al. (2010b). Besides plane parameters, the plane parameter uncertainties are calculated for each segment during weighted least squares fitting. Armed with this attribute, they proposed the MUMC algorithm, which is capable of determining the plane correspondences by maximizing the geometric consistency while minimizing the resulting uncertainty. In addition, MUMC has been proven to be faster and more robust than the ICP and 3D-NDT algorithm in plane-rich environments. Its accuracy compared to ground-truth and robustness with regard to occlusions has been evaluated in Pathak et al. (2010c) using a ground-truth enabled dataset. However, the requirement to fully determine a transformation has not been exactly worked out, and it is claimed that at least three nonparallel plane correspondences are needed for registration. As we show in Section 6.2, the sufficient condition is at least three plane correspondences with linearly independent normal vectors.

We will propose a novel registration approach in this chapter, which is close related to Makadia et al. (2006) (see Section 2.3) and Pathak et al. (2010b). It is inspired by the following ideas: First, if the area of segmented planar surfaces can be calculated, the spherical correlation can be accomplished by summing up the area multiplications

in each surface direction. Since the number of planar surfaces in a point cloud is much smaller than the number of regions in a fine resolution EGI, the computation cost of summing up the multiplications should be much lower than correlating two EGIs. Second, possible Euler angles can only appear in a subspace of SO(3); to exhaustively traverse a discretized grid of the whole SO(3) space as in Makadia et al. (2006) is not necessary. Instead, geometric constraints and other similarity metrics of planar segments can be used to prune the search space. The proposed algorithm is detailed in the next section, with the main differences from MUMC being as follows: First, only area and plane parameters of each segment are used for correspondence determination. Second, a different overall consistency metric that is similar to spherical correlation is presented.

6.2 Planar segment correspondences search

In this section, we present our point cloud registration approach which is based on area attributed planar patches. We emphasize again that no prior pose estimation from other sensors is needed; the transformation is searched globally which avoids falling into local extremal points.

The problem can be stated as follows: given two overlapping point clouds, one is the target scan Ψ_l which has been aligned to a common coordinate system, i.e., it has been fixed; the other one is the source scan Ψ_r which needed to be aligned to the coordinate system of the target scan, i.e., it is floating. Suppose they are sampled by a mobile robot at location O_l and O_r in the robot coordinate frames F_l and F_r respectively, the aim is to find the 3D rotation matrix ${}^l_r\mathbf{R}$ and 3D translation vector ${}^l_r\mathbf{t}$ which satisfy the relation

$$ {}^l\mathbf{p} = {}^l_r\mathbf{R}\,{}^r\mathbf{p} + {}^l_r\mathbf{t} \tag{6.1} $$

for a point observed in both point clouds with coordinates ${}^l\mathbf{p}$ and ${}^r\mathbf{p}$, respectively.

After plane segmentation and segment area calculation, each point cloud can be represented as an indexed planar patches using a set of triplets

$$ {}^k\mathcal{P} = \{{}^k\mathcal{P}_i\langle {}^k\hat{\mathbf{n}}_i, {}^k d_i, {}^k s_i\rangle, i = 1, 2, ...N_k\}. \tag{6.2} $$

As known, the spherical correlation of two overlapping spherical patterns will reach its maximum when they are rotationally aligned. The correlation is calculated by

$$ G(\mathbf{R}) = \int_0^{\pi}\int_0^{2\pi} H_l(\theta,\phi)\mathbf{R}H_r(\theta,\phi)d\phi d\theta, \tag{6.3} $$

where θ and ϕ denote the polar angle and azimuthal angle in spherical coordinates. $H_k(\theta,\phi), k = l, r$ is the pattern value at each point on the unit sphere surface. As mentioned, EGI has been employed as a spherical pattern for point cloud rotational alignment in Dold (2005) and Makadia et al. (2006). However, it is not applicable for long range sensors as it is a simple accumulation of surface normals. Because different parts of the point cloud have quite different densities, the count of surface normals becomes variant. For example, the laser rays sampling a planar surface will increase while the distance between the sensor and the surface decreases. On the other hand,

the area of a fully observed surface is an invariant feature which could be used as another spherical pattern. A comparison between the area and point number of a planar segment with regard to being a similarity metric has been presented in Section 5.7.

In planar surface abundant environments, the planar surfaces' area plays a dominant role in Eq. (6.3), so the correlation can be simplified to

$$G(\mathbf{R}) = \sum_{i=1}^{N_l} \sum_{j=1}^{N_r} {}^{l}s_i \, {}^{r}s_j \sigma({}^{l}\hat{\mathbf{n}}_i, \mathbf{R} \, {}^{r}\hat{\mathbf{n}}_j), \tag{6.4}$$

where

$$\sigma({}^{l}\hat{\mathbf{n}}_i, \mathbf{R} \, {}^{r}\hat{\mathbf{n}}_j) = \begin{cases} 1, & \text{if } {}^{l}\hat{\mathbf{n}}_i \cdot \mathbf{R} \, {}^{r}\hat{\mathbf{n}}_j \approx 1, \\ 0, & \text{otherwise.} \end{cases} \tag{6.5}$$

In other words, if two planar segments have the same surface normal after a rotation $\mathbf{R}$, their area multiplication will be added to the correlation. Unfortunately, this may lead to false correlations. From a geometrical point of view, i.e., the plane equation of each surface has two attributes, its normal direction and its orthogonal distance from the origin. Since only the normal direction is utilized in Eq. (6.4), false correlation resulted when two segments are from parallel surfaces, but not from the same surface, especially when the two segments are not observed in both point clouds.

To deal with this problem, we add the other geometric constraint into Eq. (6.4). As a result, Eq. (6.4) and (6.5) can be rewritten as:

$$G(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_l} \sum_{j=1}^{N_r} {}^{l}s_i \, {}^{r}s_j \sigma_{ij} \tag{6.6}$$

and

$$\sigma_{ij} = \begin{cases} 1, & \text{if } {}^{l}\mathcal{P}_i \leftrightarrow {}^{r}\mathcal{P}_j, \\ 0, & \text{otherwise,} \end{cases} \tag{6.7}$$

where ${}^{l}\mathcal{P}_i \leftrightarrow {}^{r}\mathcal{P}_j$ means ${}^{l}\mathcal{P}_i$ and ${}^{r}\mathcal{P}_j$ are from the same plane which is observed in both Ψ_l and Ψ_r, i.e., they have the same surface normal and distance from the origin. Consequently, both the rotation and translation are required in Eq. (6.6) and (6.7), since the rotation alone can only affect the surface normal.

Now the problem is reduced to determining correspondences between two sets of planar segments, which is solved by three steps: First, find all potential corresponding segment pairs with regard to self-similarity, without considering interrelations between segments, see Algorithm 6.1. Second, a breadth-first search for all potential solutions with regard to geometric constraints, see Algorithm 6.2. Third, a depth-first search for the optimum solution with regard to global consistency, see Algorithm 6.3.

Area-consistent pair enumeration

In order to make the determination both more efficient and robust, all planar segments with an area smaller than a preset threshold $\hbar$ are filtered out at the beginning in the first step (Algorithm 6.1, line 2). Then area ratio, as a self-similarity metric, is

Algorithm 6.1: Area-consistent segment pair enumeration.

Input: two indexed segment lists:
${}^{l}\mathcal{P} = \{{}^{l}\mathcal{P}_i\langle {}^{l}\hat{\mathbf{n}}_i, {}^{l}d_i, {}^{l}s_i\rangle, i = 1, 2, ...N_l\}$ from point cloud Ψ_l
${}^{r}\mathcal{P} = \{{}^{r}\mathcal{P}_i\langle {}^{r}\hat{\mathbf{n}}_i, {}^{r}d_i, {}^{r}s_i\rangle, i = 1, 2, ...N_r\}$ from point cloud Ψ_r
Output: a list of area-consistent segment pairs $\mathcal{L}$

1 $\mathcal{L} = \emptyset$;
2 remove all segments from ${}^{l}\mathcal{P}$ and ${}^{r}\mathcal{P}$ whose area is smaller than $\hbar$;
3 $N_l = \#({}^{l}\mathcal{P})$, $N_r = \#({}^{r}\mathcal{P})$;
4 **for** $i = 1 \rightarrow N_l$ **do**
5 **for** $j = 1 \rightarrow N_r$ **do**
6 **if** `areaConsistent` $({}^{l}\mathcal{P}_i, {}^{r}\mathcal{P}_j)$ **then**
7 `add` $(\mathcal{L}, \langle {}^{l}\mathcal{P}_i, {}^{r}\mathcal{P}_j\rangle)$;
8 **end**
9 **end**
10 **end**
11 **return** $\mathcal{L}$;

employed to find potential corresponding pairs which are termed *area-consistent* pairs. Note that other self-similarity metrics can also be used if available. ${}^{l}\mathcal{P}_i$ and ${}^{r}\mathcal{P}_j$ are marked as an area-consistent pair if

$$\frac{|\,{}^{l}s_i - {}^{r}s_j|}{\max({}^{l}s_i, {}^{r}s_j)} < \lambda, \tag{6.8}$$

where λ is a positive number smaller than 1 (Algorithm 6.1, line 6). λ is an important parameter: On the one hand, with a large value, the algorithm is more robust to occlusions and partial observations, at the price of enumerating more potential correspondences. On the other hand, with a small value, the algorithm is faster but may not find the correct correspondences. In the end, all area-consistent pairs are put into a list (Algorithm 6.1, line 7)

$$\mathcal{L} = \{\mathcal{L}_i\langle {}^{l}\mathcal{P}_{i1}, {}^{r}\mathcal{P}_{i2}\rangle, i = 1...N_L, {}^{l}\mathcal{P}_{i1} \in {}^{l}\mathcal{P}, {}^{r}\mathcal{P}_{i2} \in {}^{r}\mathcal{P}\}. \tag{6.9}$$

Breadth-first potential solutions search

False hypotheses are possibly present in $\mathcal{L}$, which will be eliminated in the following steps using interrelations. As is known, the rotation matrix can be computed by two nonparallel corresponding planes. Unlike rotation, three planes with linearly independent normal vectors are needed to determine the translation. In other words, the minimum set of correspondences to determine a unique transformation is three, wherein the planes have linearly independent normals. This statement is more exact than three nonparallel planes (see MUMC), because the translation along the line direction cannot be determined when the three nonparallel planes are parallel to a line. Following this idea, we try to find all possible triplets, i.e., all potential solutions, using a breadth-first search which is illustrated in Algorithm 6.2.

Algorithm 6.2: Breadth-first potential solutions search

Input: a list of area-consistent segment pairs $\mathcal{L}$
Output: a list of nonparallel segment triplets $\mathcal{T}$

$\mathcal{H} = \emptyset$, $\mathcal{T} = \emptyset$;
for $i = 1 \rightarrow N_L$ **do**
  for $j = i + 1 \rightarrow N_L$ **do**
    if `nonparallel` $(\mathcal{L}_i, \mathcal{L}_j) ==$ false **then**
      continue;
    end
    $\mathbf{R}$ = `rotation` $(\mathcal{L}_i, \mathcal{L}_j)$;
    if `locomotionReachable` $(\mathbf{R}, \Theta) =$ false **then**
      continue;
    end
    for $k = 1 \rightarrow N_L, k \neq i, k \neq j$ **do**
      key = `hash` (i, j, k) ;
      if `find` $(\mathcal{H}, key) =$ true **then**
        continue;
      end
      `add` $(\mathcal{H}, key)$;
      if `consistent` $(\mathbf{R}, \mathcal{L}_k) =$ false **then**
        continue;
      end
      if `nonparallel` $(\mathcal{L}_i, \mathcal{L}_j, \mathcal{L}_k) =$ false **then**
        continue;
      end
      if `normalsOnSamePlane` $(\mathcal{L}_i, \mathcal{L}_j, \mathcal{L}_k) =$ true **then**
        continue;
      end
      $\mathbf{t}$ = `translation` $(\mathcal{L}_i, \mathcal{L}_j, \mathcal{L}_k)$;
      if `locomotionReachable` $(\mathbf{t}, \Theta) =$ false **then**
        continue;
      end
      if `overlapping` $(\mathcal{L}_i, \mathcal{L}_j, \mathcal{L}_k) =$ false **then**
        continue;
      end
      `add` $(\mathcal{T}, \langle \mathcal{L}_i, \mathcal{L}_j, \mathcal{L}_k, \mathbf{R}, \mathbf{t} \rangle)$;
    end
  end
end
if $\mathcal{T} \neq \emptyset$ **then**
  return $\mathcal{T}$;
end
else
  return false;
end

Two area-consistent pairs are treated as nonparallel (Algorithm 6.2, line 4) if they fulfill the following condition

$$\begin{cases} |{}^{l}\hat{\mathbf{n}}_{j1} \cdot {}^{l}\hat{\mathbf{n}}_{i1}| < \tau, \\ |{}^{r}\hat{\mathbf{n}}_{j1} \cdot {}^{r}\hat{\mathbf{n}}_{i1}| < \tau, \\ |\arccos({}^{l}\hat{\mathbf{n}}_{j1} \cdot {}^{l}\hat{\mathbf{n}}_{i1}) - \arccos({}^{r}\hat{\mathbf{n}}_{j1} \cdot {}^{r}\hat{\mathbf{n}}_{i1})| < \mu, \end{cases} \tag{6.10}$$

where τ is a positive number to guarantee there is a notable angle between the two investigated planes, μ is a threshold to make sure that two planes in the map cloud have a similar angle as the other two planes in the data cloud. Then a rotation matrix is calculated by these two planes (Algorithm 6.2, line 7). The rotation is decomposed into two axis-angle rotations and computed as:

$$\begin{cases} \langle axis_1, angle_1 \rangle = \langle \frac{({}^{l}\hat{\mathbf{n}}_{i1} \times {}^{l}\hat{\mathbf{n}}_{j1}) \times ({}^{r}\hat{\mathbf{n}}_{i2} \times {}^{r}\hat{\mathbf{n}}_{j2})}{|({}^{l}\hat{\mathbf{n}}_{i1} \times {}^{l}\hat{\mathbf{n}}_{j1}) \times ({}^{r}\hat{\mathbf{n}}_{i2} \times {}^{r}\hat{\mathbf{n}}_{j2})|}, \arccos(\frac{{}^{l}\hat{\mathbf{n}}_{i1} \times {}^{l}\hat{\mathbf{n}}_{j1}}{|{}^{l}\hat{\mathbf{n}}_{i1} \times {}^{l}\hat{\mathbf{n}}_{j1}|}, \frac{{}^{r}\hat{\mathbf{n}}_{i2} \times {}^{r}\hat{\mathbf{n}}_{j2}}{|{}^{r}\hat{\mathbf{n}}_{i2} \times {}^{r}\hat{\mathbf{n}}_{j2}|}) \rangle, \\ \mathbf{R}_1 = \langle axis_1, angle_1 \rangle.\text{toRotationMatrix}(), \\ \langle axis_2, angle_2 \rangle = \langle \frac{({}^{l}\hat{\mathbf{n}}_{i1} \times {}^{l}\hat{\mathbf{n}}_{j1})}{|({}^{l}\hat{\mathbf{n}}_{i1} \times {}^{l}\hat{\mathbf{n}}_{j1})|}, \frac{1}{2}(\arccos(\mathbf{R}_1 \, {}^{r}\hat{\mathbf{n}}_{i2}, {}^{l}\hat{\mathbf{n}}_{i1}) + \arccos(\mathbf{R}_1 \, {}^{r}\hat{\mathbf{n}}_{j2}, {}^{l}\hat{\mathbf{n}}_{j1}) \rangle, \\ \mathbf{R}_2 = \langle axis_2, angle_2 \rangle.\text{toRotationMatrix}(), \\ \mathbf{R} = \mathbf{R}_2 \mathbf{R}_1. \end{cases} \tag{6.11}$$

Afterwards, the computed rotation matrix is evaluated by the robot's kinematic model Θ in order to restrict the rotation angle to being realistic (Algorithm 6.2, line 8). For this purpose, the rotation is decomposed into two successive rotations $\langle \hat{\mathbf{h}}, \alpha \rangle$ and $\langle \hat{\mathbf{z}}, \beta \rangle$, where $\hat{\mathbf{h}}.z = 0$, and $\hat{\mathbf{z}}$ points along the z-axis. According to the locomotion ability of most ground robots, α should be smaller than an angle threshold χ and there is no limitation for β. Therefore, the β angle can be ignored and α can be simply computed by

$$\alpha = \arccos(\mathbf{R}\hat{\mathbf{z}} \cdot \hat{\mathbf{z}}). \tag{6.12}$$

If the rotation is realistic, we try to find another area-consistent pair under geometric constraints. Since we are dealing with a combination but not a permutation problem, a hash table is constructed to guarantee that only unique triplets are investigated (Algorithm 6.2, line 12–15).

