FEATURE-BASED 3D SLAM

THÈSE N^{O} 3601 (2006)

PRÉSENTÉE LE 6 SEPTEMBRE 2006
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
Laboratoire de systèmes autonomes 1
SECTION MICROTECHNIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jan WEINGARTEN

Diplom-Informatiker de l'Université de Karlsruhe, Allemagne
de nationalité allemande

acceptée sur proposition du jury:

Prof. H. Bleuler, président du jury
Prof. R. Siegwart, Dr G. Gruener, directeurs de thèse
Prof. B. Merminod, rapporteur
Prof. R. Dillmann, rapporteur
Prof. W. Burgard, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2006

Acknowledgements

I wish to express my gratitude to the following people: First of all, I would like to thank Prof. Roland Siegwart for offering me the possibility to work in his great team in a very pleasant and liberal atmosphere. The last four years at his lab were a great time and I especially want to express my appreciation for giving me the chance to attend several interesting conferences.

Secondly, I want to address special thanks to Dr. Gabriel Gruener, who not only was my thesis co-director, but also my direct supervisor during the last four years. His detailed and exact feedback was available at all times and very helpful for the different publications I wrote. I would further like to thank the CSEM Alpnach Dorf for kindly sponsoring the major part of this work and especially Dr. Alain Codourey and Dr. Ulrich Claessen for their confidence and support.

Thirdly, I would like to thank Prof. Rüdiger Dillmann, Prof. Wolfram Burgard and Prof. Bertrand Merminod for being part of my thesis committee.

Then I would like to thank my long-term office mates Dr. Kai "2d" Arras for his efforts to accommodate me with the subject in a musical way ("oh, mon vecteur d'état, comme t'es vertical, comme t'es beau mon gars...") and Dr. Agostino "mass..." - no, I won't write it out - Martinelli for the great time with him and also for providing the initial framework for the Piavino project which is progressing fantastically.

Thanks also to all the ASL team, especially Marie-Jo Pellaud, Daniel Burnier for the cables, Sascha Kolski for the many breaks on the lab's terrace and Dr. Björn Jensen for the 3D range scans.

Last but not least, I would like to thank all my friends and my family, especially my lovely wife Nicola for her love and constant support and my 10-month-old son Rafael for letting me sleep at least sometimes at night.

Kurzfassung

Diese Doktorarbeit befasst sich mit einem offenen Teilproblem der Roboternavigation, der zeitgleichen Lokalisierung und Umgebungskartographierung (SLAM) im dreidimensionalen Raum. Ziel hierbei ist es, ein System zu entwickeln, das es ermöglicht, einen mobilen Roboter zuverlässig im Raum zu lokalisieren und gleichzeitig die Umgebung des Roboters in Form einer 3D-Karte zu reproduzieren. Neben dem Ermöglichen von Roboternavigation im dreidimensionalen Raum ist eine solche Karte von grossem Nutzen für komplexere Roboteraufgaben wie Szeneninterpretation oder -manipulation sowie generell für Visualisierungen interessant, wie sie häufig im Vermessungswesen, der Architektur, für den Such- und Rettungsdienst bei Katastrophenbewältigung, o. ä. benötigt werden.

Dabei stellt die dritte Dimension eine grosse Herausforderung dar. Zuerst müssen Sensoren gefunden werden, die dreidimensionale Daten liefern und den Anforderungen der mobilen Robotik genügen, d.h. kompakt sind und wenig Strom verbrauchen. Für diese Arbeit wurden zwei Sensoren in Betracht gezogen, der Swiss Ranger des CSEM (Schweizer Zentrum für Elektronik und Mikrotechnik), sowie ein eigens entwickelter rotierender Tiefensensor, der auf einem kommerziell erhältlichen 2D-Laserscanner von SICK basiert. Beides sind aktive Sensoren und beruhen auf dem Prinzip der Laufzeitermittlung des Lichts. Nach der Sensorkalibrierung und Fehleranalyse (Kapitel 2) stellt sich heraus, dass der Swiss Ranger aufgrund seiner verrauschten Daten und seines eingeschränkten Sichtbereichs für Lokalisierungs- und Kartographierungszwecke im Vergleich zum rotierenden Laserscanner weniger gut geeignet ist. Deshalb wird er in folgenden Kapiteln nicht weiter berücksichtigt.

Da ein vom Sensor generierter 3D-Scan leicht aus vielen zehntausend Punkten bestehen kann und der Roboter während einer Mission Dutzende Scans erstellt, ist es notwendig, diese Rohdaten zu komprimieren, um sie effizient darstellen und verarbeiten zu können. Die dafür in dieser Arbeit gewählte Methode ist die Extraktion von Merkmalen, welches eine detailreiche und gleichzeitig kompakte, sowie aussagekräftige Umgebunsdarstellung ermöglicht. Die gewählten Merkmale sind planare Ebenensegmente, deren probabilistische Repräsentation und Extraktion in Kapitel 3 und 4 behandelt werden. Kapitel 5 beschreibt experimentelle Resultate des SLAM-Algorithmus, der mittels eines erweiterten Kalman Filters (EKF) die Lage der verschiedenen planaren Segmente unter Berücksichtigung der Unsicherheitsinformation, sowie die Roboterposition in inkrementeller Weise schätzen kann. Dies geschieht mittels des SPmodel (Symmetries and Perturbations Model), einer Methode zur Darstellung und Verarbeitung unsicherer geometrischer Modelle. Wie zahlreiche Resultate belegen, ermöglicht der entwickelte SLAM-Algorithmus die genaue Rekonstruktion der Robotertrajektorie und -umgebung im dreidimensionalen Raum.

Schlüsselwörter: Roboterlokalisierung und Umgebungskartographierung, Modellierung von 3D-Entfernungssensoren, Probabilistische Flächenextraktion, Flächensegmentierung, Kartographierung Strukturierter Umgebungen, 3D-SLAM mittels des Erweiterten Kalman Filters, SPmodel

Abstract

This doctoral thesis deals with an open subproblem of robot navigation, namely simultaneous localization and mapping (SLAM) in three-dimensional space. The goal is to develop a system which is capable of localizing a mobile robot in a reliable way and at the same time reconstruct its environment as a three-dimensional map. Besides enabling robot navigation in 3D, such a map could be of great importance for higher-level robotic tasks, like scene interpretation or manipulation as well as visualization purposes in general, which are required in surveying, architecture, urban search and rescue and others.

The third dimension is challenging. Firstly, sensors providing three-dimensional data have to be found which suit the requirements of mobile robots and therefore have to be limited in size and power consumption. For this work, two sensors have been considered: the Swiss Ranger from CSEM (Swiss Center for Electronics and Microtechnology) and a custom-built range sensor based on a commercially available 2D laser scanner from SICK. Both are active sensors relying on measuring the time-of-flight of the emitted light. After calibration and error analysis it was concluded that the Swiss Ranger is less suited for localization and mapping than the rotating laser scanner due to its noisy data and limited field of view. It was therefore not further considered in the ensuing work.

As a single 3D scan generated by the rotating laser scanner can be composed of many tens of thousands of data points and the robot takes dozens of scans during a mission, it is necessary to compress the raw data to visualize and process it efficiently. The method chosen in this work is to use a feature-based representation, which enables detailed and, at the same time, compact and informative environment reconstruction. The chosen features are planar segments, whose probabilistic representation and extraction are described, results of the SLAM algorithm are shown. With the aid of an Extended Kalman Filter (EKF) the pose of the robot and the location of the different planar segments - considering their uncertainty - can be estimated in an incremental way. The framework defined by the SPmodel (Symmetries and Perturbations Model) is used, allowing to represent and process various uncertain geometric models. The approach is validated through different experiments with a mobile robot in an office environment.

Keywords: Mobile Robot Localization and Mapping, 3D Range Sensor Modelling, Probabilistic Feature Extraction, Plane Segmentation, Structured Environment Mapping, Extended Kalman Filter, 3D SLAM, SPmodel

Contents

1 Introduction **1**

1.1 Mobile Robot Navigation 1

1.2 State-of-the-art 2

1.3 Goal 4

1.4 Organization 4

2 Sensor Modelling **7**

2.1 Introduction 7

2.1.1 Acquiring 3D Range Data 7

2.1.2 Calibration 9

2.2 Rotating Laser Scanner 9

2.2.1 Sensor Description 9

2.2.2 Calibration 10

2.3 The CSEM Swiss Ranger 15

2.3.1 Sensor Description 15

2.3.2 Intensity-based Calibration 19

2.3.3 Depth Calibration 22

2.4 Summary 24

3 Representation **27**

3.1 Introduction 27

3.2 Planar Features 28

3.2.1 Choosing the right plane model 28

3.3 The Symmetries and Perturbation Model (SPmodel) 30

3.3.1 Introduction 30

3.3.2 Describing a plane within the SPmodel 32

3.4 Probabilistic Planar Fitting 33

3.5 Planar Segments 36

3.6 Summary 37

4 Feature Extraction **39**
4.1 Introduction 39
4.2 Planar Segmentation 40
  4.2.1 Definition 40
  4.2.2 Related Work 42
  4.2.3 Taxonomy of Surface Segmentation Algorithms 43
4.3 The Grid-based Segmentation (GBS) Algorithm 46
  4.3.1 Algorithm Description 46
  4.3.2 Results 51
  4.3.3 Discussion 52
4.4 A Region-Growing Segmentation (RGS) Algorithm 52
  4.4.1 Algorithm Description 52
  4.4.2 Results 54
  4.4.3 Discussion 57
4.5 Post-Processing 58
4.6 Summary 61

5 SLAM **63**
5.1 Introduction 63
5.2 Related Work 64
5.3 The SPmap 65
5.4 Displacement of the Mobile Robot 66
5.5 Update 68
  5.5.1 Data Association 68
  5.5.2 Relating geometric entities 70
  5.5.3 Fusion 74
  5.5.4 Adding Non-Matched Features to the Map 74
  5.5.5 Centering 75
5.6 Summary 76