If the triplet has not been investigated, $\mathcal{L}_i, \mathcal{L}_j$ and $\mathcal{L}_k$ are considered to be *normal-consistent* iff:

1. $\mathcal{L}_k$ agrees with this rotation, i.e.,

$$ {}^{l}\hat{\mathbf{n}}_{k1} \cdot (\mathbf{R} \, {}^{r}\hat{\mathbf{n}}_{k2}) > \xi, \tag{6.13}$$

   where ξ is a scalar threshold which is approximately 1.

2. $\mathcal{L}_k$ is nonparallel to both $\mathcal{L}_i$ and $\mathcal{L}_j$, see Eq. (6.10).

3. The normal of $\mathcal{L}_k$ is linearly independent of that of $\mathcal{L}_i$ and $\mathcal{L}_j$. In other words, the normals of $\mathcal{L}_i$, $\mathcal{L}_j$ and $\mathcal{L}_k$ should not be coplanar (when they are fixed to the origin). This can be tested by

$$\begin{cases} {}^{l}\hat{\mathbf{n}}_{i\times j} = \frac{{}^{l}\hat{\mathbf{n}}_{i1}\times {}^{l}\hat{\mathbf{n}}_{j1}}{\|{}^{l}\hat{\mathbf{n}}_{i1}\times {}^{l}\hat{\mathbf{n}}_{j1}\|}, \\ ds = \mathbf{0}\cdot {}^{l}\hat{\mathbf{n}}_{k1} - {}^{l}d_{k1}, \\ de = {}^{l}\hat{\mathbf{n}}_{i\times j}\cdot {}^{l}\hat{\mathbf{n}}_{k1} - {}^{l}d_{k1}, \\ \mathbf{ps} = \mathbf{0} - ds * {}^{l}\hat{\mathbf{n}}_{k1}, \\ \mathbf{pe} = {}^{l}\hat{\mathbf{n}}_{i\times j} - de * {}^{l}\hat{\mathbf{n}}_{k1}, \\ |\mathbf{pe}-\mathbf{ps}| < \tau, \end{cases} \tag{6.14}$$

where ${}^{l}\hat{\mathbf{n}}_{i\times j}$ is a unit vector parallel to the intersection line of ${}^{l}\mathcal{P}_i$ and ${}^{l}\mathcal{P}_j$; $|\mathbf{pe}-\mathbf{ps}|$ is the projection of ${}^{l}\hat{\mathbf{n}}_{i\times j}$ on ${}^{l}\mathcal{P}_k$; τ is to ensure the three normals are notably non-coplanar.

Note that only surface normals have been used in above tests. If $\mathcal{L}_k$ passes all the tests, a translation $\mathbf{t}$ is calculated using the three plane pairs in Eq. (6.15) (Algorithm 6.2, line 26):

$$\mathbf{t} = \mathbf{N}^{-1}\mathbf{D}, \tag{6.15}$$

where

$$\mathbf{N} = \begin{bmatrix} {}^{l}\hat{\mathbf{n}}_{i1}^{T} \\ {}^{l}\hat{\mathbf{n}}_{j1}^{T} \\ {}^{l}\hat{\mathbf{n}}_{k1}^{T} \end{bmatrix}, \mathbf{D} = \begin{bmatrix} {}^{l}d_{i1} - {}^{r}d_{i2} \\ {}^{l}d_{j1} - {}^{r}d_{j2} \\ {}^{l}d_{k1} - {}^{r}d_{k2} \end{bmatrix}. \tag{6.16}$$

The inverse of $\mathbf{N}$ always exists because the normals are linearly independent.

The translation is then also assessed using the robot's kinematic model (Algorithm 6.2, line 27–29), i.e., the angle

$$\alpha = \arccos\left(\frac{\mathbf{t}.z}{\sqrt{(\mathbf{t}.x)^2 + (\mathbf{t}.y)^2}}\right) \tag{6.17}$$

should be smaller than χ. If $\mathbf{t}$ passes the test, a last check upon overlapping is executed based on $\mathbf{R}$, $\mathbf{t}$ and area of each segment. Two segments in an area-consistent pair $\mathcal{L}_x$ overlap if they fulfill

$$(\mathbf{R}\,{}^{r}\mathbf{m}_{x2} + \mathbf{t} - {}^{l}\mathbf{m}_{x1})^2 < \min\left(\frac{{}^{r}s_{x2}}{\pi}, \frac{{}^{l}s_{x1}}{\pi}\right). \tag{6.18}$$

In other words, we approximate each segment as a disc, and mark two discs as overlapped if the distance between their centers is less than the smaller radius. If the triplet passes the overlapping test, it is added to a list of potential solutions $\mathcal{T}$.

The checks against the robot's kinematics are not essential, they are employed to restrict the search space if a robot kinematics model is available. The search procedure terminates when no new triplets can be found (Algorithm 6.2, line 2–36). If no triplet can pass all tests, the registration is failed; otherwise, a depth-first search algorithm is employed to find the optimum solution from $\mathcal{T}$, which is explained in the next section.

Algorithm 6.3: Depth-first optimum solution search

Input: a list of area-consistent segment pairs $\mathcal{L}$
a list of nonparallel segment triplets $\mathcal{T}$
Output: determined correspondences $\mathcal{K}$, rotation ${}^{l}_{r}\mathbf{R}$ and translation ${}^{l}_{r}\mathbf{t}$

1 $\mathcal{K} = \emptyset$, $max\_c = 0$, $max\_t = 0$;
2 **for** $i = 1 \rightarrow N_T$ **do**
3 $\quad \mathcal{K} =$ `consensus` $(\mathcal{T}_i, \mathcal{L})$;
4 $\quad \mathcal{K} =$ `unique` $(\mathcal{K})$;
5 $\quad c =$`correlation` $(\mathcal{K})$;
6 $\quad$ **if** $c > max\_c$ && $\#(\mathcal{K}) \geq max\_n$ **then**
7 $\quad\quad {}^{l}_{r}\mathbf{R} = \mathbf{R}, {}^{l}_{r}\mathbf{t} = \mathbf{t}, max\_c = c, max\_n = \#(\mathcal{K})$;
8 $\quad$ **end**
9 **end**
10 **if** $\#(\mathcal{K}) \geq \kappa$ **then**
11 $\quad$ **return** $\mathcal{K}, {}^{l}_{r}\mathbf{R}, and\, {}^{l}_{r}\mathbf{t}$;
12 **end**
13 **else**
14 $\quad$ **return** false;
15 **end**

Depth-first optimum solution search

Usually there are multiple potential solutions in $\mathcal{T}$, wherein the optimum solution is distinguished with regard to the spherical-correlation-like criterion in Eq. (6.6) using depth-first search as illustrated in Algorithm 6.3.

Explanation of line 3: For a given potential solution $\mathcal{T}_i = \langle \mathcal{L}_r, \mathcal{L}_s, \mathcal{L}_t, \mathbf{R}, \mathbf{t} \rangle$, other area-consistent pairs which are consistent with $\langle \mathbf{R}, \mathbf{t} \rangle$ can be found by geometric consistency assessments. This test should be conducted on all other items in $\mathcal{L}$ except the three fixed pairs. Suppose $\mathcal{L}_c$ is investigated, it agrees with $\langle \mathbf{R}, \mathbf{t} \rangle$ if: First,

$$\begin{cases} {}^{l}\hat{\mathbf{n}}_{c1} \cdot (\mathbf{R}\, {}^{l}\hat{\mathbf{n}}_{c2}) > \xi, \\ {}^{l}\hat{\mathbf{n}}_{c1} \cdot \mathbf{t} + {}^{l}d_{c2} - {}^{l}d_{c1} < \eta, \end{cases} \tag{6.19}$$

where η is a scalar threshold. Second, the two segments are overlapped after the transformation, see Eq. (6.18).

Explanation of line 4: Unfortunately, the set $\mathcal{K}$ may still contain non-unique correspondences, i.e., some segment ${}^{l}\mathcal{P}$ may be mapped to more than one plane in ${}^{r}\mathcal{P}$ and vice versa. The uniqueness problem is solved by sorting all $\langle {}^{l}\mathcal{P}_{x1}\, {}^{r}\mathcal{P}_{x2} \rangle \in \mathcal{T}_i$ in increasing order of distance between their centers (after transformation); then the correspondences are fixed by traversing the sorted list. If ${}^{l}\mathcal{P}_{x1}$ can be matched with multiple segments in ${}^{r}\mathcal{P}$, the pairing with smaller distance is automatically selected. Similar reasoning applies if ${}^{r}\mathcal{P}_{x2}$ can be matched with more than one segment in ${}^{l}\mathcal{P}$. The reduced and geometric consistent unique-mapping set is the consensus set of $\mathcal{T}_i$.

Thereafter, the correlation in Eq. (6.6) will be computed for the correspondences found in $\mathcal{K}$. If the correlation value is greater than the current maximum correlation max_c and the size of the consensus set is greater than or equal to the current maximum

max_n, the maximum correlation and maximum consensus set size will be replaced with the new value, and the transformation is also set (Algorithm 6.3, line 6–8). This procedure will continue until all potential solutions have been explored (Algorithm 6.3, line 2–9). In the end, if the size of the consensus correspondence set is greater than a pre-defined threshold κ, the found consensus set and transformation will be returned. Otherwise, the registration has failed (Algorithm 6.3, line 10–15).

Computational complexity analysis

Suppose there are N_l and N_r segments in the target scan and source scan respectively. The cost for area-consistent pair enumeration is $O(N_l N_r)$, which results in a list of size N_L. The number of branches needed to traverse in the breadth-first search is $\binom{N_L}{3}$, which has a time complexity of $O(N_L^3)$. Suppose N_T potential solutions are found in the breadth-first search; then in the depth-first search, traversing from the root to each leaf has a time complexity of $O(N_L)$. Hence the total time complexity of the depth-first search is $O(N_T N_L)$. In the worst case, we have

$$\begin{cases} N_L = N_l N_r, \\ N_T = \frac{N_L(N_L-1)(N_L-2)}{6} \simeq N_L^3. \end{cases} \tag{6.20}$$

As a result, the worst-case time complexity for correspondences search is $O(N_l^4 N_r^4)$. Clearly, it highly depends on the value of N_L. Actually, N_L reaches its upper bound only if all the planar segments have a similar area, which is unlikely to happen in real-world environments. Please see Table 6.2 for evidence, where it can also be seen that the time complexity is closely related to N_L. To further decrease the cost, other self-similarity metrics such as shape factor and plane parameter uncertainty can be used.

6.3 Registration result refinement

The rotation and translation for correspondences search in Section 6.2 are computed heuristically by only two and three nonparallel planar segments. As a result, they need to be refined using all determined correspondences to improve the registration accuracy. We present a closed-form solution in this section. For a planar segment ${}^r\mathcal{P}_j$ which corresponds to ${}^l\mathcal{P}_i$, the following function should be minimized when they are aligned properly by $\mathbf{R}$ and $\mathbf{t}$

$$E_{i\leftrightarrow j} = \sum_{k=1}^{N} ((\mathbf{R}\,{}^r\mathbf{p}_k + \mathbf{t}) \cdot {}^l\hat{\mathbf{n}}_i + {}^l d_i)^2, \tag{6.21}$$

where ${}^r\mathbf{p}_k, k = 1, 2, \cdots N$ denote the points associated to ${}^r\mathcal{P}_j$, ${}^l\hat{\mathbf{n}}_i$ and ${}^l d_i$ denote the plane parameters of ${}^l\mathcal{P}_i$. In other words, the points associated to ${}^r\mathcal{P}_j$ should be optimally fitted to the plane of ${}^l\mathcal{P}_i$. Considering all determined correspondences, the

following scaler should be minimized

$$E = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} ((\mathbf{R}\,{}^{r}\mathbf{p}_k + \mathbf{t}) \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + {}^{l}d_{c,l})^2, \tag{6.22}$$

where N_c is the number of determined correspondences, c, l and c, r are used to denote the two segments' indices in the cth correspondence, and $N_{c,r}$ is the number of points associated to ${}^{r}\mathcal{P}_{c,r}$.

Our method is based on the following observation: although the rotation has been computed by only two nonparallel planar segments, see Eq. (6.11), it is still accurate enough for providing a good initial value which is close to the refining result. In other words, the incremental rotations are small enough to be linearized. For example, in the case of rotation around x-axis:

$$\mathbf{R}_{x,\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\alpha \\ 0 & \alpha & 1 \end{pmatrix}. \tag{6.23}$$

As a result, the full rotation may be approximated as

$$\mathbf{R} \approx \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix} \equiv \mathbf{I}_3 + \triangle\mathbf{R} \tag{6.24}$$

for rotation angle α, β, and γ around the x, y, and z axis, where $\mathbf{I}_3$ is the 3×3 identity matrix and

$$\triangle\mathbf{R} = \begin{pmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{pmatrix}. \tag{6.25}$$

Substituting Eq. (6.24) into (6.22), we obtain

$$\begin{aligned} E &= \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} ((\mathbf{I}_3\,{}^{r}\mathbf{p}_k + \triangle\mathbf{R}\,{}^{r}\mathbf{p}_k + \mathbf{t}) \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + {}^{l}d_{c,l})^2 \\ &= \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} (\mathbf{I}_3\,{}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \triangle\mathbf{R}\,{}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + {}^{l}d_{c,l})^2. \end{aligned} \tag{6.26}$$

From the dot and cross product definition,

$$\begin{aligned} \triangle\mathbf{R}\,{}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} &= \mathbf{r} \times {}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} \\ &= \mathbf{g}_{ij} \cdot \mathbf{r}, \end{aligned} \tag{6.27}$$

where

$$\mathbf{r} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{6.28}$$

and

$$\mathbf{g}_{ij} = {}^{r}\mathbf{p}_k \times {}^{l}\hat{\mathbf{n}}_{c,l}. \tag{6.29}$$

Substituting Eq. (6.27) into (6.26), the objective function can be rewritten as

$$E = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} ({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{g}_{ij} \cdot \mathbf{r} + d)^2. \tag{6.30}$$

We now minimize E with respect to α, β, γ, t_x, t_y, and t_z by setting the partial derivatives to zero:

$$\begin{cases} \frac{\partial E}{\partial \alpha} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} 2g_{ij,x}({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot \hat{\mathbf{n}}_i + \mathbf{g}_{ij} \cdot \mathbf{r} + d) = 0, \\ \frac{\partial E}{\partial \beta} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} 2g_{ij,y}({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot \hat{\mathbf{n}}_i + \mathbf{g}_{ij} \cdot \mathbf{r} + d) = 0, \\ \frac{\partial E}{\partial \gamma} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} 2g_{ij,z}({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot \hat{\mathbf{n}}_i + \mathbf{g}_{ij} \cdot \mathbf{r} + d) = 0, \\ \frac{\partial E}{\partial t_x} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} 2n_{i,x}({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot \hat{\mathbf{n}}_i + \mathbf{g}_{ij} \cdot \mathbf{r} + d) = 0, \\ \frac{\partial E}{\partial t_y} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} 2n_{i,y}({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot \hat{\mathbf{n}}_i + \mathbf{g}_{ij} \cdot \mathbf{r} + d) = 0, \\ \frac{\partial E}{\partial t_z} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} 2n_{i,z}({}^{r}\mathbf{p}_k \cdot {}^{l}\hat{\mathbf{n}}_{c,l} + \mathbf{t} \cdot \hat{\mathbf{n}}_i + \mathbf{g}_{ij} \cdot \mathbf{r} + d) = 0. \end{cases} \tag{6.31}$$

These equations can be collected and written in matrix form

$$\mathbf{M}\mathbf{x} = \mathbf{y}, \tag{6.32}$$

where

$$\mathbf{M} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} \begin{pmatrix} g_{ij,x}g_{ij,x} & g_{ij,x}g_{ij,y} & g_{ij,x}g_{ij,z} & g_{ij,x}n_{i,x} & g_{ij,x}n_{i,y} & g_{ij,x}n_{i,z} \\ g_{ij,y}g_{ij,x} & g_{ij,y}g_{ij,y} & g_{ij,y}g_{ij,z} & g_{ij,y}n_{i,x} & g_{ij,y}n_{i,y} & g_{ij,y}n_{i,z} \\ g_{ij,z}g_{ij,x} & g_{ij,z}g_{ij,y} & g_{ij,z}g_{ij,z} & g_{ij,z}n_{i,x} & g_{ij,z}n_{i,y} & g_{ij,z}n_{i,z} \\ n_{i,x}g_{ij,x} & n_{i,x}g_{ij,y} & n_{i,x}g_{ij,z} & n_{i,x}n_{i,x} & n_{i,x}n_{i,y} & n_{i,x}n_{i,z} \\ n_{i,y}g_{ij,x} & n_{i,y}g_{ij,y} & n_{i,y}g_{ij,z} & n_{i,y}n_{i,x} & n_{i,y}n_{i,y} & n_{i,y}n_{i,z} \\ n_{i,z}g_{ij,x} & n_{i,z}g_{ij,y} & n_{i,z}g_{ij,z} & n_{i,z}n_{i,x} & n_{i,z}n_{i,y} & n_{i,z}n_{i,z} \end{pmatrix}, \tag{6.33}$$

$$\mathbf{x} = \begin{pmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{pmatrix}^{\mathrm{T}}, \tag{6.34}$$

Figure 6.1: The area where the first 20 scans in the Barcelona dataset were obtained ("FIB plaza" of the UPC Nord Campus in Barcelona, adopted from Nokia Maps.)

and

$$\mathbf{y} = \sum_{c=1}^{N_c} \sum_{k=1}^{N_{c,r}} \begin{pmatrix} g_{ij,x}({}^r\mathbf{p}_k \cdot {}^l\hat{\mathbf{n}}_{c,l} + d) \\ g_{ij,y}({}^r\mathbf{p}_k \cdot {}^l\hat{\mathbf{n}}_{c,l} + d) \\ g_{ij,z}({}^r\mathbf{p}_k \cdot {}^l\hat{\mathbf{n}}_{c,l} + d) \\ n_{i,x}({}^r\mathbf{p}_k \cdot {}^l\hat{\mathbf{n}}_{c,l} + d) \\ n_{i,y}({}^r\mathbf{p}_k \cdot {}^l\hat{\mathbf{n}}_{c,l} + d) \\ n_{i,z}({}^r\mathbf{p}_k \cdot {}^l\hat{\mathbf{n}}_{c,l} + d) \end{pmatrix}. \tag{6.35}$$

This is a linear matrix equation which can be solved by standard methods. We choose to use Cholesky decomposition because $\mathbf{M}$ is symmetric.

6.4 Experiments and results

The algorithm has been implemented in C++ and all the experiments were carried out on a standard PC. The code has been published online, please refer to Appendix A for the access to it. The Eigen library (Guennebaud et al., 2010) has been employed for linear algebra. PCL (Rusu and Cousins, 2011) has been utilized for 3D visualization. Due to space limitation, we cannot present all the pairwise registration results. Instead, one typical result in each dataset has been selected for illustration.

All the four datasets have been employed to evaluate the proposed approach: The Texas dataset was acquired using a relatively narrow FoV scanner compared to the other three. The Bremen dataset contains ground-truth, which is employed for benchmarking the accuracy and robustness (with regard to occlusions and partial observations) of the search algorithm.

As mentioned in Section 6.2, parameter tuning is necessary for the correspondence search. Experimental tuning has been performed offline in this work. The parameters are closely related to the measurement model of the 3D scanner. Actually the parameters can be tuned by aligning several scan pairs manually or by existing registration approaches such as ICP and NDT. For segmentation, the cached octree region growing algorithm and the ASUFRI method have been chosen for generating area attributed planar segments.

Table 6.1: Tuned parameters for the correspondence search algorithm, which is used in the Barcelona dataset.

Parameter	$\hbar$ [m^2]	λ	τ	μ [deg]	χ [deg]	ξ	η	κ
Value	1.5	0.4	0.866	1.5	30.0	0.99965	0.03	4

6.4.1 The Barcelona Dataset

To evaluate the proposed transformation search algorithm, the first 20 point clouds in the dataset were selected. The 20 scans were gathered at the "FIB plaza" of the UPC Nord Campus in Barcelona, a 3D map of the plaza from "Nokia Maps" is shown in Figure 6.1. We did not select more point clouds because the accumulation registration error increases along with the number of point clouds and the map would be bent. As a result, loop closure detection techniques and error distribution methods are needed to refine the map. The tuned parameters for this dataset are illustrated in Table 6.1. Furthermore, since odometry information is available in this dataset, the scans can also be registered by local registration algorithms such as ICP and NDT. As a benchmark for comparison, a standard ICP implementation in PCL was employed. In the implementation, octree is utilized for nearest neighbor search to accelerate the convergence speed, and the robustness was increased by rejecting inconsistent correspondences. Comparison to other state of the art implementations of local registration algorithms is interesting, but has not been done at present and will be investigated in the future.

Since ground-truth of the robot poses is not provided in the dataset, manual pairwise inspection has been employed to check whether the registration is successful. For the sake of completeness, a 3D map based on the recorded odometry only is shown in Figure 6.2(a). It is significantly bent by the odometry, especially at the bottom right part of the figure which has been marked by a red rectangle. Figure 6.2(c) illustrates the registration result of ICP without initial pose estimation from odometry, which is definitely unusable. The registration result of odometry-assisted ICP is depicted in Figure 6.2(b). The maximum correspondence distance was set to 0.1 m during the experiment (other thresholds between 0.05 and 0.5 m have also been tested, but yielded worse results). The same area as that in Figure 6.2(a) has also been marked, and a close-up view of the marked area has been visualized in Figure 6.2(d), where map improvements by the ICP algorithm can hardly be observed. The map is still bent, which is unsatisfying for localization and path planning.

For the proposed search algorithm, all successive scan pairs have been found to be aligned properly without using the odometry information. Two typical scan pairs are visualized in Figure 6.3, with the new scan colored green and the old one colored red. The left column in each row shows the "registration" results of two scans after an identity transformation, i.e, without any pose estimation. The right column shows the registration results of the proposed approach.

The reconstructed map from the selected 20 point clouds has been drawn from two perspectives in Figure 6.4, for an animated video of the map please see Appendix A. The recovered poses of the robot have been drawn in Figure 6.5. The robot moved very

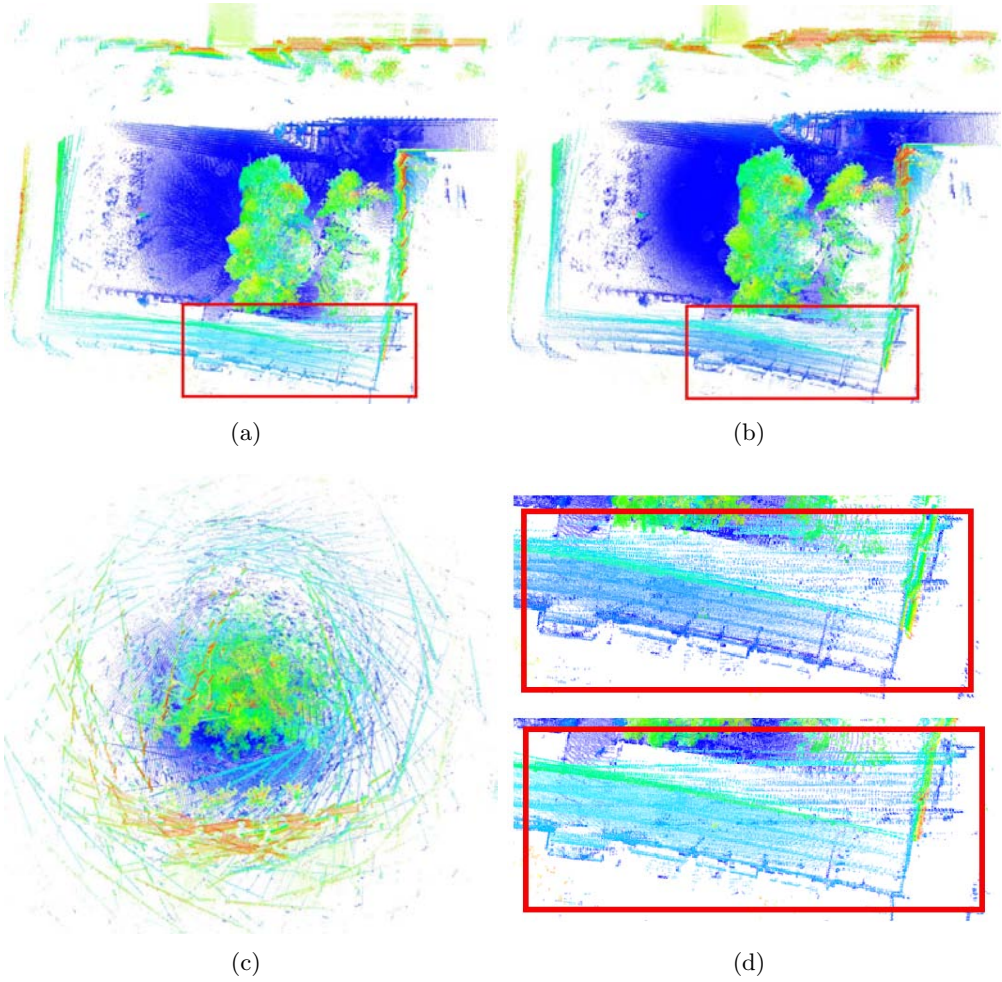
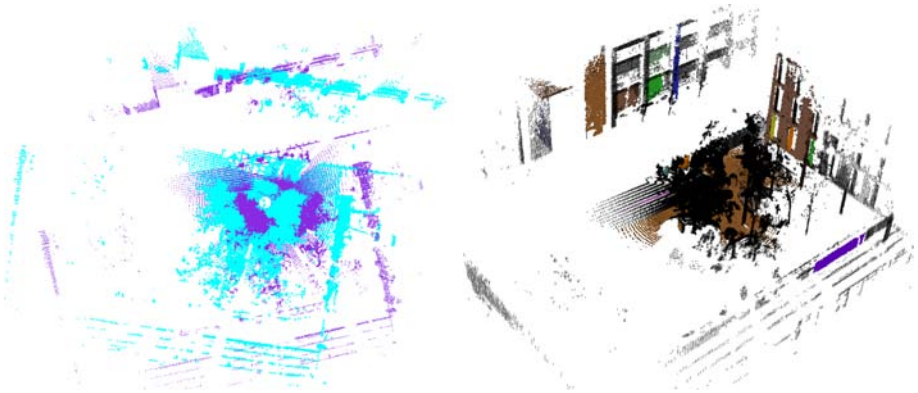


Figure 6.2: (a): A 3D "map" based on the first 20 scans in the Barcelona dataset, created using only odometry information; (b): ICP registration result, the recorded odometry was employed as prior pose estimation; (c): ICP registration result, no prior pose estimation was used; (d): Close-up view of marked area in (a) and (b), with the top from (a) and bottom from (b) having been used.

slightly from scan 1 to scan 2 and from scan 10 to scan 11, so only 18 poses can be seen from Figure 6.5(a). Since the robot moved very little along the z-axis, it is omitted in the figure. Please note the large translations between successive scans except the aforementioned two pairs. The robot has moved about 39.0 m in total, i.e., the average translation between successive scans is approximately 2.3 m. The processing time at each step has been depicted in Table 6.2, where n_l and n_r denote the number of planar segments considered for transformation search, n_{ac} and n_c stand for the number of found area-consistent pairs and correspondences. It can be seen that the search time is strongly dependent on n_{ac}, as the number of combinations to be checked for fixing three nonparallel pairs is $\binom{n_{ac}}{3}$. The average time for each step has also been illustrated in the table. More importantly, less than 2.21 s (1.20 s on average) were needed to align a new scan to the fixed coordinate system.



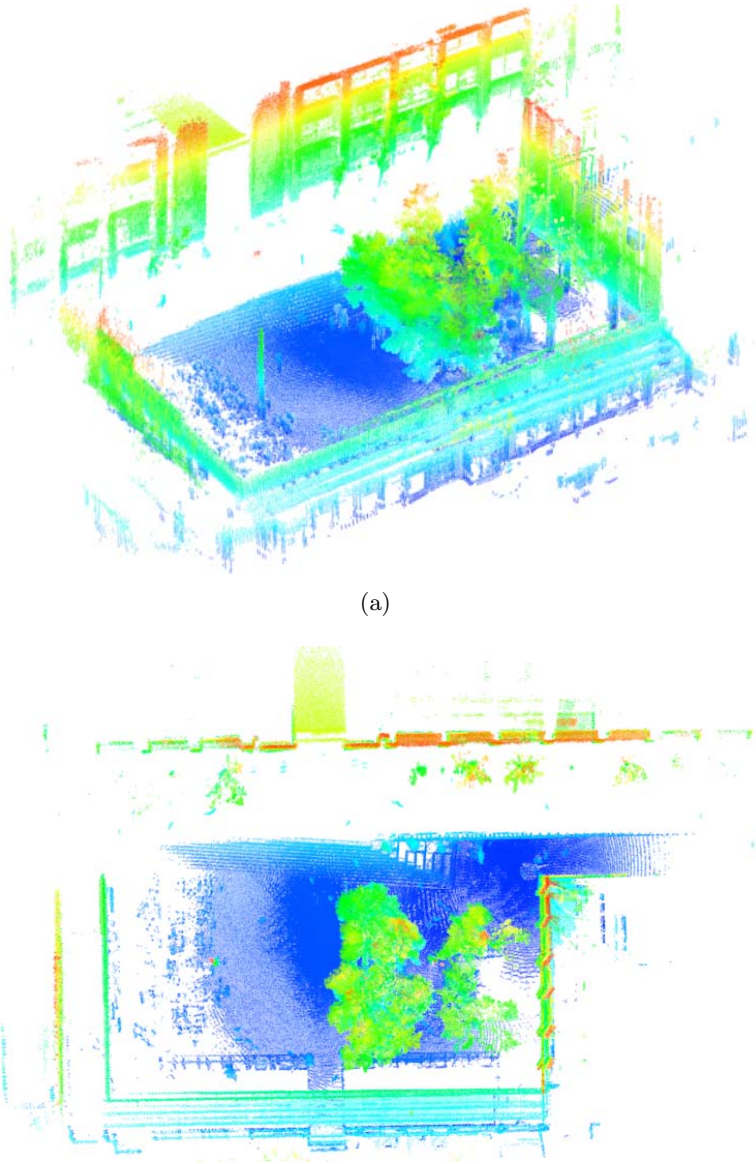
(a) (b)

Figure 6.3: A pairwise (scan 5 $\leftrightarrow$ 6) registration result in the Barcelona dataset. (a): before registration; (b): registration result, points of planar surfaces whose correspondences have been determined are shown in the same color, other points are shown in black.