6 Experimental Results **79**
6.1 Introduction 79
6.2 Simulation 80
6.3 Using Infinite Planes 82
  6.3.1 Corridor 82
  6.3.2 A long corridor 83
  6.3.3 A ramp 83
  6.3.4 A Loop 83
  6.3.5 Discussion 83
6.4 Using Planar Segments 95
  6.4.1 Introduction 95

6.4.2 Incremental map building using planar segments 96
6.5 Discussion 96
6.6 Problematic Issues 101
6.7 Summary 102

7 Conclusions and Outlook 103

A Some Mathematical Background 107
A.1 Error Propagation 107
A.2 Transforming 3D points with Uncertainty 108
A.3 The Symmetries and Perturbations Model (SPmodel) 110
A.3.1 Introduction to Location Vectors and Homogeneous Matrices . . . 110
A.3.2 Operations in 3D 111
A.3.3 Jacobians of the Composition 112
A.3.4 Jacobians of the Inversion 113
A.3.5 Differential Transforms 113
A.3.6 Operations with SP-Locations 114

Chapter 1

Introduction

1.1 Mobile Robot Navigation

A fundamental issue in mobile robotics is navigation. Only a robot capable of navigating in a safe and reliable way can achieve true autonomy forming the basis for a vast number of potential new applications. These typically are service robots for the household and the elderly people, autonomous security and surveillance systems, automatic surveying robots for 3D reconstruction, automatically guided vehicle systems for logistics, mobile industrial robots, robots for urban-search-and-rescue or hazardous areas in general, space exploration robots etc.

From the robot's point of view, methods for navigation try to answer one or several of the following three questions:

1. Where am I?
   This question is addressed by localization algorithms aiming at estimating the pose of the mobile robot given a sequence of sensor measurements and an a priori map. If the latter is not available a priori, it has to be estimated at the same time while tracking the robot's pose, a process called simultaneous localization and mapping (SLAM) or concurrent mapping and localization (CML).

2. Where do I go?
   This question is addressed by path-planning and exploration algorithms, which are higher-level mechanisms telling the robot where to go next. If for example the goal of the robot is to create a complete map of an environment this algorithm will keep the robot heading for unvisited areas.

3. How do I get there?
   The final question is addressed by local path-planning and obstacle-avoidance algorithms which define how the robot moves from a point in space to another, while optimizing its trajectories w.r.t. to safety, efficiency, and so on.

This work falls into the first category, addressing the problem of localization and mapping. The aim is to estimate both the robot pose and the map of its environment given a sequence of measurements gathered by its proprioceptive and exteroceptive sensors. The importance of an underlying localization and mapping system cannot be overstated, it is fundamental for higher-level robotic tasks like scene exploration or manipulation.

1.2 State-of-the-art

State-of-the-art approaches for metric localization and mapping are probabilistic methods explicitly considering uncertainty information modelling sensor noise and imperfections in robot motion. They can be categorized w.r.t. the representation used as presented in table 1.1. The most important representations used are feature-based, grid-based or based on raw data.

For feature-based approaches the assumption is made that the physical environment can be modelled by geometric features. This is the case for a vast number of man-built environments like for example building interiors or cities which can be represented by a set of points, lines or planes. Due to the small quantity of data required to represent these features, the resulting maps are compact and the associated algorithms efficient in comparison to approaches using more memory like occupancy grids. On the other hand, they require a reliable feature extraction mechanism, which depending on the chosen feature type and quality of the sensor data can be a non-trivial task. Secondly, in feature-based approaches, the data association or correspondence problem has to be solved, which is the problem of finding features in scans taken from different locations corresponding to the same physical entity. As the true displacement of the robot between these locations is generally unknown - the wheel encoders merely provide an initial estimation - this can be difficult. Especially when closing a larger loop, the accumulated odometry errors lead to a considerable offset between newly observed features and old features contained in the map which has to be compensated.

The Extended Kalman Filter (EKF) is a widely used estimation tool applicable to feature-based localization and mapping. It has the advantage that it provides an analytical solution to the SLAM problem which leads to efficient algorithms with high accuracy. After initial work by [Smith et al., 1990], a variety of publications followed, for example [Leonard and Durrant-Whyte, 1991], [Castellanos et al., 1999], [Guivant et al., 2002] or [Arras et al., 2003]. As the complexity of the standard version of EKF-SLAM scales quadratically with the number of features, more efficient EKF-SLAM variants were developed [Guivant and Nebot, 2001a], allowing to use the EKF approach in real-time also for larger environments. Further disadvantages of the EKF-based SLAM approach are its restriction to Gaussian error distributions which have a single peak and therefore can represent only a single hypothesis about the robot pose at the same time. Elaborate extensions of the standard EKF approach address this issue by introducing multiple hypothesis (see [Jensfelt and Kristensen, 2001] or [Arras et al., 2003]), enabling global localization, which is the problem of localizing the robot given an a priori map without knowing its initial pose. Another disadvantage is that the EKF is a sub-optimal estimation technique making substantial use of linearization. This issue can for example be addressed by the Unscented Kalman Filter (UKF) introduced by [Julier and Uhlmann, 1997], who use a sample-based representation of the Gaussian distribution, leading to an improved estimation performance and easier implementation, than the standard EKF, as no Jacobians have to be calculated.

Another family of approaches uses a grid-based representation [Moravec, 1988], [Elfes, 1989b], [Fox et al., 1998], [Burgard et al., 1999]. This grid represents the belief about the robot pose as a probability distribution over all possible robot poses. Hence, any probability distribution can be represented in this discretized way, also multimodal distributions, required for global localization, which is not possible with the standard EKF approach. On the other hand, these so-called Markov localization approaches are not suited for SLAM, require a considerable amount of memory and are limited in precision due to their discrete nature.

Particle Filters [Gordon et al., 1993], [Fox et al., 2001] approximate the belief distribution of the robot pose by a set of particles which are not constrained to a grid. They can be used for global localization but also for SLAM, called FastSLAM as presented by [Montemerlo et al., 2002] in conjunction with a feature-based representation or a grid-based representation [Haehnel et al., 2003]. These approaches lead to efficient SLAM variants, as they can be tuned to scale linearly with an increasing number of map features, but on the other hand are sensitive to the so-called particle depletion problem [Van der Merwe et al., 2000], which is the problem of loosing the correct particle in a local loop for example [Stachniss et al., 2005]. Furthermore, their performance directly depends on the number of chosen particles.

Finally, scan-matching approaches were used successfully to address the SLAM problem. The idea is to align consecutive scans taken by the external sensors from the robot at different locations and thereby estimate its trajectory as well as create a consistent map. [Lu and Milios, 1994] presented a 2D version, [Besl and McKay, 1992] a version for 3D space, the popular Iterative Closest Point (ICP) algorithm. It looks for closest point pairs in two different scans and iteratively minimizes their relative transform. [Surmann et al., 2003] used the latter to create precise 3D maps using a mobile robot with a rotating laser scanner and also presented a method for generating globally consistent maps by a subsequent off-line refinement. As scan-matching is a tracking method, it is not suited for global localization, requiring the ability to model multimodal belief distributions of the robot pose. Note that up to now, all approaches successfully applied to 3D SLAM are based on the ICP algorithm.

1.3 Goal

Until now, SLAM was primarily addressed in 2D space. This restricts the use of such approaches to flat, non-overlapping environments and also limits the number of exploitation possibilities of the generated maps, as they represent only partial information of the three-dimensional world. A laser scanner mounted horizontally on the mobile robot for example can easily overlook table tops leading to a collision during navigation in the worst case. Higher-level tasks like scene manipulation, more sophisticated path-planning or robot navigation in non-flat terrain in general require an overarching three-dimensional representation.

The goal of this thesis was to address these issues by developing a feature-based SLAM approach for structured 3D environments and validate it experimentally. The features used are planar segments approximating underlying dense point clouds generated by a custom-built 3D laser scanner and a time-of-flight range camera. The main contributions are two newly developed feature segmentation algorithms capable of processing probabilistic data, a probabilistic method to fit planes to uncertain point data and the experimental validation of the EKF-SLAM algorithm extended to 3D space.

1.4 Organization

This report is organized as follows. In the next chapter, the two 3D sensors considered for this work are presented along with their calibration. The first is a custom-built rotating laser scanner and the second a time-of-flight camera developed by the Swiss Center of Electronics and Microtechnology (CSEM). In the ensuing chapter, the feature representation is presented which is based on planar segments represented within the SPmodel, modelled as probabilistic infinite planes with an associated set of polygons. These have to be extracted from the raw data, which is dealt with in chapter 4. The next chapter presents the developed SLAM approach followed by the chapter presenting the experimental results.

Method	Repres.	Dim.	Basic Concept	Pros	Cons	References
EKF	Geometric Features	2D	robot pose and features are stored and updated in a stochastic map, equivalent to a random variable with Gaussian distribution	analytical solution, therefore efficient, precise, compact and informative map	single-hypothesis in basic version, Gaussian assumption, linearization errors, data association	[Smith et al., 1990], [Leonard and Durrant-Whyte, 1991], [Castellanos et al., 1999], [Arras et al., 2003]
Markov	Occupancy Grid	2D	Robot state and environment are represented in a regular grid, representing probability distribution over all possible robot poses	multi-hypothesis, global localization possible	only localization, precision limited by cell size, high memory consumption	[Moravec, 1988], [Elfes, 1989b], [Fox et al., 1998], [Fox et al., 1999], [Burgard et al., 1999]
Particle Filters	Samples combined with grid / features	2D	a number of particles represents different robot trajectory and maps	efficient, intuitive implementation	precision depends on number of particles, particle depletion problem	[Gordon et al., 1993], [Fox et al., 2001],[Montemerlo et al., 2002], [Haehnel et al., 2003], [Montermerlo et al., 2003]
Scan-Matching	Raw Data	2D / 3D	scans are aligned by minimizing relative transforms between corresponding point pairs in an iterative way	data association solved automatically, no feature extraction	single hypothesis, map not directly globally consistent, map consists of raw data	[Lu and Milios, 1997], [Besl and McKay, 1992], [Surmann et al., 2003]

Table 1.1: Overview of some popular methods used for localization and mapping. Note that the majority of approaches operates in 2D space. Only the method mentioned last has been successfully applied to three-dimensional data sets.