Table 6.2: Processing time at each step for the first 20 scans in the Barcelona dataset.

	seg. [s]	area [ms]	n_l	n_r	n_{ac}	n_c	search [s]
1	1.024	3.17	–	–	–	–	–
2	0.974	3.42	30	39	525	27	0.080
3	0.966	3.03	39	48	1063	14	0.356
4	1.005	3.16	48	51	1564	21	1.199
5	1.073	3.24	51	39	1178	21	0.558
6	1.162	3.77	39	42	920	17	0.222
7	1.263	4.09	42	39	859	17	0.182
8	1.273	4.41	39	37	741	10	0.139
9	1.039	3.23	37	34	616	17	0.076
10	0.854	2.69	34	25	391	8	0.022
11	0.861	2.68	25	28	354	18	0.016
12	0.783	2.51	28	28	402	7	0.014
13	0.764	2.47	28	31	427	12	0.022
14	0.763	2.55	31	25	361	9	0.024
15	0.784	2.46	25	28	301	11	0.015
16	0.871	2.65	28	28	320	11	0.011
17	0.983	3.09	28	29	346	4	0.010
18	1.489	4.79	29	25	324	7	0.020
19	1.582	4.91	25	25	230	4	0.010
20	1.346	4.20	25	29	280	6	0.024
average	1.043	3.33	–	–	–	–	0.158

Normally, for ground mobile robots with an aLRF, the time to move a robot from one scan position to the next position with a distance of more than 2 m and to take a 3D scan is longer than the above registration time. Therefore, for these robotic systems, the proposed approach can be performed online.



(a)

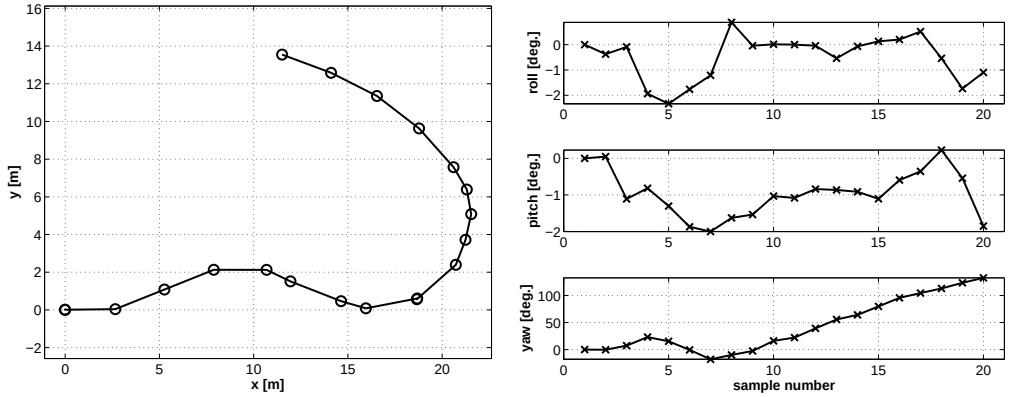
(b)

Figure 6.4: Two perspectives of the reconstructed map from the first 20 point clouds in the Barcelona dataset. The points are colored by height.

6.4.2 The Hamburg Dataset

Two parameters are different from the Barcelona dataset in the correspondence search step: $\hbar$ has been changed to 0.5 because there are many small surfaces whose area is smaller than 1.5 m^2, and λ has been modified to 0.35 to speed up the search procedure. In other words, the approach works after only slight parameters changes, which indicates the robustness of the registration approach.

This dataset is more challenging than the Barcelona dataset because larger rotations and translations have been made intentionally during successive scans. This is intended



(a) (b)

Figure 6.5: The recovered robot poses during the first 20 scans in the Barcelona dataset where the robot started from the coordinate origin. (a): x, y positions; (b): roll, pitch and yaw angles in degrees.

to test the robustness of the search algorithm. As in the Barcelona dataset, ground-truth of the robot poses is unavailable, hence manual pairwise inspection has been employed. All 18 successive scan pairs have been found to be aligned properly. A typical registration result has been illustrated in Figure 6.6.

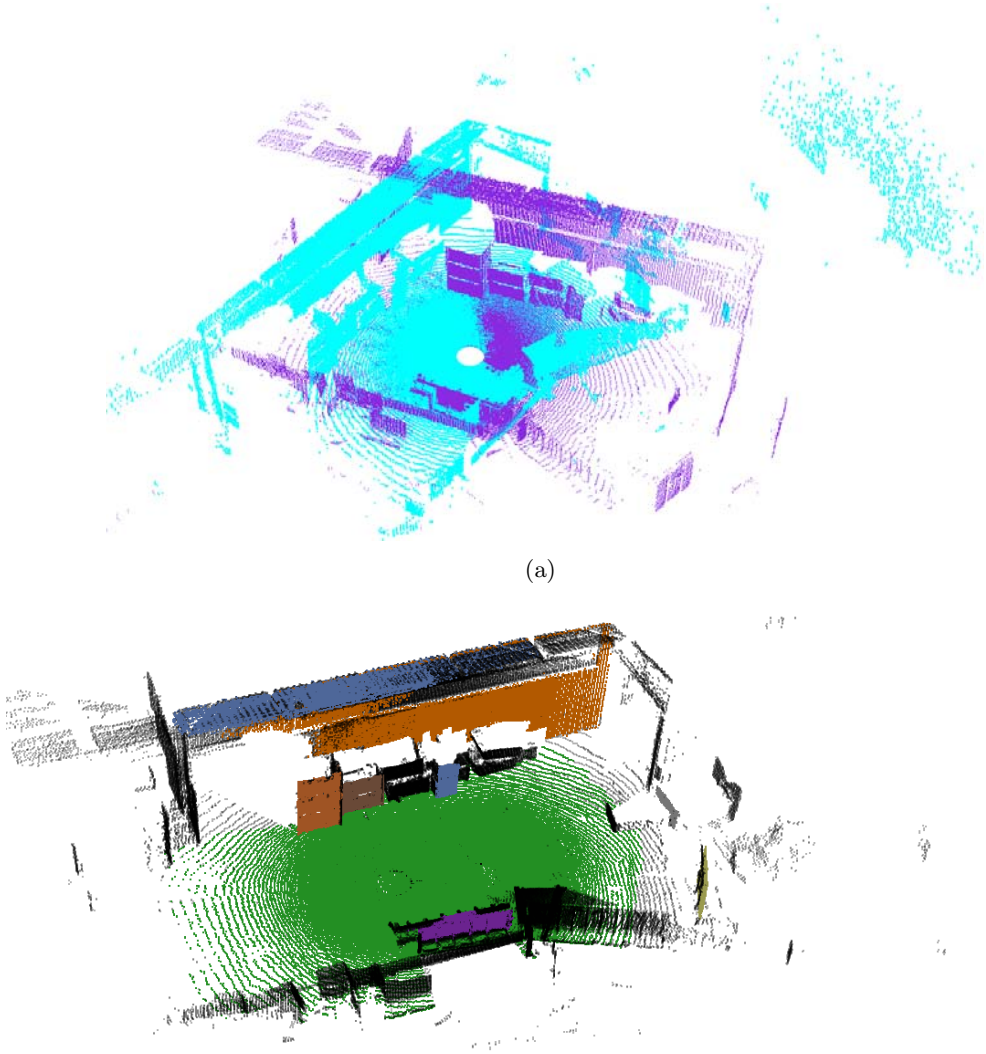
The reconstructed map from the 19 point clouds has been illustrated from two different viewpoints in Figure 6.7, see Appendix A for the access to an animated video of the map. The recovered poses of the robot have been drawn in Figure 6.8. Note the larger translations between successive scans compared to the Barcelona dataset. The robot has moved about 56.5 m in total, which means an average translation of 3.1 m between successive scans. The largest translation happened between scan 7 and scan 8 (about 5.40 m), which is quite large compared to the sensor's detection range (30 m). Large yaw angle changes between successive scans can be seen in Figure 6.8(b), for example scan pair $2 \leftrightarrow 3$, $6 \leftrightarrow 7$, $11 \leftrightarrow 12$ and $16 \leftrightarrow 17$.

From this experiment, we can conclude that the proposed approach is robust, as only a slight parameter change was made compared to that used in the dataset above. Moreover, it has the ability to align point clouds with large rotations and translations, without employing any prior pose estimations.

6.4.3 The Texas Dataset

This dataset is chosen for evaluating our registration approach on a sensor with relatively narrow FoV, since both sensors employed for the previous datasets have an almost omni-directional FoV. Note that ICP has already been found to produce several failed pair alignments (see Pathak et al., 2010a, Section 3.2). The tuned parameters for registration are depicted in Table 6.3. It can be seen that only the following parameters have been changed compared to Table 6.1:

1. $\hbar$ has been set to 0.2. This is intuitive because the planar surfaces in this dataset are smaller.



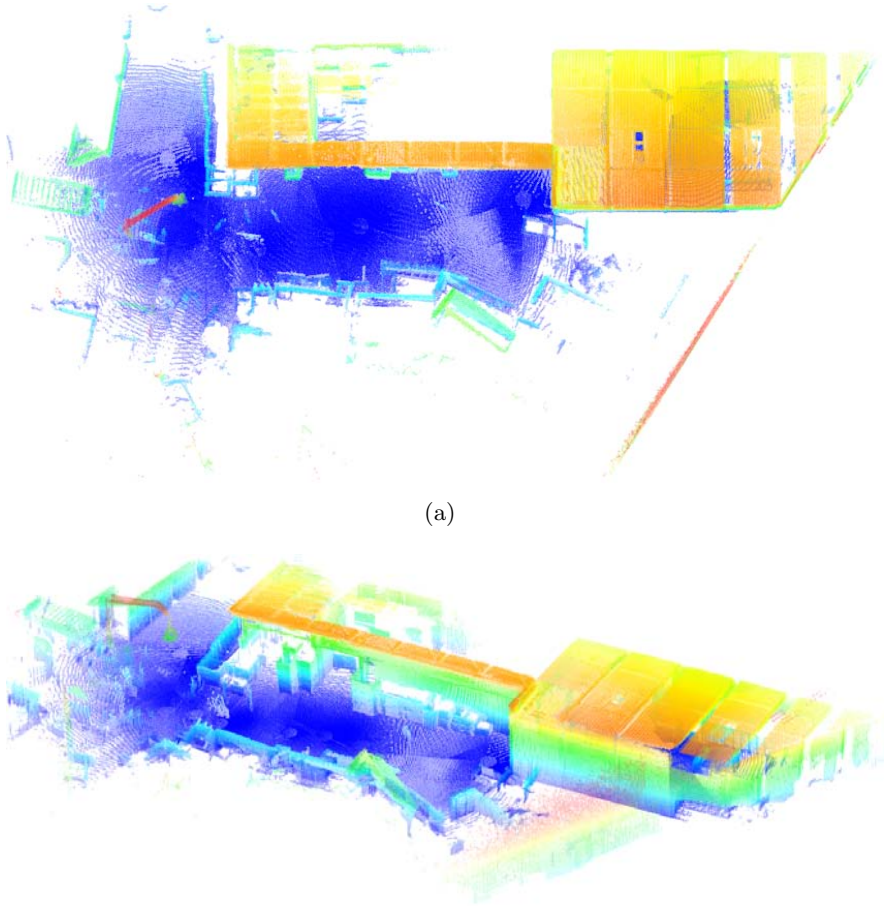
(a)

(b)

Figure 6.6: A pairwise (scan 9 $\leftrightarrow$ 10) registration result in the Hamburg dataset. (a): before registration; (b): registration result, points of planar surfaces whose correspondences have been determined are shown in the same color, other points are shown in black.

2. λ has been set to 0.5 in order to deal with occlusions, especially partial observations as the dataset is from a non-full FoV scanner. Clearly, this will explore more potential correspondences, which improves the robustness while requiring more computational time.
3. μ and η have been set to 3.0 and 0.036 respectively. This is mainly due to the fact that the flatness of planar surfaces is worse than that of those in the Barcelona dataset, as many planar surfaces are bent. As a result, the estimated plane parameters have a higher uncertainty, so we use larger thresholds for geometric consistency evaluation.

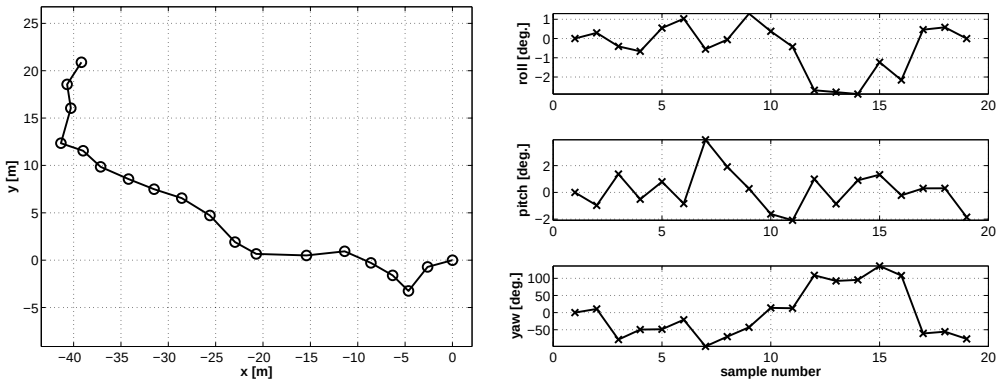
Again, there is no ground-truth in this dataset, thus manual pairwise inspection has been employed for each alignment. All successive point cloud pairs have been



(a)

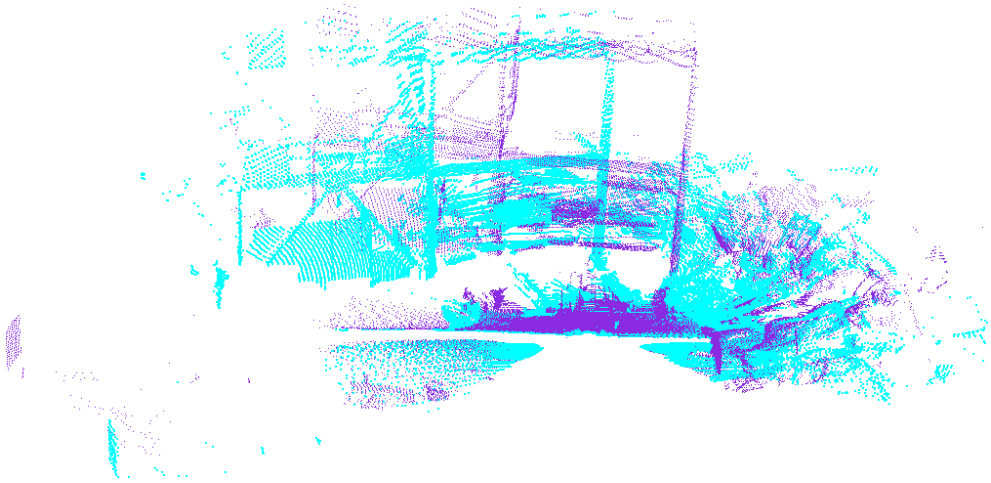
(b)

Figure 6.7: Two perspectives of the reconstructed map from the 19 point clouds in the Hamburg dataset, the points have been colored by height.

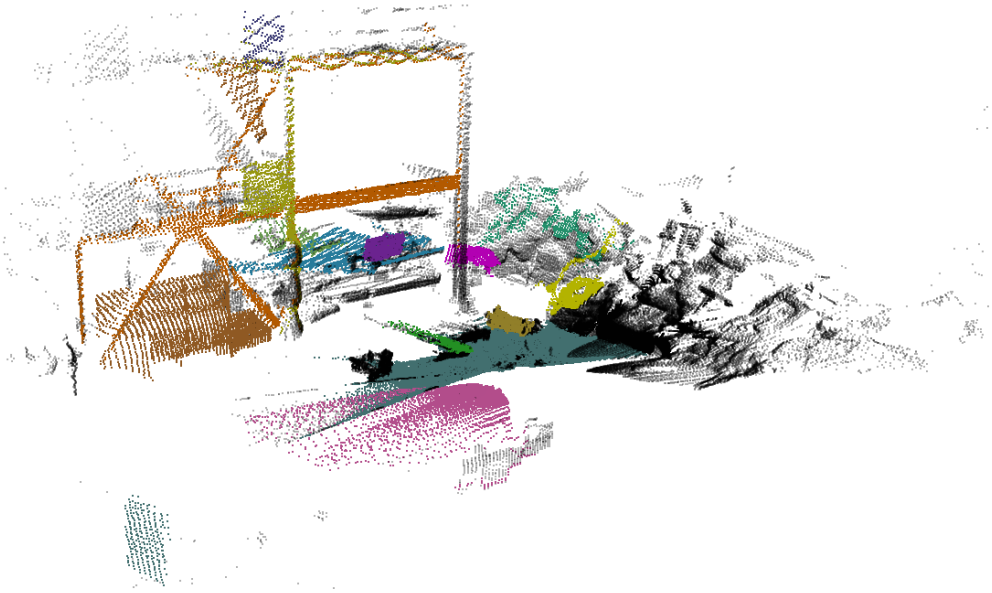


(a) (b)

Figure 6.8: The recovered robot poses during the 19 scans in the Hamburg dataset. (a): x, y positions; (b): roll, pitch and yaw angles in degree.



(a)



(b)

Figure 6.9: A pairwise (scan 11 $\leftrightarrow$ 12) registration result in the Texas dataset. (a): before registration; (b): registration result, points of planar surfaces whose correspondences have been determined are shown in the same color, other points are shown in black.

Table 6.3: Tuned parameters in the correspondence search step for the Texas dataset.

Parameter	$\hbar$ [m^2]	λ	τ	μ [deg]	χ [deg]	ξ	η	κ
Value	0.2	0.5	0.866	3.0	30.0	0.99965	0.036	4

found to be aligned properly by our algorithm. A typical registration result is shown in Figure 6.9, where the matched segments have been marked using the same color. The registration results of 26 scans are illustrated in Figure 6.10, and the recovered poses

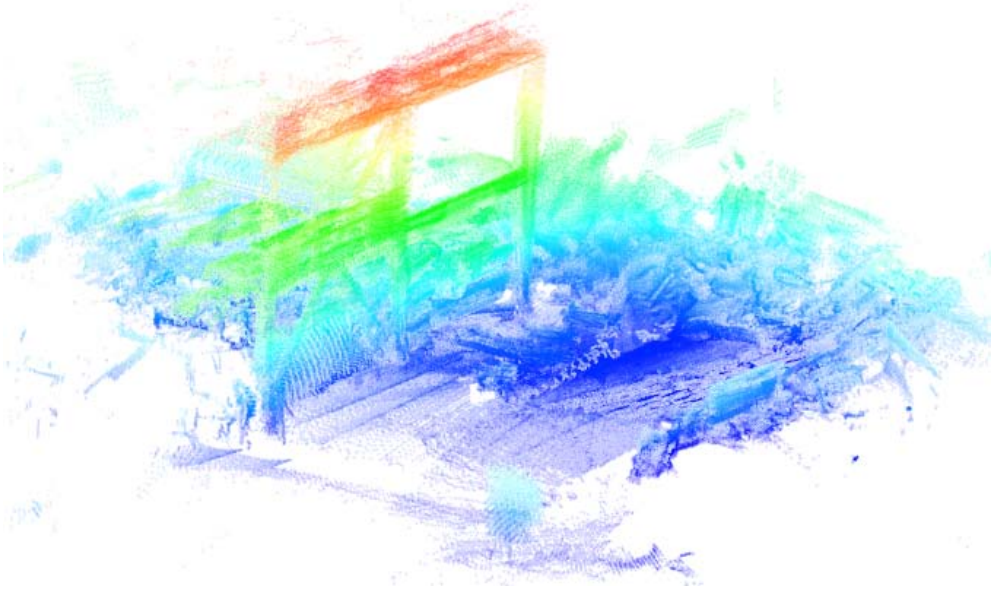
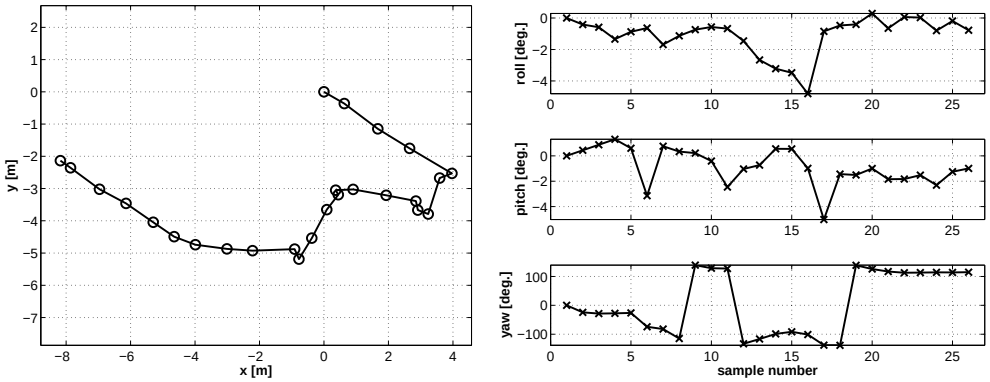


Figure 6.10: Registration result of all 26 scans from the Texas dataset, the points have been colored by height.



(a) (b)

Figure 6.11: The recovered robot poses in the Texas dataset. (a): x, y positions; (b): roll, pitch, yaw angles.

of the robot have been drawn in Figure 6.11. An animated video of the map can be downloaded from the Web (see Appendix A).

6.4.4 The Bremen Dataset

In contrast to the aforementioned datasets, this dataset has a ground-truth which is obtained based on commercial reflective marker-based registration, see Pathak et al. (2010c) for detail. Except being employed for evaluating registration accuracy with ground-truth, this dataset has also been employed to evaluate the robustness of MUMC and the predicted power of two overlap-metrics. It is challenging for registration due to a large distance between successive scan poses (see the ground-truth in Table 6.5).

Table 6.4: Tuned parameters in the transformation search step for the Bremen dataset.

Parameter	$\hbar$ [m^2]	λ	τ	μ [deg]	χ [deg]	ξ	η	κ
Value	5.0	0.5	0.985	2.0	30	0.9994	0.05	4

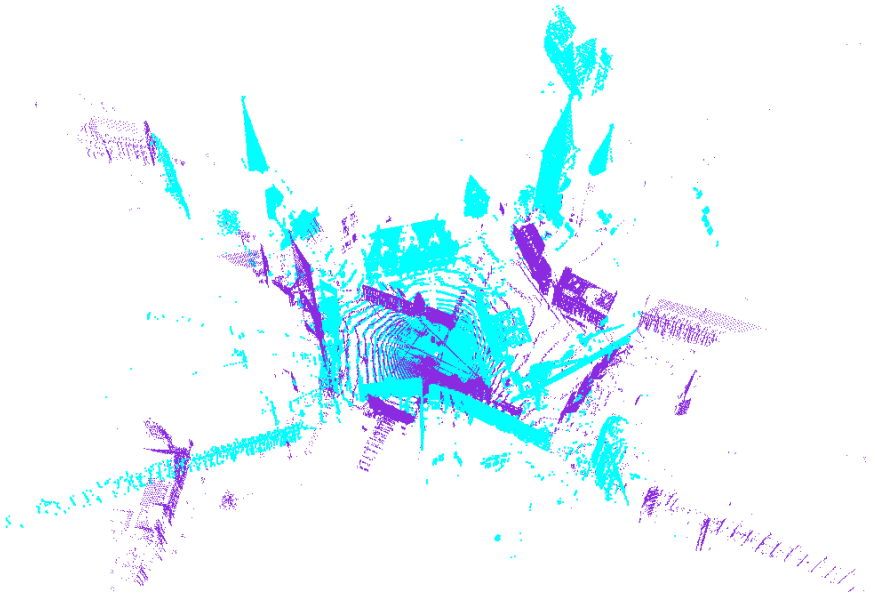
As a result, this dataset features a high presence of occlusions and partial observations. It is employed for two purposes in this work: registration accuracy evaluation and robustness test with respect to occlusions and partial observations.

Tuned thresholds for the search algorithm are depicted in Table 6.4, $\hbar$ has been changed to 5.0 because of the large scale of the point clouds. There is one failed pair registration (scan 0 to 1); this is due to an insufficient overlap. A comparison of all registration results against the ground-truth is illustrated in Table 6.5. For each pairwise registration, the first row shows the registration result of our algorithm, the second row lists the ground-truth. Again, no initial pose estimation from odometry or other sensors was employed in our algorithm. A typical registration result is shown in Figure 6.12, where points of corresponding planar surfaces are shown in the same color, and other points are shown in black. It can be seen that the registration result agrees well with the ground-truth in the successful cases. Compared to the result of MUMC, our algorithm yields similar accuracy. In other words, the yaw is always estimated more accurately than roll and pitch, and the z displacement is worse than the x and y displacement. The main reason has been analyzed in Pathak et al. (2010c), i.e., the ground is not matched due to unevenness. In Table 6.5, n_l and n_r denote the number of planar segments considered for transformation search, while n_c stands for the number of found correspondences. The time for the transformation search has also been listed in the table.

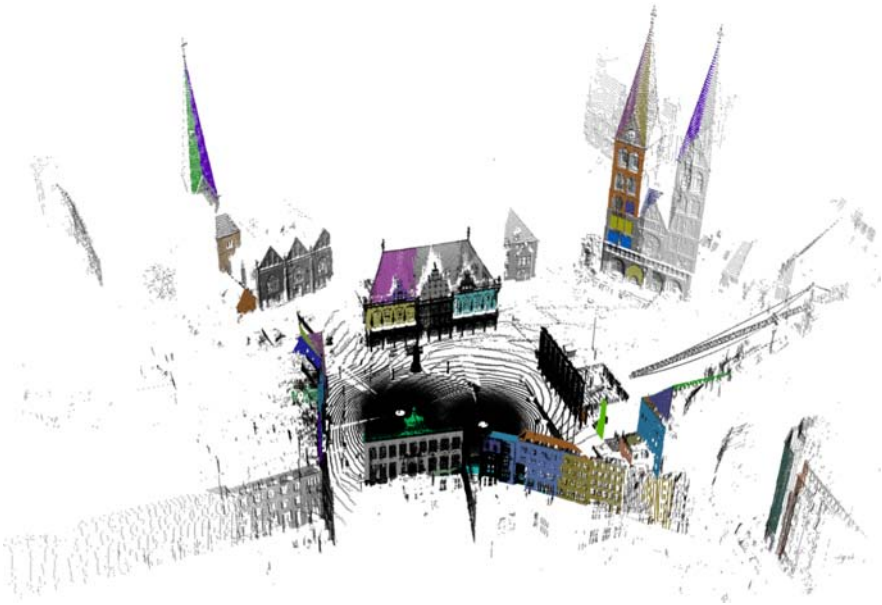
Compared to MUMC, we use only the plane parameters and area of each segment for determining segment correspondences. On the one hand, our algorithm needs less time. On the other hand, segments with small overlap can be missed in the result, see the number of found correspondences in Table 6.5. In addition, our algorithm has other limitations: there is no uncertainty in the registration result, which is essential for embedding it into a SLAM system. This, however can be overcome by calculating the uncertainty matrix as in Pathak et al. (2010b).

6.5 Map merging

Map merging is the problem of fusing two or more partial maps into a global consistent map. It is typically required for multi-robot exploration, as partial maps gathered by different robots should be fused to build a globally consistent map. It has potential application areas such as search and rescue missions. A team of robots are commonly used to cover the scenario as soon as possible, as time is critical for searching victims and survivors. In certain situations, urban environments for example, there are many occasions when the GNSS receiver fails to deliver enough information to support an error free navigation. Therefore, no global coordinate system is available at the beginning on which the robots' poses can be mapped, resulting in no known relation between the



(a)



(b)

Figure 6.12: A pairwise (scan 11 ↔ 12) registration result in the Bremen dataset. (a): before registration; (b): registration result, points of planar surfaces whose correspondences have been determined are shown in the same color, other points are shown in black.

individual maps the robots generate. Each robot has its own reference frame which is usually fixed to its starting pose. Map merging is also useful for a single robot if it has several runs from different starting positions in the same large environment. Partial maps are obtained when either run can cover the environment due to power limitation etc. Individual maps, from individual robots in the multi-robot case, or individual runs

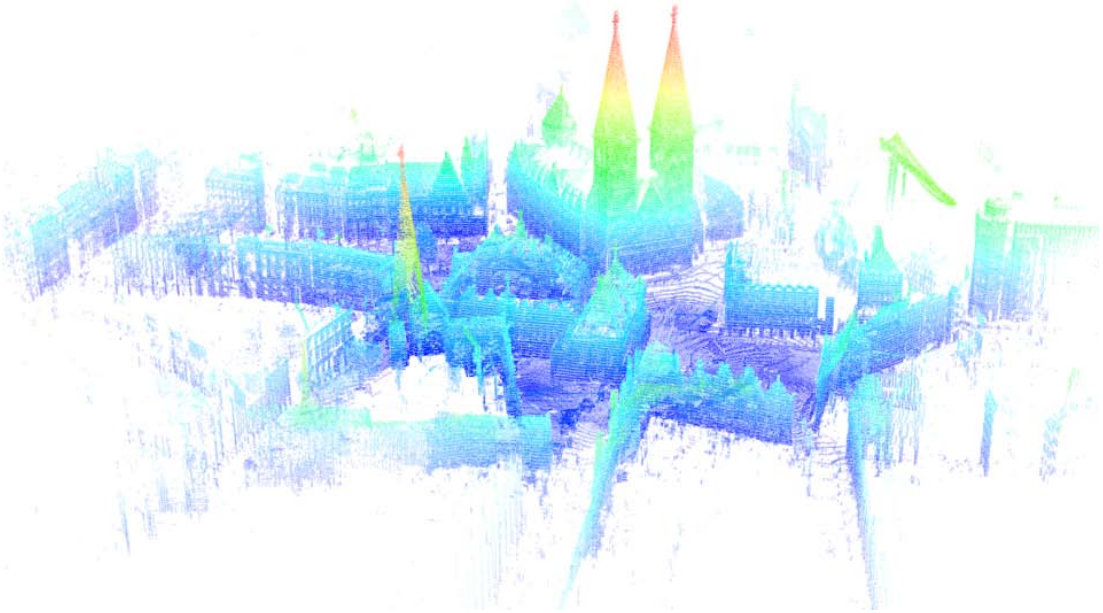


Figure 6.13: Registration result of scan 2 to 13 in the Bremen dataset. The points have been colored by height.

in the single robot case, will typically cover only a part of the environment. Therefore, they should be aligned to a common coordinate system in order to construct a more complete world model.