Chapter 2

Sensor Modelling

2.1 Introduction

A fundamental requirement for the reliable functioning of a mobile robot is its ability to perceive its environment. This is achieved by using *exteroceptive* (or external) sensors, which in contrast to *proprioceptive* (or internal) sensors, do not measure internal states but the physical environment of the robot. This generally involves some kind of distance measurement, which is directly provided by the sensors used in this work, both relying on the time-of-flight measurement principle. The first is a custom-built range finder based on a commercially available 2D laser scanner LMS 200 developed by Sick and the second the Swiss Ranger 2b developed by the CSEM (Swiss Center for Electronics and Microtechnology), a new time-of-flight camera generating range images in real-time. Compared to other 3D sensors suitable for mobile robots, which are based on triangulation, like stereo vision or structured light cameras, these sensors provide distance measurements directly in a dense, regular form. They fulfill the requirements for mobile robots as they are limited in size, weight and power consumption.

After presenting related work, this chapter describes the two 3D sensors mentioned above, their underlying measurement principle and calibration.

2.1.1 Acquiring 3D Range Data

With the commercial availability of 2D laser scanners for over a decade now and many two-dimensional robotic problems solved, research has started to focus on 3D. In a first step, 3D data can be generated by sweeping a 2D laser sensor in perpendicular direction to its scanning plane through a scene and registering all scans, requiring precise knowledge of the scanner pose. The necessary hardware can be obtained by adding a vertically mounted 2D laser scanner to an existing mobile robot platform operating in two-dimensional space (as it is for example done in [Thrun et al., 2001], [Mahon and Williams, 2003], [Hähnel et al., 2003]). The existing localization system provides the poses in 2D used to register the scan data taken by the additional sensor in a consistent way. Figure 2.1 shows the Pygmalion robot of the Autonomous Systems Lab equipped with such a system and a resulting 3D map of a corridor. As presented in [Weingarten et al., 2004a], this leads to highly precise three-dimensional maps.

However, if the environment is not strictly flat, if it contains undulated areas, ramps or stairs these approaches can fail, as the robot pose can no more be represented in 2D space. Therefore, the natural next step is to localize the robot directly in 3D space. A single 2D laser scanner pointed upwards wouldn't provide the required information, hence a real 3D sensor is needed.

Another possibility to obtain 3D data is to rotate the 2D scanner with a stepping motor on a separate platform mounted onto the mobile robot. In the literature, two main design approaches of such systems can be found. Systems that are able to continuously rotate the laser scanner requiring slip rings and some ingenious design [Steinhaus and Dillmann, 2003], [Wulf and Wagner, 2003], or "nodding" laser scanners which are simpler to build [Surmann et al., 2001], [Matos et al., 2004] as no slip-rings are required. In this work a nodding system was built using a stepping motor rotating the sensor via a belt transmission.

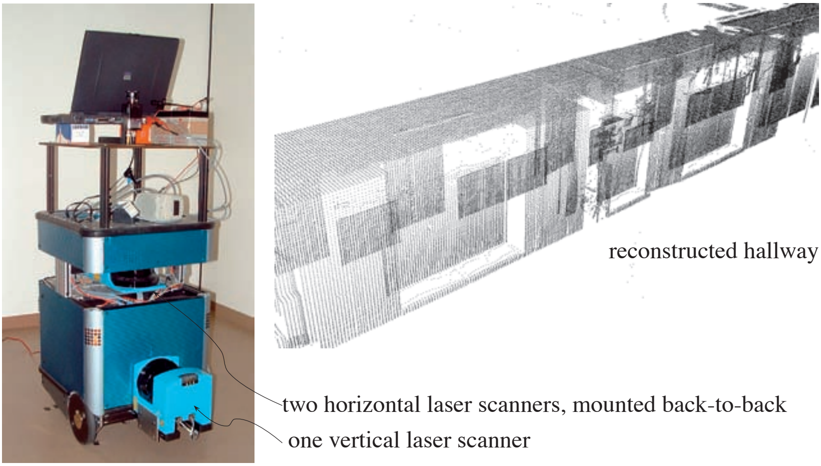

reconstructed hallway
two horizontal laser scanners, mounted back-to-back
one vertical laser scanner

Figure 2.1: In a first experiment, an additional vertically mounted laser scanner was added to the existing two horizontal laser scanners of the ”Pygmalion” robot. Capable of localizing itself in the (flat) office environment of our lab, this system is able to create precise 3D maps composed of vertical 2D slices. An example map (right) shows a part of a corridor of our lab.

A different possibility to generate 3D data is to directly use a dedicated 3D sensor. As already mentioned, the Swiss Center for Electronics and Microtechnology (CSEM) recently developed the so-called Swiss Ranger, a time-of-flight camera producing range images in real-time. It generates a 3D scan of the scene at once, up to 30 times per second, so the robot doesn't need to *stop-and-go*.

Both sensors are evaluated for this work which firstly requires their calibration, described in the following.

2.1.2 Calibration

Intrinsic calibration refers to the process of setting the magnitude of the output (or response) of a measuring instrument or sensor to the magnitude of the input property within specified accuracy and precision [Wikipedia, 2006]. This comprises the following:

Measurement Process	The description of the environmental conditions, the ground-truth used and the approach.
Mathematical Model	The development of a mathematical model for evaluation of the calibration considering all influencing systematic factors.
Error Analysis	The analysis of residual statistical errors.
Output	The output consisting of calibration values and their associated uncertainties (if available).

Extrinsic calibration refers to the process of relating the reference frame of the measurement instrument to another reference like the global coordinate frame. In more practical words, it is aimed at finding the location of the sensor coordinate frame with respect to some other reference frame. This is typically required in multi-sensor fusion were the data of different sensors has to be registered in a single coordinate frame.

2.2 Rotating Laser Scanner

2.2.1 Sensor Description

The custom-built 3D laser scanner is composed of the well-known LMS 200 laser scanner, developed by SICK, mounted on a rotating aluminium support, driven by a Nanotec PD4-I57 stepping motor and connected via a belt-transmission (see figure 2.2).

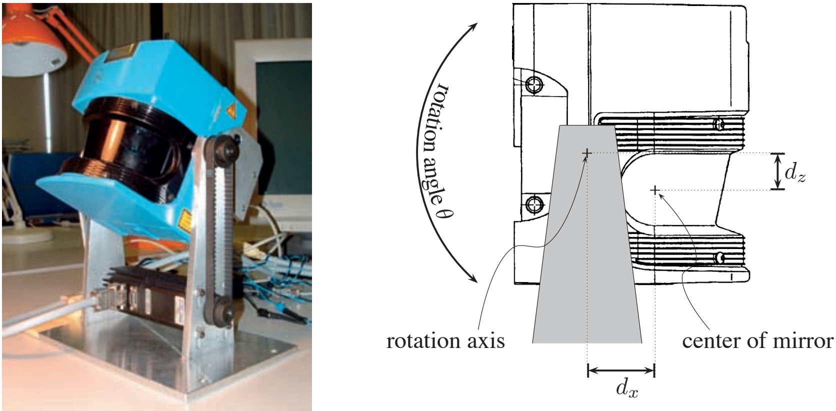


Figure 2.2: The photo on the left shows the 3D sensor built, composed of a Sick LMS 200 laser scanner mounted on a rotating support linked over a v-ribbed belt to the stepping motor. The schematic drawing on the right illustrates the calibration parameters, i. e. the translational offsets d_x and d_z between the center of the rotation axis and the center of the mirror wheel of the laser scanner, the origin of the 2D range measurements.

The angular resolution of the stepping motor is set to $0.45°$, the chosen number of steps is 601 with an angular range covering 270 degrees from $\theta^{min} = -45°$ to $\theta^{max} = 225°$ with respect to the horizontal plane. As the 2D laser scanner measures 361 distances per scan at an angular resolution of $0.5°$, the number of points of a complete 3D scan reaches $361 \times 601 = 216961$. The horizontal viewing angle is $(361 - 1) \cdot 0.5° = 180°$. With the current configuration, it takes around a minute for a complete scan which is slow enough to avoid synchronization problems between the laser scanner and the stepping motor. In a recent further development, the resolution of the 2D scanner was limited to $1.0°$ allowing a scanning rate of 75Hz for the 2D scans. This allowed to take a full 3D scan of in this case $181 \times 290 = 52490$ points in less than 4 seconds. Both configurations were used for experiments described in chapter 6. See table 2.1 for detailed specifications.

2.2.2 Calibration

Before calibrating the full 3D scanner system, the 2D laser scanner being rotated has to be calibrated. With the aid of related work [Reina and Gonzalez, 1997], [Diosi and Klemman, 2003], [Ye and Borenstein, 2002], it can be concluded that even though the SICK LMS 200 laser scanner tends to have a warming-up phase affecting measured distances, it still produces reliable measurements within the sub-centimeter range at varying tem-

Number of pixels	361 × 601 (configurable)
Depth resolution [mm]	approx. 1
Maximum range [m]	32
Horizontal field of view	180° or 100°
Vertical field of view	up to 330° (configurable)
Interface	RS232 / RS422
Dimensions [m]	0.32(W) × 0.4(H) × 0.19(D)
Weight [kg]	approx. 10
Power consumption [W]	< 40

Table 2.1: Specifications of the custom-built 3D laser scanner

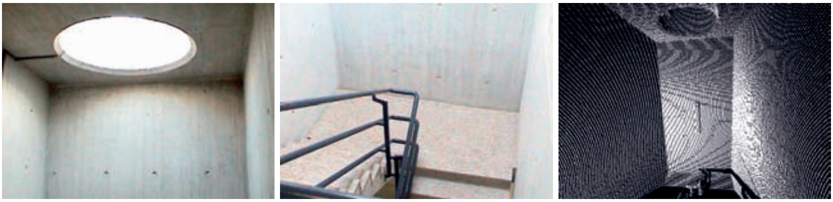


Figure 2.3: The scene chosen for calibration is the top floor of the stairway leading to our lab. It features a cuboidal structure which is easy to measure by hand. The right-most image shows a 3D scan of the scene, whereas the other two are photos. Note that the hole in the roof can be found again in figure 2.4.

peratures, on different materials and at different incidence angles. In the following, the fourfold calibration scheme mentioned above is applied to the rotating laser scanner.

Measurement Process

In order to test the performance and repeatability of the complete system, an environment composed of flat walls approximating a cuboid was chosen and measured by hand. This environment is then again measured by the rotating laser scanner resulting in a point cloud of 216916 data points (see right image of figure 2.3). After manual segmentation and least-square planar fitting (see next chapter), the resulting plane parameters can be used to evaluate the length, width and height of the room. Table 2.2 shows a quantitative comparison between these measured values and the ground-truth measured by hand.

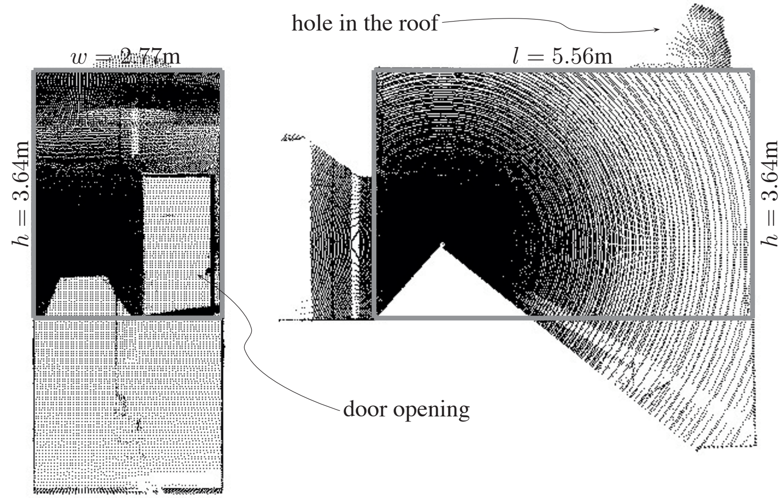


Figure 2.4: The calibration scene shown as 3D point cloud with the superimposed cuboid structure (depicted by gray rectangle), measured by hand. The left image represents the view from the front, the right image the view from the right side. Note that some features from figure 2.3, like the hole in the roof can be recognized.

	ground-truth [m]	measured by 3D scanner [m]
l	5.56	5.55
w	2.77	2.78
h	3.64	3.64

Table 2.2: Comparison of ground-truth measurements with measured values. The residual error is of an order of magnitude of a centimeter and is assumed to arise from material dependencies.

Mathematical Model

The mathematical model is the equation relating the raw distance measurements $\{\rho_{ij} | i \in [1;601], j \in [1;361]\}$ provided by the 2D laser scanner to the output 3D points $(x_{ij}, y_{ij}, z_{ij})^T$ in Cartesian space, including all calibration parameters. Viewed from the local coordinate frame S of the sensor located in the center of the rotating axis (see figure 2.2), it takes the following form (written in homogeneous coordinates):

$$\begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \\ 1 \end{bmatrix} = \mathbf{H}_S^{D_{ij}} \begin{bmatrix} \rho_{ij} \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.1a}$$

with

$$\mathbf{H}_S^{D_{ij}} = \begin{bmatrix} c_{\theta_i} c_{\phi_j} & -c_{\theta_i} s_{\phi_j} & s_{\theta_i} & c_{\theta_i} d_x + s_{\theta_i} d_z \\ s_{\phi_j} & c_{\phi_j} & 0 & 0 \\ -s_{\theta_i} c_{\phi_j} & s_{\theta_i} s_{\phi_j} & c_{\theta_i} & -s_{\theta_i} d_x + c_{\theta_i} d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.1b}$$

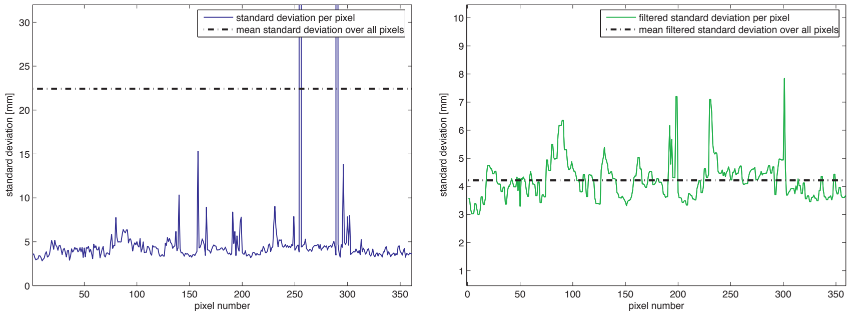
Note that c_{θ_i} stands for $\cos \theta_i$, s_{ϕ_j} for $\sin \phi_j$ etc. and $\theta_i = d_\theta - (i-1)t_\theta,\ (i = 1 \ldots 601)$ is the rotation angle depicted in figure 2.2, depending on the current rotation step number i, angular offset d_θ and angular step size t_θ. ϕ_j is the angle of the rotating mirror of the 2D laser at which the current measurement j is taken. It ranges from $\phi_1 = -\pi/2$ to $\phi_{361} = \pi/2$.

Error Analysis

To model the residual statistical error inherent to the 3D measurements, the following assumptions are made. Firstly, it is assumed that the strongest statistical error e_ρ stems from different surface characteristics of the scanned objects. This directly affects 2D range measurements $\hat{\rho}_{ij}$ along the laser beam. The errors arising from angular uncertainties are assumed to be negligible. The other error source involved is assumed to perturb angle $\hat{\theta}_i$ at which the 2D laser scanner is oriented and called e_θ.

The distributions chosen to model these errors are both Gaussian distributions, which can be justified by the central limit theorem of statistics [Bar-Shalom and Li, 1993]. It states that the sum of a set of independent random variables with arbitrary probability distributions and finite variance will approximately be normally distributed. Hence, the error model used is given by

$$\begin{aligned} \rho_{ij} &= \hat{\rho}_{ij} + e_\rho, \quad & e_\rho &\sim N(0, \sigma_\rho), \\ \theta_i &= \hat{\theta}_i + e_\theta, \quad & e_\theta &\sim N(0, \sigma_\theta). \end{aligned} \tag{2.2}$$



(a) Analysis of the statistical error using the raw data. Note that the highest standard deviation peak reaches 5238mm at pixel 255. These outliers are caused by edges and reflecting material which cannot be measured reliably, leading to a strongly biased average standard deviation over all pixels (dotted line).

(b) After filtering the outliers by sweeping a local window of 3 pixels over the data and calculating the median, these outliers can be suppressed. The resulting mean standard deviation lies between 4 and 5 mm.

Figure 2.5: The evaluation of the per pixel standard deviation σ_ρ of a sequence of 500 2D scans taken by the SICK LMS 200 laser scanner in a scene including edges and reflecting material. Graph (a) shows some strong peaks believed to be caused by edges which cannot be measured reliably. After filtering the outliers, an ambient noise of the order of magnitude of around 5 millimeters can be identified (b).

Calibration Output

Manual optimization using the cuboid environment mentioned above lead to the following settings used throughout this work:

$$\begin{aligned} d_x &= 0.035\text{m} \\ d_z &= -0.027\text{m} \\ d_\theta &= 45^\circ \\ t_\theta &= 0.45^\circ \\ \sigma_\rho &= 0.005\text{m} \\ \sigma_\theta &= 0.05^\circ \end{aligned} \tag{2.3}$$

σ_ρ was found by evaluating a hundred scans of a flat wall taken by the SICK LMS 200 laser scanner (see figure 2.5). σ_θ was manually set, as no experimental evaluation setup was found. $\sigma_\theta = 0.05^\circ$ turned out to be a value corresponding well to the measured data and the motor specifications.

Taking the above findings into account, probabilistic Cartesian 3D data can be generated by propagating errors from their sources - the 2D laser measurements ρ and the angle θ of the stepping motor - into the resulting Cartesian data. This can be achieved using standard error propagation techniques as described in the Appendix. An example 3D scan with depicted uncertainty ellipsoids is shown in figure 2.6.

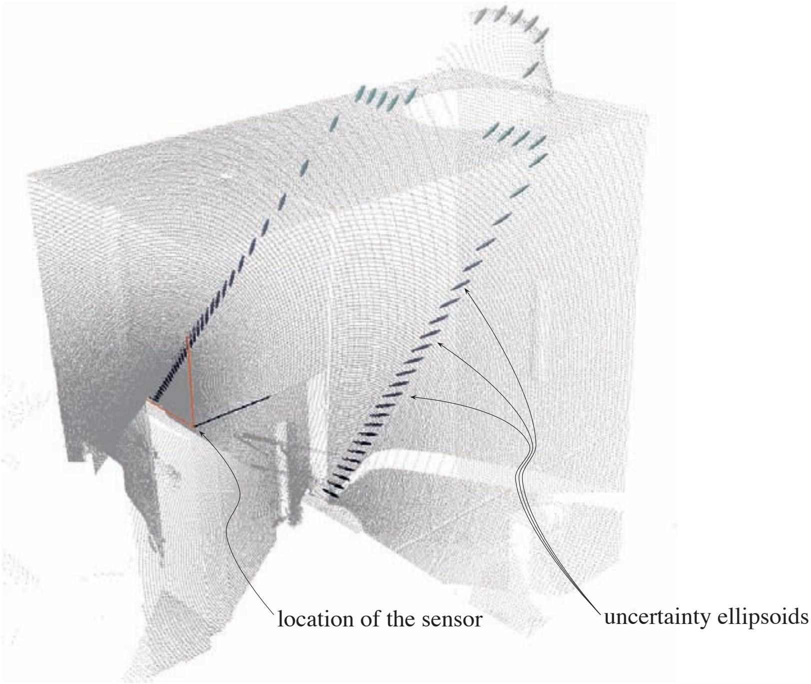


Figure 2.6: Visualization of the uncertainty of the raw data depicted in gray. Note that the uncertainty ellipsoids (10 times magnified) corresponding to every fifth data point of a single 2D scan are shown. The location of the 3D sensor is illustrated by the coordinate frame (x-axis is black).