6.5.1 Related work on map merging

As stated in Konolige et al. (2003), "map merging is an interesting and difficult problem, which has not enjoyed the same attention that localization and map building have". Though a decade has past since this statement and although papers have been published during this period, it remains an open question, especially for 3D map merging. There are two kinds of map merging, i.e., topological map merging and metric map merging. Again, only research related to metric map merging is considered in this thesis.

Among the few papers on this topic, Carpin and his colleagues have several contributions: In Carpin et al. (2005), an adaptive random walk based motion planning is employed for transformation search, with the goal to maximize a target overlapping function. In a subsequent refinement (Birk and Carpin, 2006), mechanisms for detecting failures are introduced, which improves its robustness. Due to its iterative nature, the computational requirements are high which limits it to offline data processing. Then in Carpin (2008a,b), the Hough spectrum is utilized to extract spectral information from maps, and partial maps are combined based on matched spectral features. Similarly, Hough peak matching has been applied to map merging in Saeedi et al. (2012b), where the occupancy grid map is transformed to the Hough space first. The properties of the Hough transform are then used to find corresponding regions in the maps.

The problem is addressed as SLAM using a single virtual robot in Adluru et al. (2008), where the individual local maps and their shape information constitute the sensor information for the virtual robot. The Rao-Blackwellized particle filtering SLAM

Table 6.5: Comparison of registration results to the ground-truth. Registration: √, succeeded; ×, failed.

	Rotation [deg]			Translation [m]							
Pair	roll	pitch	yaw	t_x	t_y	t_z	n_l	n_r	n_c	Time [s]	Succ.
0→1	–	–	–	–	–	–	–	–	–	–	×
	0.32	0.77	-157.20	-1.513	41.223	-0.189					
1→2	0.31	0.97	43.43	-23.716	-23.943	0.660	91	123	36	9.000	√
	-0.08	-0.03	43.48	-23.452	-24.141	-0.640					
2→3	0.94	-0.47	-24.43	-9.687	-36.038	0.219	123	131	20	21.098	√
	0.07	0.08	-24.79	-9.972	-36.047	-0.322					
3→4	3.24	1.59	158.54	36.661	0.912	0.677	131	72	17	3.948	√
	1.13	1.11	158.48	36.575	0.917	1.063					
4→5	0.46	0.66	72.68	-22.191	0.143	0.094	72	64	10	0.540	√
	0.40	0.67	72.72	-22.198	0.123	0.142					
5→6	-1.51	1.04	-132.22	-21.318	5.882	0.040	64	55	6	0.303	√
	-1.36	0.55	-132.30	-21.377	5.685	-0.098					
6→7	1.56	0.56	-78.85	5.346	-20.355	-0.234	55	59	9	0.201	√
	0.59	1.11	-78.76	5.327	-20.383	-0.189					
7→8	-0.96	1.32	-51.57	25.244	-8.081	-0.700	59	84	7	0.407	√
	0.30	1.69	-51.52	25.295	-8.059	-0.447					
8→9	1.91	-2.39	-124.71	-8.534	26.683	-0.837	84	119	17	4.687	√
	2.23	-2.36	-124.58	-8.528	26.661	-0.932					
9→10	4.21	-0.02	153.42	-11.940	-22.754	0.868	119	138	24	20.817	√
	3.78	-0.21	153.52	-11.964	-22.750	0.753					
10→11	0.38	-1.59	-79.11	-12.244	20.766	-0.880	138	132	42	25.831	√
	0.86	-1.02	-79.05	-12.226	20.764	-0.663					
11→12	-0.72	0.21	-140.51	4.891	-37.653	0.021	132	126	20	18.324	√
	0.15	-0.13	-140.58	4.922	-37.676	0.689					

framework is then used to solve the map merging problem. Saeedi et al. (2012a) suggested to employ a probabilistic version of the Generalized Voronoi Diagram to determine the relative transformation between maps.

It should be noted that all above approaches are limited to merging 2D occupancy grid maps. To the best of our knowledge, 3D map merging is still an unexplored area. In this section, we try 3D map merging using our registration technique, which is described in the next section.

6.5.2 Planar segments based map merging

Our main assumption for map merging is the same as for scan registration, i.e., there is no knowledge about the relative positions of the partial maps which are to be fused. The only precondition in our work is that maps to be merged exhibit at least some overlap, since merging non-overlapping maps without known spatial relation is an ill-defined problem. In other words, we are not going to answer the question whether the given maps overlap or not, but going to identify the overlapping regions between maps and compute the corresponding transformation matrices which join the maps at a common frame.

The diagram of our planar segments based map merging approach is shown in Figure 6.14, which is almost the same as that for pairwise registration. Actually, it is

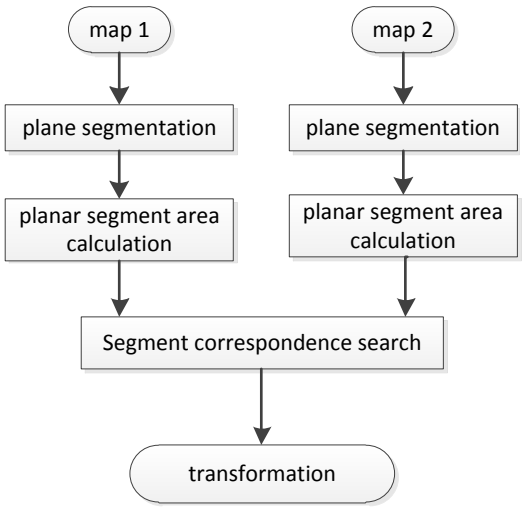


Figure 6.14: Diagram of planar segments based map merging, where the merging process is divided into three steps.

a straightforward extension of the pairwise registration, where scans are replaced by partial maps. The main differences lie in the first two steps:

- In the first step, for pairwise registration, which plane segmentation algorithm should be employed depends on whether the point cloud is organized; for map merging, only the cached octree region growing algorithm can be used.
- In the second step, for pairwise registration, which planar area calculation algorithm should be utilized depends on whether an image-like structure is available; for map merging, only the alpha-shape based method can be applied.

In order to evaluate the applicability of our registration to map merging, the Bremen dataset is employed again in a different manner. As shown in Figure 6.15, all the scans can make a loop by adding an edge (marked with a different color) between the first and last scan. Partial maps therefore can be constructed by adjacent scans. To evaluate the approach, partial maps from every two adjacent scans have been built based on ground-truth. As a result, we have the same number of partial maps as the scans. For the sake of clarity, we use $\mathcal{M}_i$ to denote a partial map, where i is the index of the first scan contained in it. For example, $\mathcal{M}_3$ stands for the combination of scans 3 and 4. Furthermore, the lower boundary of the overlapping area between partial maps is known, i.e., the overlapping area between $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ is no smaller than scan $i+1$.

To benchmark our approach, it has been applied to fusing all successive partial maps $\mathcal{M}_i, i = 1, 2, ..., 13$, a typical result is shown in Figure 6.16. Errors compared to the ground-truth and computational time for each pairwise fusing have been detailed in Table 6.6. The only threshold change compared to that of pairwise scan registration in this dataset (see Table 6.4) is that $\hbar$ is changed from 5.0 to 25.0 to scale the search space. It can be seen that the registration agrees well with the ground-truth, regarding both rotation and translation. The maximum time for fusing such two maps is 33.744 s. Considering map merging is not a frequent operation for multi-robot exploration, our approach is promising for online data processing.

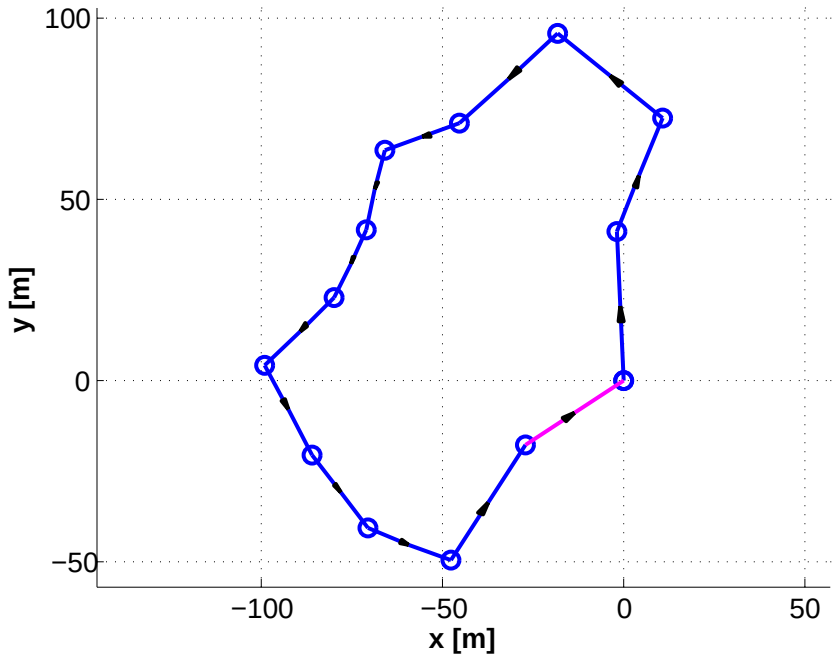


Figure 6.15: Ground-truth of sensor poses in the Bremen dataset.

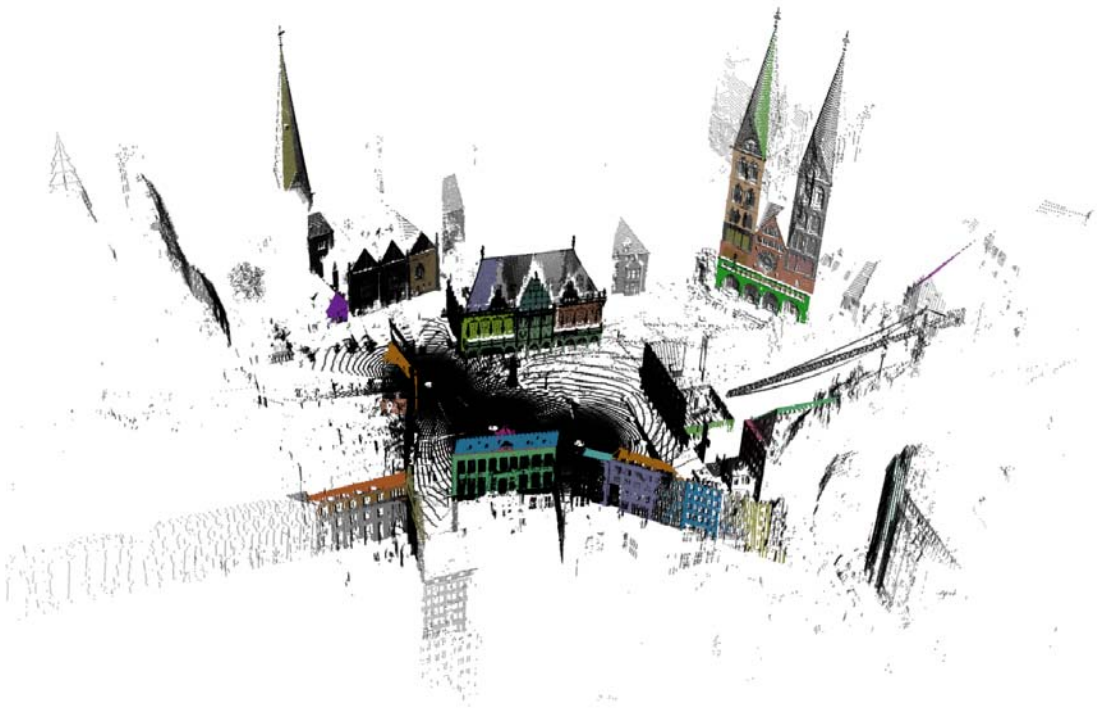


Figure 6.16: Map merging result of $\mathcal{M}_{10} \leftrightarrow \mathcal{M}_{11}$, points of planar surfaces whose correspondences have been determined are shown in the same color, other points are shown in black.

Table 6.6: Map merging results for the Bremen dataset. The relative error is given in each row, the ground-truth can be inferred from Table 6.5.

	Rotation [deg]			Translation [m]			
Pair	roll	pitch	yaw	t_x	t_y	t_z	Time [s]
1→2	0.06	-0.02	0.19	0.052	0.022	0.018	7.179
2→3	-0.49	0.01	0.24	0.083	0.100	-0.029	12.512
3→4	0.02	-0.01	-0.05	-0.100	-0.045	-0.027	10.302
4→5	-0.01	-0.01	0.01	0.008	-0.016	0.007	1.577
5→6	0.01	0.01	-0.01	-0.045	-0.016	-0.004	0.173
6→7	-0.05	-0.05	0.06	-0.041	0.045	-0.009	0.049
7→8	-0.04	-0.03	-0.03	-0.010	-0.031	-0.006	0.085
8→9	-0.11	0.06	0.38	0.042	-0.192	-0.038	0.883
9→10	-0.05	0.03	0.08	0.071	-0.175	-0.089	6.716
10→11	0.09	-0.04	0.00	-0.032	0.003	0.016	21.878
11→12	0.17	0.31	-0.05	0.023	0.002	-0.005	33.744
12→13	-0.06	-0.01	-0.01	0.016	-0.045	0.009	30.306

Our registration approach has been extended to the domain of map merging in this section, we show that our planar segments based map merging can efficiently deal with 3D map merging. Our approach has been evaluated using the Bremen dataset, with regard to both speed and accuracy. To the best of our knowledge, related work is still dealing with 2D map merging, hence cannot be compared to ours. In conclusion, our registration approach has a good applicability to map merging.

6.6 Summary

A planar segments based scan registration approach has been presented, which is resolved by determining the correspondences between two sets of area attributed planar segments. No prior pose estimation is required in the approach, i.e., it belongs to global registration. The global consistency is solved by maximizing a spherical-correlation-like metric, which is computed from the area of planar segments, where the segment attributes (plane parameter and area) are used for pruning the search space in SE(3). Four datasets have been employed to evaluate the search algorithm. From the experiments, the search algorithm is able to deal with data from different 3D scanners, is faster than ICP with higher accuracy, and has similar accuracy and robustness (with regard to occlusions) compared to MUMC. The algorithm has then been extended to the domain of map merging, and benchmarked with speed and accuracy. From the experiment, we claim our approach has a good applicability to map merging hence provides a solution for 3D map merging.

Although promising results have been obtained, the registration algorithm has its limitations. One critical point are the requirements of at least three corresponding non-parallel planar surface pairs, which cannot be guaranteed even in some urban areas. This excludes its application in environments with few planar surfaces. Additionally,

the registration uncertainty has not been considered yet, which is essential for embedding the approach into a SLAM system. Lastly, there are several parameters that should be selected properly according to the specific sensor, although they can be tuned intuitively.

Chapter 7

Conclusion

> Study the past, if you would divine the future.
>
> Confucius

THIS dissertation deals with 3D outdoor mapping. By restricting our focus to only plane-rich environments, a planar segments based scan registration approach has been presented. In such environments, the proposed algorithms can reconstruct an accurate 3D map fast and reliably, where initial pose estimations from other sensors are not required. As is known, for a mobile robot, the ability to model its surroundings with on-board sensors is a very fundamental issue not only for navigation, but also for object recognition and robot-environment interaction. This dissertation has thus made an important step towards fully autonomous field robots. In this chapter, the novel contributions are summarized in Section 7.1 along with the main conclusions to be drawn from the presented experiments. Current limitations and open questions to the presented approaches are then listed in Section 7.2. Finally, Section 7.3 outlines a number of possible improvements and directions for future research.