2.3 The CSEM Swiss Ranger

2.3.1 Sensor Description

The Swiss Ranger is a novel, time-of-flight, solid-state imaging device that delivers distances as well as gray-level (i.e. intensity) images, developed by CSEM [Lange and Seitz,

Number of pixels	up to $160(\mathrm{h}) \times 124(\mathrm{v})$
Pixel Size	$39.2\mu\mathrm{m}(\mathrm{h}) \times 54.8\mu\mathrm{m}(\mathrm{v})$
Depth resolution [mm]	down to 5
Wavelength illumination [nm]	870
Illumination power [mW]	800 optical
Maximum range [m]	7.5
Frame rate [fps]	up to 30
Diagonal field of view (HxW)	$\pm 30°(\pm 21° \times \pm 23°)$
Interface	USB 2.0
Connector	Mini USB Type-B
Power supply	+ 12V / 1.5 A DC
Power consumption [W]	18 max.
Lens	f=8mm, $\mathrm{F}/\# = 1.4, \mathrm{M}12 \times 0.5$
Dimensions [mm]	$135(\mathrm{W}) \times 45(\mathrm{H}) \times 32(\mathrm{D})$
Weight [kg]	0.2

Table 2.3: The specifications of the CSEM Swiss Ranger 2b time-of-flight camera

2001]. It has been demonstrated that for a large range of illumination levels the range accuracy is essentially only limited by the shot noise of the available light [Oggier et al., 2004]. This allows to predict reliably the obtainable range resolution. In other words, for every measurement there is a reliable prediction of its resolution, which can be exploited by robot navigation algorithms. As the Swiss Ranger is a new sensor, its measurement principle will be explained in more detail in the following.

Measurement Principle

The camera is based on a 2-dimensional dedicated image sensor and a modulated light source. Every pixel on the sensor samples the amount of modulated light reflected by objects in the scene. This is done four times every period at equal intervals (see figure 2.7). Let these measurements be named m_1 to m_4. These four quantities allow to recover the sinusoidal incoming signal. The phase shift between the emitted light and the returning signal is given by

$$\varphi = \arctan\left(\frac{m_4 - m_2}{m_1 - m_3}\right) \tag{2.4}$$

and determines the distance to the objects in the scene:

$$L = L_{max}\frac{\varphi}{2\pi} \tag{2.5}$$

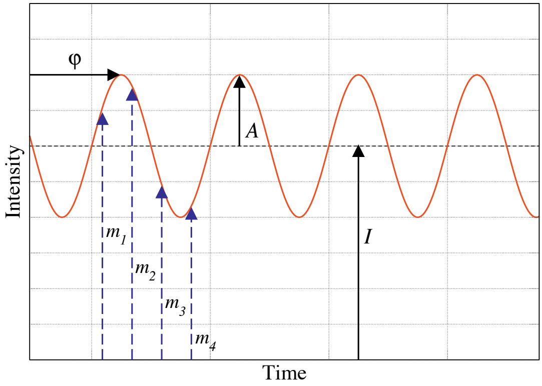


Figure 2.7: Measurement principle of the Swiss Ranger. Four intensity measurements (m_i) done at equal intervals each period allow to recover the measured modulated sinusoidal: the phase shift φ, the average intensity I, and the amplitude A.

L_{max} is the non-ambiguity range of the sensor, determined by the modulation frequency of the emitted light. The intensity of the objects in the image is recovered from the average light reflected:

$$I = \frac{m_1 + m_2 + m_3 + m_4}{4} \tag{2.6}$$

The amplitude of the measured sinusoidal

$$A = \frac{\sqrt{(m_3 - m_1)^2 + (m_4 - m_2)^2}}{2} \tag{2.7}$$

allows to predict the quality of the measurement:

$$\Delta L = \frac{L_{max}}{\sqrt{8}} \frac{\sqrt{I}}{2A} \tag{2.8}$$

Implementation

The model 2b of the Swiss Ranger's image sensor has been implemented on 0.8 μm CMOS / BCCD technology. It contains 160 by 124 pixels, each pixel being 39.2 μm wide and 54.8 μm high. The emitted light modulation frequency is typically 20 MHz, yielding a non-ambiguity range of 7.5m. The modulated illumination is generated by a set of 48 near-infrared LEDs. The field of view depends on the lense used, which in our case is about 43° (horizontally) and 46° (vertically), implying an angular resolution of 0.35° in the worst case. Figure 2.8 shows the Swiss Ranger 2b, used in this work. As sunlight contains near-infrared light it can interfere with the received reflected sensor light beam dramatically decreasing the measurement performance. A recently developed new version of the Swiss Ranger (CSEM Swiss Ranger 3000) was developed specifically to address this issue and should perform better in this respect. Note that the sensor contains no moving parts. The Swiss Ranger 2b was available as an evaluation prototype. It includes an FPGA that recovers the data from the sensor and applies equations (2.4) to (2.7). A USB interface is used to talk with the camera from a client. After the client requests a picture, an array of pixels representing the measured (and calculated) distance and intensity is returned. Several parameters can be adjusted by the client through the USB interface, like L_{max}, data filtering by amplitude threshold, etc. Note that in its current implementation, the FPGA does not include a mode that allows the client to recover the raw measurements

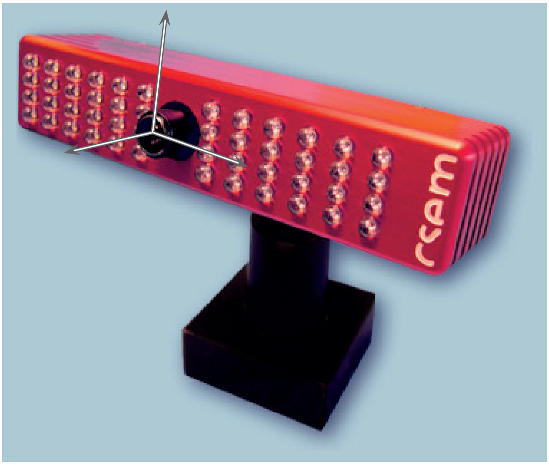


Figure 2.8: The time-of-flight Swiss Ranger 2b sensor from CSEM. The 48 LEDs mounted on the front emit infrared light at a modulation frequency of 20 MHz. The returning reflected part is captured by the central lens and refracted on the CMOS sensor (with 160 × 124 pixels), capable of determining the phase difference between emitted and received light which is proportional to the object distance. The superimposed coordinate system used for calibration is right-handed (y-axis is pointing upwards).

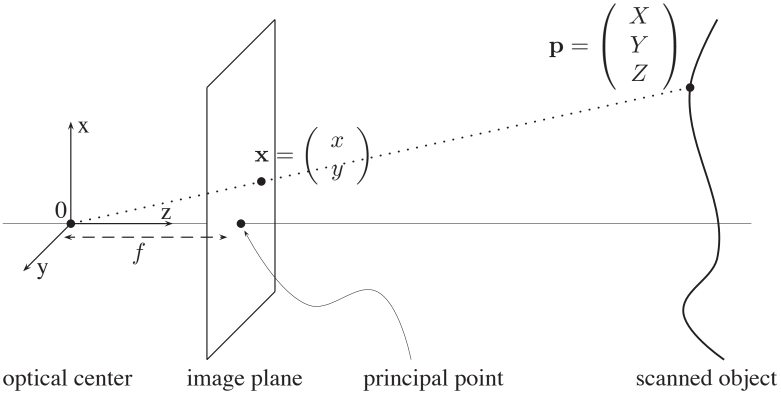

$\mathbf{p} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$
x
$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$
0
z
y
f
optical center
image plane
principal point
scanned object

Figure 2.9: The (idealized) frontal pinhole imaging model: the image $\mathbf{x}$ of a point $\mathbf{p} = (X, Y, Z)^T$ is the intersection of the ray going through the optical center 0 and the image plane, located at focal distance f in front of the origin. Note that equation 2.9 can easily be verified.

m_i.

2.3.2 Intensity-based Calibration

In order to calibrate the CSEM Swiss Ranger time-of-flight camera, a two-stage procedure was adopted. As the sensor features a standard optical lens refracting the reflected light onto the imaging chip, distortions and misalignments can be compensated using common video camera calibration techniques which is described in the following.

Measurement Process

In this work, the freely available Matlab Camera Calibration Toolbox developed by [Bouguet, 2000], based on work by [Tsai, 1992], [Zhang, 1999] and [Heikkila and Silven, 1997], was used, relying on the frontal pinhole imaging model depicted in figure 2.9. The fundamental law governing subsequent equations is the theorem of intersecting lines applied to the pinhole model:

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z} \tag{2.9}$$

In order to be able to find all necessary calibration parameters, a set of images representing a chessboard of known dimension is required. The various corners of the black fields are used as input points for the iterative calibration procedure. The equations involved are presented in the following.

Mathematical Model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_d \\ 1 \end{bmatrix} \tag{2.10}$$

with pixel coordinates (u, v) originating at the top left of the image plane. s_x and s_y is the size in meters of a single pixel along the x-axis and y-axis, respectively. s_θ is a skew factor[1] modelling the angle between the x- and y-axis, f is the focal length of the lens. The 5-vector $\kappa = [\kappa_1, \kappa_2, \kappa_3, \kappa_4, \kappa_5]^T$ contains both radial and tangential (represented by δ) distortion coefficients. The corrected point coordinates $\mathbf{x}_d$ are related to the input (normalized) point coordinates $\mathbf{x}_n$ by the following equations:

$$\mathbf{x}_d = \left(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_5 r^6\right) \mathbf{x}_n + \delta \tag{2.11a}$$

and

$$\delta = \begin{bmatrix} 2\kappa_3 xy + \kappa_4(r^2 + 2x^2) \\ \kappa_3(r^2 + 2y^2) + 2\kappa_4 xy \end{bmatrix} \tag{2.11b}$$

where

$$r^2 = x^2 + y^2, \tag{2.11c}$$

$$\mathbf{x}_n = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{Z_C} \begin{bmatrix} X_C \\ Y_C \end{bmatrix}. \tag{2.11d}$$

All parameters mentioned so far are referred to as *intrinsic* parameters. Note that the scanned point represented w.r.t. to the camera frame C is given by $[X_C, Y_C, Z_C]$. It is related to the same point $[X_0, Y_0, Z_0]$ with respect to the global reference W through a rigid-body motion represented by a homogeneous transform $\mathbf{H}_{CW}$:

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix}}_{\mathbf{H}_{CW}} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{2.12}$$

---

[1] not to be confused with $\sin\theta$

The rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and the translation $\mathbf{T} \in \mathbb{R}^{3\times1}$ are the so-called *extrinsic* parameters.

Due to the complexity of the involved terms modelling tangential and radial distortions, an analytical expression for computing the normalized image coordinates $x_n = [x, y]^T$ from the pixel coordinates (u, v) does not exist. However, the toolbox provides a numerical implementation, the *normalize*-function, allowing to compute a priori all normalized images coordinates $\mathbf{x}_n = [x, y]^T$ for input pixel coordinates (u, v). For the implementation in this work, they have been stored in a lookup table. These normalized image coordinates can then be used with the measured distance ρ to reconstruct a point $[X_C, Y_C, Z_C]$ in 3D:

$$X_C = xZ_C \tag{2.13a}$$

$$Y_C = yZ_C \tag{2.13b}$$

$$Z_C = \cos(\phi_H)\cos(\phi_V)\rho \tag{2.13c}$$

with

$$\phi_H = \arctan(x) \tag{2.13d}$$

$$\phi_V = \arctan(y). \tag{2.13e}$$

Calibration Output

The Matlab Calibration Toolbox provides versatile means to recover the intrinsic and extrinsic parameters from a sequence of pictures taken from different viewpoints of a chessboard with known dimensions. In this case, it produced the following output:

```
Focal Length:     fc = [ 204.22773   145.37965 ]
                  ± [ 4.44863   3.14147 ]
Principal point: cc = [ 83.05954   61.32086 ]
                  ± [ 5.51981   4.60090 ]
Skew:             alpha_c = [ 0.00000 ]±[0.00000 ]
                  => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:       kc = [ -0.11358   -0.01498   0.00051   0.00101 0.00000 ]
                  ± [ 0.05126   0.15562   0.00577   0.00440   0.00000 ]
Pixel error:      err = [ 0.26284   0.26848 ]
```

Here, the focal length `fc` is stored in pixels and can be related to the focal length f in meters by using the size of a single pixel $(s_x, s_y) = (39.2\mu m, 54.8\mu m)$ taken from the sensor specification sheet. Note that the estimated focal length corresponds with the data from the objective manufacturer (see table 2.3):

$$\begin{aligned} f &= 204.2 \cdot 39.2\mu m \approx 8.0mm \\ &= 145.4 \cdot 54.8\mu m \approx 8.0mm \end{aligned}$$

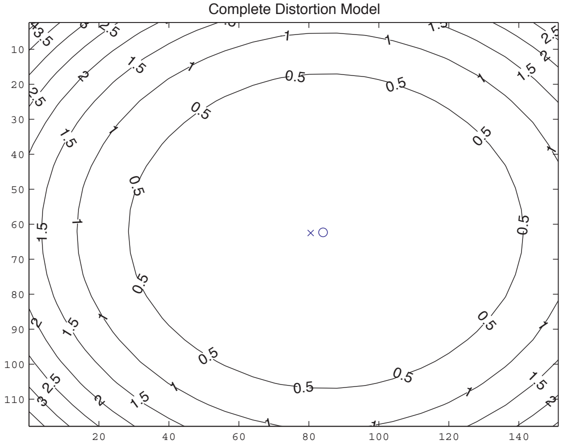


Figure 2.10: The distortion model of the CSEM Swiss Ranger optics in [pixel], output by the Matlab Calibration Toolbox.

Vector `cc` corresponds to the coordinates of the principal point (o_x, o_y). `alpha_c` is the skew factor s_θ mentioned above and `kc` contains radial and tangential distortion coefficients[2]. See figure 2.10 for a visualization of the distortion model. Note that a substantial distortion can be observed leading to an offset of several pixels in the corners.

2.3.3 Depth Calibration

Measurement Process

A test environment using a planar wall[3] located at a fixed distance (1.4m) perpendicular to the optical axis of the sensor was used to perform the tests. While this is certainly not representative for all measurement situations, it has the advantage that the ground truth represented by a wall is precisely known and only depends on the distance from the sensor.

As described above, the sensor performs a time-of-flight measurement which detects a phase shift in the modulated emitted signal (2.4) which can in turn be translated into a distance (2.5). In practice, due to propagation delay in the driving circuits of the camera,

---

[2] `kc` corresponds to the 5-vector κ mentioned above

[3] of a flat, painted, metallic material with a slightly increased reflectivity in the orange color

a distance offset has to be included[4]. The offset is determined experimentally during calibration and can be set by the user on the Swiss Ranger configuration registers.

Mathematical Model

The precision of the sensor can be quantified by comparing the distance measured by the Swiss Ranger with the ground truth for several distances. Using a planar surface for calibration simplifies this process, since the expected range can be easily calculated for all 19840 pixels. It was observed that the distance vs. phase offset relation did not hold, as (2.5) models: calibrating the offset with a close-by reference wall would not hold for far references. Nevertheless, let us remember that the actual raw data is the 4 m_i discussed above. A direct calibration would be based on these quantities and not on the results of (2.4) and (2.5). Nonetheless, our evaluation prototype delivered only distance and intensity directly. In order to calibrate the distance measurement under the given conditions, the following empirical relation is proposed:

$$L = k(L_{max}\frac{\varphi}{2\pi} + L_0) \tag{2.14}$$

where L_0 is the distance offset and k is a linearization factor. Through calibration the following values were found: $L_0 = 0.6$ m, $k = 1.18$.

Error Analysis

To get an idea of the statistical spread of a data set taken in an office environment, one thousand consecutive measurements[5] were taken of the same reference surface (see figure 2.11). The resulting standard deviation, averaged over all 19840 pixels, is $\sigma = 0.009$m. Note that only the accuracy in the direction of the received light beams is considered, which changes with varying distances, used materials, etc. Also note that this definition of accuracy (averaged over all pixels) differs from the camera's specifications, which is done for one pixel alone.

---

[4] a delay of 1 ns already implies an offset of about 0.15 m

[5] Sensor configuration used: long integration time $t_{long} = 48 * 256\mu s$, short integration time $t_{short} = 16 * 256\mu s$

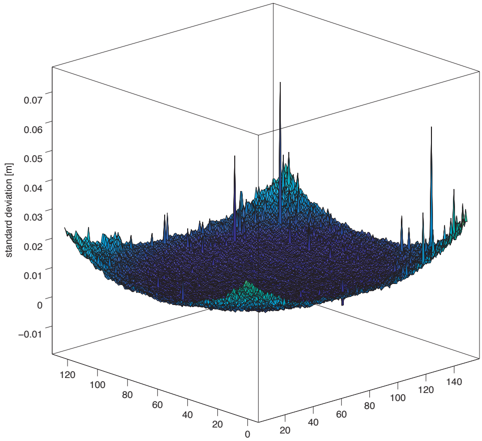


Figure 2.11: Per pixel error analysis of the CSEM Swiss Ranger based on 1000 scans of a flat wall. The raw measurements plotted in polar coordinates lead to the curved shape. Note that, except for some pixels which lead to a standard deviation of several centimeters, the average σ is below a centimeter.

Figure 2.12 shows an example scan of the Swiss Ranger using the above calibration with associated 3D reconstruction in Cartesian space. Note that a substantial amount of systematic errors remains. The reconstructed wall in the background makes this especially clear, as is not flat as expected.

These findings are fortified by [Kahlmann and Ingensand, 2004], who recently presented a thorough sensor analysis of the Swiss Ranger. They conclude that the Swiss Ranger is a very interesting measurement system but needs some improvement, namely a better lens with less distortions and a single and more powerful light source.

2.4 Summary

This chapter presented two 3D sensors usable for mobile robots and their calibration. Firstly a custom-built 3D laser scanner based on a commercially available 2D laser scanner rotated by a stepping motor and secondly the CSEM Swiss Ranger. Even though the latter is able to deliver range images at frequencies of up to 30 fps which is two orders of magnitude faster than the former, it is believed that due to its limitations shown above it is not well-suited for generating precise, metric maps. These limitations are its restricted field-of-view of around 45°, its strong dependency on the material of the scanned surface as well as the incidence angle, leading to distorted scans. However, as shown in [Weingarten et al., 2004b], it provides very useful information for obstacle avoidance purposes.