7.1 Contributions

Rather than repeating the list of contributions presented in Section 1.3, this section highlights the three most important achievements of this dissertation:

- The first notable contribution of this work are four complementary strategies for point cloud plane segmentation. Which strategy should be employed depends on whether the given point cloud is organized and the cluttered level of the environment. The cached octree region growing algorithm is emphasized here, as it can segment unorganized point clouds. In other words, it does not require the range image data structure which is advantageous compared to state-of-the-art algorithms. Furthermore, its time complexity is only $O(N \log N)$.

- Another main contribution of this dissertation are the area calculation methods for planar segments. Three approaches for calculating the area of planar segments have been presented, namely a 2D Delaunay triangulation based method,

a 2D alpha-shape based method, and a range-image based method. Being benchmarked with real-world datasets and compared with state-of-the-art algorithms, the alpha-shape based method is recommended if the given segments are from unorganized point clouds; otherwise, the range-image based method should be employed.

- The third important contribution is a constraint global search algorithm for determining the correspondences between two sets of planar segments, where the search space is pruned by both self-similarity (area) and interrelations (geometric constraints) and the objective function to maximize a spherical-correlation-like metric. Based on the found correspondences, a transformation matrix to align the corresponding scans are computed. From field experiments, the search algorithm is able to deal with data from different 3D scanners; in addition, it is accurate, fast and robust.

A mapping system constructed by the above techniques has been evaluated with four outdoor real-world datasets. Based on the results, it can be concluded that the approach provides an online solution for aLRF equipped robots. Furthermore, although proposed for outdoor environments, the approach is clearly capable to deal with indoor environments where planar surfaces are predominant. Actually, the ability to deal with indoor scenarios has been confirmed in the Hamburg dataset. Therefore, the workspace of service robots can be expanded to outdoor, at least to urban environments. This is particularly important for aging people who lack full mobility, as they should not be confined to their houses.

The approach has also been extended to the domain of map merging, which is an important issue for multi-robot exploration, especially for rescue robots. We human beings have experienced the 2008 Sichuan earthquake (Wikipedia, 2013a) in China and the 2011 Tōhoku earthquake (Wikipedia, 2013b) in Japan; sadly, there was another big earthquake (Wikipedia, 2013c) in China during the preparation of this thesis. In such situations, a team of rescue robots can be very useful (sometimes necessary), for example to inspect dangerous areas and to search for victims and survivors. Our approach can serve a proper perception module for them, i.e., scan registration for each single robot to build partial maps and map merging to construct a globally consistent map.

In conclusion, our registration pipeline is simple, easy to implement yet efficient and robust for plane-rich environments, can be employed with different 3D scanners, and has been extended to fusing maps. It has broad application prospects such as safety, security, rescue robots, and urban service robots, but is not limited to them.

7.2 Limitations and open questions

Although promising results have been obtained, our approach still has its limitations and open questions:

- An intrinsic limitation is the requirement of three planar surfaces with linearly independent normals, hence it cannot be applied to environments which cannot fulfill this requirement.

- Another drawback of the approach is the lack of pose covariance, which is essential for SLAM frameworks to distribute cumulated errors during incremental registration.
- As discussed, several parameters should be turned properly for both segmentation and correspondence determination, which has been done experimentally in this thesis. Strategies to automatically find optimum parameters will be more efficient and robust.
- Only the area has been employed as a self similarity metric. Although the approach has been found to be fast and reliable, other metrics such as the 2D shape and plane parameter uncertainties can be employed for further search space pruning.
- As surveyed in Chapter 2, there are many fundamental issues for robotic mapping. Our approach is still far from fully autonomous exploration; for example, loop closure detection, registration failure prediction, global relaxation, and drivable area detection have not been investigated yet.

7.3 Future research directions

According to the limitations presented in the above section, it is of foremost interest to apply the idea to higher order surfaces, i.e., determining corresponding segments by maximizing a spherical-correlation-like matrix and then computing the transformation based on matched segments. Quadrics are possible candidates for the representation of nonplanar segments. Furthermore, as stated in Chapter 5, our pixel-neighbor information based area calculation method is not limited to planar surfaces. Therefore, it can be used to supply an area attribute to each segment for data association.

To derive the pose covariance for our registration approach is very important in future work. Armed with pose covariances, the approach can be easily embedded into the popular pose-graph SLAM framework. A possible solution is to use the method of MUMC, i.e., calculating the pose uncertainty based on plane parameter uncertainties of each matched segment pair.

Planar segments based loop closure detection is also of interest. The 3D-NDT histogram has demonstrated its ability in this area. Note that the primary components of a 3D-NDT histogram are constructed from implicit small planar segments. Therefore, it is attractive to build similar histograms based on explicitly large planar segments and evaluate their performance with regard to loop closing.

It can be seen in Chapter 6 that unmatched planar segments have not been utilized in our approach. Actually, they can be employed with multiple purposes: First, the overlap between them can increase the registration's robustness, as has been done in MUMC. Second, the ratio between the area of matched and unmatched segments can be used to predict the confidence of the registration; at least, it can measure the overlapping ratio of two scans.

As discussed, the complexity of the segment correspondence search highly depends on the number of area-consistent pairs. For two given scans, the number of area-consistent pairs is related to two parameters, the minimum area $\hbar$ and the area ratio

λ. The effect of λ has been discussed in Chapter 6. For $\hbar$, on the one hand, if it is too large, the possibility to fail will be increased; on the other hand, if it is too small, the time complexity increases quickly. Suppose there are N_l and N_r planar segments in the target and source scan, and N area-consistent segment pairs between them. The breadth-first search has a complexity of $O(N^3)$. Theoretically, segment triplets with linearly independent normals can also be enumerated in each scan first, which has a time complexity of $O(N_l^3)$ and $O(N_r^3)$. Then consistent triplet pairs can be searched between two sets of them. It would be interesting to compare the performance of the two search strategies with regard to speed.

Another interesting idea to bound the computational cost is to construct a constellation graph for each scan based on detected surfaces. A constellation graph is an undirected graph which encodes the spatial relations between the segments. Then the segment correspondences can be inferred from matching constellation graphs. As is known, there are available algorithms for matching partially overlapped graphs. So the remaining problem is how to construct the constellation graphs which should be rotation invariant.

For the experiments in this thesis, all algorithms have been implemented to run under a single thread; there are several parts that can be accelerated by multiple-threading. For example, in the region growing algorithms, the selected seeds can be assigned to different threads for growing; in the segment area calculation algorithms, especially for the alpha-shape based method, the segments can be assigned to different threads; and in the correspondence search procedure, the found triplets can be grouped into different threads.

Appendix A

Source code, datasets, and videos

The source code for all proposed algorithms in this dissertation, including data gathering, point cloud segmentation, planar segment area calculation, as well as planar segments based scan registration and map merging, can be accessed from the following URL: `https://github.com/junhaoxiao/TAMS-Planar-Surface-Based-Perception.git`. The turned segmentation and registration parameters for each dataset have been provided together with the code, therefore the remaining segmentation, registration and map merging results can be easily reproduced. Furthermore, a manual about how to compile the code has also been supplied.

The organized point clouds for the Barcelona dataset and the reordered point clouds for the Texas dataset can be easily recovered with the method presented in Chapter 3. As an alternative, they can also be downloaded from the following URL: `http://tams.informatik.uni-hamburg.de/research/datasets.php`.

Videos showing the registration results for all datasets in Chapter 6 can be downloaded from the following URL: `http://tams.informatik.uni-hamburg.de/videos/index.php`.

References

Adept MobileRobots (2012). Pioneer 3-at. `http://www.mobilerobots.com/researchrobots/p3at.aspx`. Retrieved May 4, 2013.

Adler, B., Xiao, J., and Zhang, J. (2012). Towards autonomous airborne mapping of urban environments. In *IEEE International Conference on Multisensor Fusion and Information Integration*, Hamburg, Germany.

Adler, B., Xiao, J., and Zhang, J. (2013). Finding next best views for autonomous uav mapping through gpu-accelerated particle simulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Under review, available at `http://tams.informatik.uni-hamburg.de/people/adler/publications/adler_2013_submitted_to_iros.pdf`.

Adluru, N., Latecki, L., Sobel, M., and Lakaemper, R. (2008). Merging maps of multiple robots. In *19th International Conference on Pattern Recognition*, pages 1–4.

Amenta, N., Bern, M., and Kamvysselis, M. (1998). A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, New York, NY, USA.

Andreasson, H. and Lilienthal, A. J. (2010). 6d scan registration using depth-interpolated local image features. *Robotics and Autonomous Systems*, 58(2):157–165.

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117.

Bajaj, C. L., Bernardini, F., and Xu, G. (1995). Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 109–118, Los Angeles, USA.

Bauer, J., Karner, K., Schindler, K., Klaus, A., and Zach, C. (2003). Segmentation of building models from dense 3d point-clouds. In *27th Workshop of the Austrian Association for Pattern Recognition*, pages 253–259, Laxenburg, Austria.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256.

Biber, P. and Strasser, W. (2003). The normal distributions transform: a new approach to laser scan matching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2743–2748, Las Vegas, Nevada, USA.

Birk, A. and Carpin, S. (2006). Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, 94(7):1384–1397.

Birk, A., Vaskevicius, N., Pathak, K., Schwertfeger, S., Poppinga, J., and Buelow, H. (2009). 3-d perception and modeling. *IEEE Robotics & Automation Magazine*, 16(4):53–60.

Blaer, P. and Allen, P. (2007). Data acquisition and view planning for 3-d modeling tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 417–422, San Diego, California, USA.

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

Borrmann, D., Elseberg, J., Lingemann, K., Nchter, A., and Hertzberg, J. (2008). Globally consistent 3d mapping with scan matching. *Robotics and Autonomous Systems*, 56(2):130–142.

Bosse, M. and Zlot, R. (2009a). Continuous 3d scan-matching with a spinning 2d laser. In *IEEE International Conference on Robotics and Automation*, pages 4312–4319, Kobe, Japan.

Bosse, M. and Zlot, R. (2009b). Keypoint design and evaluation for place recognition in 2d lidar maps. *Robotics and Autonomous Systems*, 57(12):1211–1224.

Bosse, M. and Zlot, R. (2010). Place recognition using regional point descriptors for 3d mapping. In *Field and Service Robotics*, volume 62 of *Springer Tracts in Advanced Robotics*, pages 195–204.

Boston Dynamics (2013). Bigdog - the most advanced rough-terrain robot on earth. `http://www.bostondynamics.com/robot_bigdog.html`. Retrieved April 22, 2013.

Carpin, S. (2008a). Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316.

Carpin, S. (2008b). Merging maps via hough transform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1878–1883, Nice, France.

Carpin, S., Birk, A., and Jucikas, V. (2005). On map merging. *Robotics and Autonomous Systems*, 53(1):1–14.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155.

Connolly, C. (1985). The determination of next best views. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, pages 432–435, St. Louis, Missouri, USA.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665.

Da, T. K. F. (2012). 2d alpha shapes. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.1 edition.

Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

Dold, C. (2005). Extended gaussian images for the registration of terrestrial scan data. In *Proceedings of the ISPRS Workshop Laser scanning*, pages 180–185, Enschede, the Netherlands.

Dorninger, P. and Nothegger, C. (2007). 3d segmentation of unstructured point clouds for building modelling. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(3/W49A):191–196.

Doroodgar, B., Ficocelli, M., Mobedi, B., and Nejat, G. (2010). The search for survivors: Cooperative human-robot interaction in search and rescue environments using semi-autonomous robots. In *IEEE International Conference on Robotics and Automation*, pages 2858–2863, Anchorage, Alaska, USA.

Douillard, B., Quadros, A., Morton, P., Underwood, J., De Deuge, M., Hugosson, S., Hallstrom, M., and Bailey, T. (2012). Scan segments matching for pairwise 3d alignment. In *IEEE International Conference on Robotics and Automation*, pages 3033–3040, St. Paul, Minnesota, USA.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the segmentation of 3d lidar point clouds. In *IEEE International Conference on Robotics and Automation*, pages 2798–2805, Shanghai, China.

Dube, D. and Zell, A. (2011). Real-time plane extraction from depth images with the randomized hough transform. In *IEEE International Conference on Computer Vision Workshops*, pages 1084–1091, Barcelona, Spain.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping (slam): part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110.

Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559.

Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72.

Eich, M., Dabrowska, M., and Kirchner, F. (2010). Semantic labeling: Classification of 3d entities based on spatial feature descriptors. In *Best Practice Algorithms in 3D Perception and Modeling for Mobile Manipualtion, IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the rgb-d slam system. In *IEEE International Conference on Robotics and Automation*, St. Paul, Minnesota, USA.

Fischer, D. and Kohlhepp, P. (2000). 3d geometry reconstruction from multiple segmented surface descriptions using neuro-fuzzy similarity measures. *Journal of Intelligent and Robotic Systems*, 29:389–431.

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

FLIR (2013). Pan-Tilt Unit-D48E. `http://www.flir.com/mcs/view/?id=53670`. Retrieved April 22, 2013.

Frese, U., Larsson, P., and Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Georgiev, K., Creed, R. T., and Lakaemper, R. (2011). Fast plane extraction in 3d range data based on line segments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3808–3815, San Francisco, CA, USA.

González-Baños, H., Mao, E., Latombe, J.-C., Murali, T., and Efrat, A. (2000). Planning robot motion strategies for efficient model construction. In *International Symposium on Robotics Research*, volume 9, pages 345–352.

Granström, K. and Schön, T. (2010). Learning to close the loop from 3d point clouds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2089–2095, Taipei, Taiwan.

Granström, K., Schn, T. B., Nieto, J. I., and Ramos, F. T. (2011). Learning to close loops from range data. *The International Journal of Robotics Research*, 30(1):80–117.

Grisetti, G., Kummerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43.

Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. `http://eigen.tuxfamily.org`. Retrieved April 22, 2013.

Hai, D., Zhang, H., Xiao, J., and Zheng, Z. (2010). Cooperate localization of a wireless sensor network (wsn) aided by a mobile robot. In *IEEE International Workshop on Safety Security and Rescue Robotics (SSRR)*, pages 1–6, Bremen, Germany.

Hänel, D., Burgard, W., and Thrun, S. (2003). Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27.

Harati, A., Gächter, S., and Siegwart, R. (2007). Fast range image segmentation for indoor 3d-slam. In *IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France.

Harati, A. and Siegwart, R. (2007). Orthogonal 3d-slam for indoor environments using right angle corners. In *European Conference on Mobile Robots*, Freiburg, Germany.

He, W., Ma, W., and Zha, H. (2005). Automatic registration of range images based on correspondence of complete plane patches. In *Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 470–475, Ottawa, Canada.

Hegde, G.-P. and Ye, C. (2009). Extraction of planar features from swissranger sr-3000 range images by a clustering method using normalized cuts. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4034–4039, St. Louis, MO, USA.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.

Herbert, M., Caillas, C., Krotkov, E., Kweon, I. S., and Kanade, T. (1989). Terrain mapping for a roving planetary explorer. In *IEEE International Conference on Robotics and Automation*, pages 997–1002, Scottsdale, Arizona, USA.

Hert, S. and Schirra, S. (2012). 2d convex hulls and extreme points. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.1 edition.

Ho, K. and Newman, P. (2007). Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3):261–286.

Hokuyo (2009). UTM-30LX. `http://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html`. Retrieved April 22, 2013.

Holenstein, C., Zlot, R., and Bosse, M. (2011). Watertight surface reconstruction of caves from 3d laser data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3830–3837, San Francisco, CA, USA.