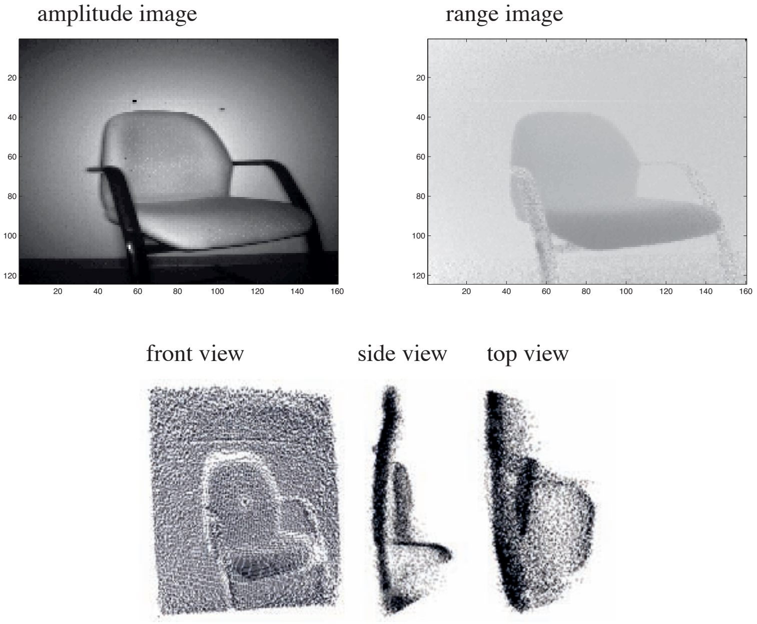


Figure 2.12: An example scan showing an office chair. The top left image shows the returned amplitude, the top right the range image, whereas the 3 images below show different views of the reconstruction in cartesian space. The left image shows the front view, the middle image the view from the left and the right image the top view. Note that the wall behind the chair should be flat.

Chapter 3

Representation

3.1 Introduction

In order to achieve true autonomy for a mobile robot, an internal representation of its physical environment is required. The sensor data, the robot gathers with its exteroceptive sensors, has to be translated into an appropriate representation that allows the robot to find and keep its way along its mission. Used representations range from occupancy grid maps [Moravec and Elfes, 1985], [Elfes, 1989a], [Dellaert et al., 1999], [Fox et al., 1999] to maps based on geometric features like points [Durrant-Whyte, 1996], [Dissanayake et al., 2001], lines [Castellanos and Tardós, 1999], [Arras et al., 2003] or planes [Horn and Schmidt, 1995a], [Kohlhepp et al., 2004]. Beside these so-called metric maps, topological maps are also used, representing the topology of space rather than its physical shape [Tomatis et al., 2003].

Depending on the application, different map types are appropriate. When working in structured environments like building interiors or cities, it can be advantageous to use this a priori information for the following reasons: Firstly, the generated feature-based maps are compact as they consist of a set of features, which can be described by a few parameters only. On the other hand, grid-based representations are a discretization of the space generally requiring large amounts of memory. Secondly, feature-based approaches can lead to more precise maps and robot localization as no space discretization occurs. A space decomposed into an occupancy grid does not include information falling bellow the size of a grid cell which in general is of an order of magnitude of around a decimeter. Thirdly, a robot extracting and analyzing features from its environment actually learns these features from its environment. In later processing steps, these features can be used to gather higher-level knowledge about the robot's environment. A combination of several orthogonal planar features may form a room, e.g., or planar features with a certain surface area and orientation could be interpreted as doors, etc. Finally, as sensor data always is noisy, extracting features represents a way of filtering noise. A plane composed of thousand data points is likely to actually exist whereas a single data point could be an outlier. The main disadvantage of a feature-based approach is that depending on the feature type used, a big effort has to be summoned in order to extract this feature in a robust way. This is especially difficult with noisy sensor data containing irregularities and outliers. This chapter describes how the chosen features can be described mathematically.

3.2 Planar Features

In this work, planar segments are used to represent the environment of the robot. A planar segment is firstly composed of an infinite plane described in general by the following equation:

$$Ax + By + Cz + D = 0, \tag{3.1}$$

where A, B, C, D are the plane parameters and (x, y, z) the coordinates of a 3D point lying in the plane. Secondly, it consists of a set of supporting points, defining the spacial boundaries of the plane in two dimensions. Before delving into the representation used for the segment information, a survey of planar models used in the literature is presented in the following.

3.2.1 Choosing the right plane model

Different plane models can be found in the literature (see table 3.1). Not all of them are well-suited for least-square fitting problems and error analysis as some have singularities. The most popular plane model is the Hessian Normal Form (see model 2 of table 3.1)

$$\mathbf{n}\mathbf{p}_i - d \equiv n_x x_i + n_y y_i + n_z z_i - d = 0, \tag{3.2}$$

which is equivalent to model 1. All data points $\mathbf{p}_i = (x_i, y_i, z_i)^T$ that lie on the plane defined by the unit normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ and the perpendicular distance to the origin d satisfy equation (3.2). In reality however, the points $\mathbf{p}_i$ rarely lie exactly on the plane, hence the value ϵ_i is introduced on the right of (3.2) standing for the fitting error, which corresponds to the perpendicular distance to the plane:

$$n_x x_i + n_y y_i + n_z z_i - d = \epsilon_i \tag{3.3}$$

Taking the sum over all squared error distances yields the regression problem

$$R(n_x, n_y, n_z, d) = \sum_{i=0}^{N} \epsilon^2 = \sum_{i=0}^{N} (n_x x_i + n_y y_i + n_z z_i - d)^2 \tag{3.4}$$

#	Equation	Plane Parameters
1	Ax + By + Cz + D = 0	A,B,C,D
2	$\mathbf{n}\mathbf{x} - d = 0$	$\mathbf{n} = (n_x, n_y, n_z)^T$, $\|\mathbf{n}\| = 1$, d
3	$x \cos\theta \cos\varphi + y \cos\theta \sin\varphi + z \sin\theta - \rho = 0$	θ, φ, ρ
4	$Z = aX + bY + d$	a, b, d
5	$ax + by + cz + 1 = 0$	a, b, c
6	$\nu = \frac{1}{\sqrt{1+d^2}} \begin{pmatrix} \mathbf{n} \\ d \end{pmatrix}$	$\nu = (\nu_1, \nu_2, \nu_3, \nu_4)^T$, $\|\nu\| = 1$

Table 3.1: Different plane models found in the literature. Note that the most general model 1 has four parameters, although three parameters are sufficient to describe a plane (see model 3). Model 2 is the popular Hesse notation formed by a unit normal vector $\mathbf{n}$ and the orthogonal distance to the origin d. Model 4 and 5 both have singularities and are therefore not to be used without restrictions. Model 6 [Kanatani, 1996] provides a way to include all plane parameters in single unit vector. Depending on the application, different plane models are appropriate. In this work, model 2 is used for evaluating the plane parameters and model 4 for error propagation.

used again in section 3.4. Dividing (3.3) by $(-d)$ and substituting accordingly yields model 5:

$$ax_i + by_i + cz_i + 1 = \frac{\epsilon_i}{-d} \tag{3.5}$$

The associated regression problem does not minimize the sum of the squared distances but $\sum_{i=0}^{N} \left(\epsilon_i/(-d)\right)^2$. Furthermore, if $(x_i, y_i, z_i) = (0, 0, 0)$, the plane parameters are undefined. Model 4 is found dividing (3.3) by n_z and substituting again, yielding

$$Z - aX - bY - d = \frac{\epsilon_i}{-n_z}. \tag{3.6}$$

It can be observed, that in the related regression problem the sum to be minimized is $\sum_{i=0}^{N} \left(\epsilon_i/(-n_z)\right)^2$. This corresponds to the orthogonal least-square distance when $n_z = \pm 1$ only. Here, the singularity is reached with a vertical plane ($n_z = 0$). As model 4 of table 3.1 can be conveniently solved for the model parameters analytically, it will be used for error propagation as described below.

Even though model 3 is the minimal plane model without singularities, its corresponding nonlinear regression problem is difficult to tackle analytically and therefore avoided. However, due to its minimality, it is the best choice for a 3D Hough Transform [Hough, 1959] for example, where the lower number of parameters reduces the computational cost.

Model 6 in table of 3.1 was introduced by [Kanatani, 1996] and is not further considered here, as it is not intuitive and leads to singular covariance matrices which are complicated to deal with.

In this work, a combination of the Hessian Normal Form and model 4 was chosen for extracting planes from 3D point clouds and calculating the associated uncertainty. This procedure is thoroughly described in section 3.4. The next section describes how the infinite plane extracted using the Hessian Normal Form is represented within the scope of robot localization and mapping. The SPmodel [Castellanos and Tardós, 1999] is used, which allows to represent and process a multitude of geometric features with associated uncertainties.

3.3 The Symmetries and Perturbation Model (SPmodel)

3.3.1 Introduction

The Symmetries and Perturbations Model (SPmodel) is a framework for representing and processing uncertain geometrical data and was introduced by Tardós, Castellanos et al. [Tardós, 1992], [Castellanos et al., 1999], [Castellanos and Tardós, 1999].