Honda (2013). Asimo. `http://asimo.honda.com/`. Retrieved April 22, 2013.

Horn, B. (1984). Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686.

Howard, A., Turmon, M., Matthies, L., Tang, B., Angelova, A., and Mjolsness, E. (2006). Towards learned traversability for robot navigation: From underfoot to the far field. *Journal of Field Robotics*, 23(11-12):1005–1017.

Huhle, B., Magnusson, M., Straer, W., and Lilienthal, A. J. (2008). Registration of colored 3d point clouds with a kernel-based extension to the normal distributions transform. In *IEEE International Conference on Robotics and Automation*, pages 4025–4030, Pasadena, California, USA.

Iagnemma, K. and Buehler, M. (2006a). Special issue on the darpa grand challenge, part i. *Journal of Field Robotics*, 23(8):461–652.

Iagnemma, K. and Buehler, M. (2006b). Special issue on the darpa grand challenge, part ii. *Journal of Field Robotics*, 23(9):655–835.

Johnson, A. (1997). *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Kaminade, T., Takubo, T., Mae, Y., and Arai, T. (2008). The generation of environmental map based on a ndt grid mapping–proposal of convergence calculation corresponding to high resolution grid-. In *IEEE International Conference on Robotics and Automation*, pages 1874–1879, Pasadena, California, USA.

Kaushik, R. and Xiao, J. (2012). Accelerated patch-based planar clustering of noisy range images in indoor environments for robot mapping. *Robotics and Autonomous Systems*, 60(4):584–598.

Kaushik, R., Xiao, J., Joseph, S., and Morris, W. (2010). Fast planar clustering and polygon extraction from noisy range images acquired in indoor environments. In *International Conference on Mechatronics and Automation*, pages 483–488, Xi'an, China.

Kohlhepp, P., Bretthauer, G., Walther, M., and Dillmann, R. (2006). Using orthogonal surface directions for autonomous 3d-exploration of indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3086–3092, Beijing, China.

Kohlhepp, P., Pozzo, P., Walther, M., and Dillmann, R. (2004). Sequential 3d-slam for mobile action planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 722–729, Sendai, Japan.

Konolige, K., Bowman, J., Chen, J., Mihelich, P., Calonder, M., Lepetit, V., and Fua, P. (2010). View-based maps. *The International Journal of Robotics Research*, 29(8):941–957.

Konolige, K., Fox, D., Limketkai, B., Ko, J., and Stewart, B. (2003). Map merging for distributed robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 212–217.

Leonard, J. and Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349.

Magnusson, M. (2009). *The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection.* PhD thesis, Örebro University. Örebro Studies in Technology 36.

Magnusson, M., Andreasson, H., Nüchter, A., and Lilienthal, A. J. (2009a). Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform. *Journal of Field Robotics*, 26(11-12):892–914.

Magnusson, M. and Duckett, T. (2005). A comparison of 3D registration algorithms for autonomous underground mining vehicles. In *Proceedings of the European Conference on Mobile Robotics (ECMR 2005)*, pages 86–91.

Magnusson, M., Lilienthal, A., and Duckett, T. (2007). Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827.

Magnusson, M., Nüchter, A., Lorken, C., Lilienthal, A., and Hertzberg, J. (2009b). Evaluation of 3d registration reliability and speed– a comparison of icp and ndt. In *IEEE International Conference on Robotics and Automation*, pages 3907–3912, Kobe, Japan.

Makadia, A., Patterson, A., IV, and Daniilidis, K. (2006). Fully automatic registration of 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1304, New York, NY, USA.

May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., and Hertzberg, J. (2009). Three-dimensional mapping with time-of-flight cameras. *Journal of Field Robotics*, 26(11-12):934–965.

Milford, M. and Wyeth, G. (2008). Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics*, 24(5):1038–1053.

Mobarhani, A., Nazari, S., Tamjidi, A. H., and Taghirad, H. (2011). Histogram based frontier exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1128–1133, San Francisco, CA, USA.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI National Conference on Artificial Intelligence*, Edmonton, Canada.

Nagatani, K., Matsuzawa, T., and Yoshida, K. (2010). Scan-point planning and 3-d map building for a 3-d laser range scanner in an outdoor environment. In Howard, A., Iagnemma, K., and Kelly, A., editors, *Field and Service Robotics*, volume 62 of *Springer Tracts in Advanced Robotics*, pages 207–217. Springer Berlin Heidelberg.

NASA (April 2, 2012b). Robonaut 2 team receives aiaa robotics award. `http://www.nasa.gov/mission_pages/station/main/robonaut.html`. Retrieved April 22, 2013.

NASA (August 6, 2012a). Curiosity: Nasa's next mars rover. `http://www.nasa.gov/mission_pages/msl/index.html`. Retrieved April 22, 2013.

Nguyen, V., Harati, A., and Siegwart, R. (2007). A lightweight slam algorithm using orthogonal planes for indoor mobile robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 658–663, San Diego, California, USA.

Nüchter, A. and Hertzberg, J. (2008). Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926.

Nüchter, A., Lingemann, K., and Hertzberg, J. (2006). Extracting drivable surfaces in outdoor 6d slam. In *International Symposium on Robotics and German Conference Robotik*, Munich, Germany.

Nüchter, A., Lingemann, K., and Hertzberg, J. (2007a). Cached kd tree search for icp algorithms. In *International Conference on 3D Digital Imaging and Modeling*, pages 419–426.

Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. (2007b). 6d slam — 3d mapping outdoor environments. *Journal of Field Robotics*, 24(8–9):699–722.

Nüchter, A., Surmann, H., and Hertzberg, J. (2003). Planning robot motion for 3d digitalization of indoor environments. In *International Conference on Advanced Robotics*, pages 222–227.

O'Rourke, J. (1987). *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford.

Pathak, K., Birk, A., Vaskevicius, N., Pfingsthorn, M., Schwertfeger, S., and Poppinga, J. (2010a). Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84.

Pathak, K., Birk, A., Vaskevicius, N., and Poppinga, J. (2010b). Fast registration based on noisy planes with unknown correspondences for 3-d mapping. *IEEE Transactions on Robotics*, 26(3):424–441.

Pathak, K., Borrmann, D., Elseberg, J., Vaskevicius, N., Birk, A., and Nüchter, A. (2010c). Evaluation of the robustness of planar-patches based 3d-registration using marker-based ground-truth in an outdoor urban scenario. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5725–5730, Taipei, Taiwan.

Paul, R., Triebel, R., Rus, D., and Newman, P. (2012). Semantic categorization of outdoor scenes with uncertainty estimates using multi-class gaussian process classification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2404–2410, Vilamoura, Algarve, Portugal.

Payeur, P., Hebert, P., Laurendeau, D., and Gosselin, C. M. (1997). Probabilistic octree modeling of a 3d dynamic environment. In *IEEE International Conference on Robotics and Automation*, pages 1289–1296.

Pomerleau, F., Liu, M., Colas, F., and Siegwart, R. (2012). Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14):1705–1711.

Poppinga, J. (2010). *Towards Autonomous Navigation for Robots with 3D Sensors*. PhD thesis, Jacobs University Bremen.

Poppinga, J., Birk, A., and Pathak, K. (2008a). Hough based terrain classification for realtime detection of drivable ground. *Journal of Field Robotics*, 25(1-2):67–88.

Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. (2008b). Fast plane detection and polygonalization in noisy 3D range images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3378–3383, Nice, France.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, Kobe, Japan.

Rabbani, T., Dijkman, S., van den Heuvel, F., and Vosselman, G. (2007). An integrated approach for modelling and global registration of point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(6):355– 370.

Rabbani, T. and van den Heuvel, F. (2005). Automatic point cloud registration using constrained search for corresponding objects. In *7th Conference on Optical 3-D Measurement Techniques*, pages 177–186, Vienna, Austria.

Ruhnke, M., Steder, B., Grisetti, G., and Burgard, W. (2010). Unsupervised learning of compact 3d models based on the detection of recurrent structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan.

Rusu, R., Marton, Z., Blodow, N., Holzbach, A., and Beetz, M. (2009a). Model-based and learned semantic object labeling in 3d point cloud maps of kitchen environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3601–3608, St. Louis, MO, USA.

Rusu, R. B., Blodow, N., and Beetz, M. (2009b). Fast point feature histograms (fpfh) for 3d registration. In *IEEE International Conference on Robotics and Automation*, Kobe, Japan.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation*, pages 1–4, Shanghai, China.

Ryde, J. and Hu, H. (2010). 3d mapping with multi-resolution occupied voxel lists. *Autonomous Robots*, 28(2):169–185.

Saeedi, S., Paull, L., Trentini, M., Seto, M., and Li, H. (2012a). Efficient map merging using a probabilistic generalized voronoi diagram. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4419–4424, Vilamoura, Algarve, Portugal.

Saeedi, S., Paull, L., Trentini, M., Seto, M., and Li, H. (2012b). Map merging using hough peak matching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4683–4688, Vilamoura, Algarve, Portugal.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Proceedings of Robotics: Science and Systems*, pages 26–27.

Shade, R. and Newman, P. (2011). Choosing where to go: Complete 3d exploration with stereo. In *IEEE International Conference on Robotics and Automation*, pages 2806–2811, Kobe, Japan.

Sprickerhof, J., Nüchter, A., Lingemann, K., and Hertzberg, J. (2011). A heuristic loop closing technique for large-scale 6d slam. *Automatika*, 52(3):199–222.

Steder, B., Grisetti, G., and Burgard, W. (2010). Robust place recognition for 3d range data based on point features. In *IEEE International Conference on Robotics and Automation*, pages 1400–1405, Anchorage, Alaska, USA.

Stelldinger, P. (2008). Topologically correct surface reconstruction using alpha shapes and relations to ball-pivoting. In *19th International Conference on Pattern Recognition*, pages 1–4, Tampa, Florida, USA.

Stoyanov, T. and Lilienthal, A. (2009). Maximum likelihood point cloud acquisition from a mobile platform. In *International Conference on Advanced Robotics*, pages 1–6.

Stoyanov, T., Magnusson, M., Almqvist, H., and Lilienthal, A. (2011). On the accuracy of the 3d normal distributions transform as a tool for spatial representation. In *IEEE International Conference on Robotics and Automation*, pages 4080–4085, Shanghai, China.

Stoyanov, T., Magnusson, M., Andreasson, H., and Lilienthal, A. J. (2010). Path planning in 3d environments using the normal distributions transform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3263–3268, Taipei, Taiwan.

Stoyanov, T., Magnusson, M., Andreasson, H., and Lilienthal, A. J. (2012). Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31(12):1377–1393.

Stoyanov, T., Magnusson, M., and Lilienthal, A. J. (2013). Comparative evaluation of the consistency of three-dimensional spatial representations used in autonomous robot navigation. *Journal of Field Robotics*, 30(2):216–236.

Takeuchi, E. and Tsubouchi, T. (2006). A 3-d scan matching using improved 3-d normal distributions transform for mobile robotic mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3068–3073, Beijing, China.

Thrun, S., Burgard, W., Fox, D., et al. (2005). *Probabilistic robotics*, volume 1. MIT press Cambridge.

Thrun, S. and Montemerlo, M. (2006). The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692.

Tong, C. H., Barfoot, T. D., and Dupuis, . (2012). Three-dimensional slam for mapping planetary work site environments. *Journal of Field Robotics*, 29(3):381–412.

TOSY (2013). TOPIO: TOSY pingpong playing robot. `http://topio.tosy.com/about.shtml?lang=en`. Retrieved April 22, 2013.

Trevor, A., Rogers, J., and Christensen, H. (2012). Planar surface slam with 3d and 2d sensors. In *IEEE International Conference on Robotics and Automation*, pages 3041–3048, St. Paul, Minnesota, USA.

Triebel, R., Pfaff, P., and Burgard, W. (2006). Multi-level surface maps for outdoor terrain mapping and loop closing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2276–2282, Beijing, China.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W. ., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

Valencia, R., Teniente, E., Trulls, E., and Andrade-Cetto, J. (2009). 3d mapping for urban service robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 303–308, St. Louis, MO, USA.

Vaskevicius, N., Birk, A., Pathak, K., and Schwertfeger, S. (2010). Efficient representation in 3d environment modeling for planetary robotic exploration. *Advanced Robotics*, 24(8-9):1169–1197.

Viejo, D. and Cazorla, M. (2007). 3d plane-based egomotion for slam on semi-structured environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2761–2766, San Diego, California, USA.

Vosselman, G., Dijkman, S., et al. (2001). 3d building model reconstruction from point clouds and ground plans. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/W4):37–44.

Weingarten, J. (2006). *Feature-based 3D SLAM*. PhD thesis, EPFL.

Weingarten, J. and Siegwart, R. (2005). Ekf-based 3d slam for structured environment reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3834–3839, Edmonton, Alberta, Canada.

Weingarten, J. and Siegwart, R. (2006). 3d slam using planar segments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3062–3067, Beijing, China.

Wikipedia (2013a). 2008 sichuan earthquake. `http://en.wikipedia.org/wiki/2008_Sichuan_earthquake`. Retrieved May 1, 2013.

Wikipedia (2013b). 2011 tōhoku earthquake and tsunami. `http://en.wikipedia.org/wiki/2011_T%C5%8Dhoku_earthquake_and_tsunami`. Retrieved May 1, 2013.

Wikipedia (2013c). 2013 lushan earthquake. `http://en.wikipedia.org/wiki/2013_Lushan_earthquake`. Retrieved May 1, 2013.

Woods Hole Oceanographic Institution (2013). Hybrid remotely operated vehicle nereus: Exploring the oceans' deepest depths. `http://www.whoi.edu/main/nereus`. Retrieved April 22, 2013.

Wulf, l. and Wagner, B. (2003). Fast 3d scanning methods for laser measurement systems. In *International Conference on Control Systems and Computer Science*, pages 312–317, Bucharest, Romania.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, Anchorage, Alaska, USA.

Xiao, J., Adler, B., and Zhang, H. (2012a). 3d point cloud registration based on planar surfaces. In *IEEE International Conference on Multisensor Fusion and Information Integration*, pages 40–45, Hamburg, Germany.

Xiao, J., Adler, B., Zhang, J., and Zhang, H. (2013). Planar segment based three-dimensional point cloud registration in outdoor environments. *Journal of Field Robotics*, pages n/a–n/a.

Xiao, J., Zhang, J., Zhang, J., Zhang, H., and Hildre, H. (2011). Fast plane detection for slam from noisy range images in both structured and unstructured environments. In *IEEE International Conference on Mechatronics and Automation*, pages 1768–1773, Beijing, China.

Xiao, J., Zhang, J. H., Adler, B., Zhang, H., and Zhang, J. W. (2012b). 3d point cloud plane segmentation for slam in structured and unstructured environments. *Robotics and Autonomous Systems*. Under review, available at `http://tams.informatik.uni-hamburg.de/people/xiao/publications/xiao_submitted_to_ras_2012.pdf`.

Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53, New York, NY, USA. ACM.

Yao, J., Taddei, P., Ruggeri, M. R., and Sequeira, V. (2011). Complex and photo-realistic scene representation based on range planar segmentation and model fusion. *The International Journal of Robotics Research*, 30(10):1263–1283.

Yvinec, M. (2012). 2d triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.1 edition.

Zhang, H., Zhang, H., Wang, X., Lu, H., Yang, S., Lu, S., Xiao, J., Sun, F., Hai, D., and Zheng, Z. (2009). Nubot team description paper 2009. In *RoboCup 2009*.

Zhang, J., Xiao, J., Zhang, J., Zhang, H., and Chen, S. (2011). Integrate multi-modal cues for category-independent object detection and localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 801–806, San Francisco, CA, USA.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152.