Within this framework, the location $\mathbf{L}_{WF}$ of a geometrical object F is defined by four parameters: The location vector $\mathbf{x}_{WF} = (x, y, z, \phi, \theta, \psi)^T$ [1] from the world coordinate frame W into the local object coordinate frame F, the self-binding matrix $\mathbf{B}_F$ accounting for symmetries, the locally defined perturbation vector[2] $\mathbf{p}_F$ representing the error with its associated covariance matrix $\mathbf{C}_F$ encoding the uncertainty information. A complete feature within the SPmodel is then given by quadruple called location

$$\mathbf{L}_{WF} = (\hat{\mathbf{x}}_{WF}, \hat{\mathbf{p}}_F, \mathbf{B}_F, \mathbf{C}_F). \tag{3.7}$$

The symmetries of a geometric object are the set of transforms preserving it w.r.t. its locally attached reference frame. A point in 3D space for example can be arbitrarily rotated around itself, hence its symmetries are the set of all rotations $\{R_\phi, R_\theta, R_\psi\}$ around that point. An infinite plane in 3D space can be rotated around its normal vector and translated along its x- and y-axes lying in the plane. Hence, its symmetries have three degrees of freedom, one rotational and two translational. The self-binding matrix $\mathbf{B}_F$ encodes these symmetries by selecting those components of the full-rank differential vector $\mathbf{d}_F = (d_x, d_y, d_z, d_\phi, d_\theta, d_\psi)^T$ which actually represent an effective error. In the case of

---

[1] The angles ϕ, θ, ψ represent the RPY-rotations (roll, pitch, yaw) around the global z-axis, the y-axis and the x-axis, respectively

[2] The perturbation vector $\mathbf{p}_F$ is formed by multiplying the binding matrix $\mathbf{B}_F$ with the full rank differential vector $\mathbf{d}_F = (d_x, d_y, d_z, d_\phi, d_\theta, d_\psi)^T$

the plane, the binding matrix $\mathbf{B}_F$ becomes

$$\mathbf{B}_F = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.8}$$

The perturbation vector $\mathbf{p}_F$ represents these effective errors and is obtained by multiplying the binding matrix with the full-rank differential vector:

$$\mathbf{p}_F = \mathbf{B}_F \mathbf{d}_F. \tag{3.9}$$

Its covariance is given by

$$\mathbf{C}_F = E[(\mathbf{p}_F - \hat{\mathbf{p}}_F)(\mathbf{p}_F - \hat{\mathbf{p}}_F)^T], \tag{3.10}$$

with

$$\hat{\mathbf{p}}_F = E[\mathbf{p}_F]. \tag{3.11}$$

Table 3.2 shows more example features represented within the SPmodel framework. Note that the feature type is defined by the binding matrices.

Geometric Object	Binding Matrix $\mathbf{B}$	Perturbation Vector
Point	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$\mathbf{p} = (d_x, d_y, d_z)^T$
Infinite Plane	$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$\mathbf{p} = (d_z, d_\theta, d_\psi)^T$
Infinite Line	$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	$\mathbf{p} = (d_y, d_z, d_\phi, d_\theta)^T$
Full 3D Location	$\mathbf{I}_6$	$\mathbf{p} = (d_x, d_y, d_z, d_\phi, d_\theta, d_\psi)^T$

Table 3.2: The binding matrices encodes the symmetries of a geometric object. It selects all components of the differential location vector $\mathbf{d} = (d_x, d_y, d_z, d_\phi, d_\theta, d_\psi)^T$ which can actually represent an effective error.

The SPmodel framework provides several operations to transform geometrical objects defined by these so-called locations: the most important are the *composition* $\oplus$ of two locations and the *inversion* $\ominus$ of a location. The composition or compounding operation is equivalent to the head-to-tail relationship described in [Smith et al., 1990]. Refer to the Appendix and the literature for more details on the SPmodel. Within the scope of this work, the SPmodel is used wherever uncertain information appears and has to be transformed. Details are provided at the appropriate location.

3.3.2 Describing a plane within the SPmodel

An infinite plane in Hessian notation is formed by a normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ and a perpendicular distance to the origin d. In this work, it is extended by the center of gravity $\mathbf{o} = (o_x, o_y, o_z)^T$ of the supporting points, which adds location information to the planes and makes visualization clearer as the planes normals don't necessarily intersect with the origin. Section 3.4 presents how these plane model parameters can be calculated from an input point cloud. In order to convert the above notation of a plane into the SPmodel notation, firstly a location vector $\mathbf{x}_{WP} = (x, y, z, \psi, \theta, \psi)^T$ has to be found, which defines the local reference frame of the plane. It is given by

$$\mathbf{x}_{WP} = \begin{bmatrix} o_x \\ o_y \\ o_z \\ \arctan 2(n_y, n_x) \\ \arccos(n_z) \\ 0 \end{bmatrix} \tag{3.12}$$

with plane origin $\mathbf{o} = (o_x, o_y, o_z)^T$. A complete feature is then defined by the quadruple $\mathbf{L}_{WP} = (\mathbf{x}_{WP}, \mathbf{C}_P, \mathbf{p}_P, \mathbf{B}_P)^T$ with perturbation vector $\mathbf{p}_P = (0, 0, 0)^T$, associated covariance matrix $\mathbf{C}_P \in \mathbb{R}^{3\times 3}$, and binding matrix

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

encoding symmetries. The next section shows how the covariance matrix $\mathbf{C}_P$ can be found depending on the uncertainty of the input points.

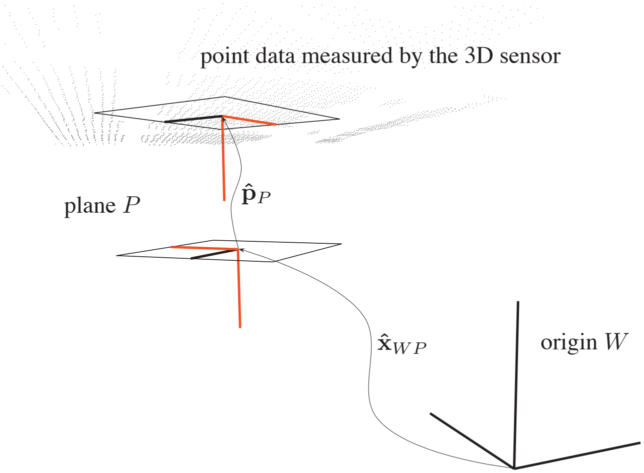


Figure 3.1: Illustration of a plane represented by the SPmodel. Its local reference frame is defined by the estimated location vector $\hat{\mathbf{x}}_{WP} \in \mathbb{R}^{6\times 1}$ with respect to the world reference W. The plane is called centered, if its perturbation vector $\hat{\mathbf{p}}_p$ pointing to its center of gravity equals zero.

3.4 Probabilistic Planar Fitting

This section describes how the plane parameters with associated covariance matrix are extracted from an input point cloud. It is assumed, that the segmentation problem described in the next chapter is solved and the point cloud $P = \{\mathbf{p}_i = (x_i, y_i, z_i)^T | i = 1...N_P\}$ used to find the plane parameters actually represents a planar region without outliers. Furthermore, it is assumed the 3D sensor generating the point cloud is well calibrated, justifying the use of an error model considering random errors only, represented by a normal distribution.

The plane parameters are found in two consecutive steps, the first moments are found by principal component analysis (PCA). Principal component analysis, also called *Karhunen-Loeve* transform is a linear transform that chooses a new coordinate system for a data set such that the greatest variance comes to lie on the first axis, the principal axis, the second greatest on the second and so on. Applied to a set of points representing range measurements of a plane, the first two principal axes lie in the plane and the third axis represents the plane normal, as the variance along the plane normal is the smallest. Starting point is the regression problem

$$R(n_x, n_y, n_z, d) = \sum_{i=0}^{N} w_i (n_x x_i + n_y y_i + n_z z_i - d)^2 \tag{3.14}$$

which has to be minimized. The weighting factor is defined by

$$w_i = \frac{1}{trace(C_i)}, \tag{3.15}$$

where $C_i \in \mathbb{R}^{3\times 3}$ is the covariance matrix of the raw data point i. Deriving (3.14) with respect to d and setting it equal to 0 yields

$$\begin{aligned}
\partial_d R(n_x, n_y, n_z, d) &= 0 \\
\Longleftrightarrow 2\sum_{i=0}^{N} w_i (n_x x_i + n_y y_i + n_z z_i - d)(-1) &= 0 \\
\Longleftrightarrow \sum_{i=0}^{N} w_i (n_x x_i + n_y y_i + n_z z_i) &= \sum_{i=0}^{N} w_i d \\
\Longleftrightarrow \frac{1}{\sum_{i=0}^{N} w_i} \sum_{i=0}^{N} w_i (n_x x_i + n_y y_i + n_z z_i) &= d \\
\Longleftrightarrow \frac{1}{\sum_{i=0}^{N} w_i} \begin{bmatrix} \sum_{i=0}^{N} w_i x_i \\ \sum_{i=0}^{N} w_i y_i \\ \sum_{i=0}^{N} w_i z_i \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} &= d \\
\Longleftrightarrow \mathbf{o} \cdot \mathbf{n} &= d
\end{aligned} \tag{3.16}$$

This means that the best-fitting plane passes through the center of gravity represented by $\mathbf{o} = (o_x, o_y, o_z)^T$. After translating the data points by $-\mathbf{o}$ into the origin, the plane normal $\mathbf{n} = (n_x, n_y, n_z)^T$ is found by calculating the eigenvector corresponding to the smallest eigenvalue of

$$\mathbf{A} = \begin{bmatrix} \sum_{i=0}^{N} w_i x_i^2 & \sum_{i=0}^{N} w_i x_i y_i & \sum_{i=0}^{N} w_i x_i z_i \\ \sum_{i=0}^{N} w_i x_i y_i & \sum_{i=0}^{N} w_i y_i^2 & \sum_{i=0}^{N} w_i y_i z_i \\ \sum_{i=0}^{N} w_i x_i z_i & \sum_{i=0}^{N} w_i y_i z_i & \sum_{i=0}^{N} w_i z_i^2 \end{bmatrix}. \tag{3.17}$$

The second moments are calculated by propagating the error of the raw data into the plane parameters. This can be conveniently carried out using model 4 of table 3.1 on the plane data viewed w.r.t. the local coordinate frame defined by the first moments found above. In this case, the plane parameters are known to be $\mathbf{n} = (0, 0, 1)^T$ and $d = 0$, hence the regression problem related to model 4 minimizes the orthogonal least-square distance.

Let $P^t = \{p_i^t = (x_i^t, y_i^t, z_i^t)^T | i = 1...N\}$ be the point cloud of the plane translated into the world origin and rotated into the global xy-plane using the plane parameters calculated above, i.e. center of gravity $\mathbf{o}$ and plane normal $\mathbf{n}$. The regression problem using model 4 of table 3.1 can be written in matrix terms, yielding:

$$\begin{bmatrix} w_1 z_1^t \\ w_2 z_2^t \\ \vdots \\ w_N z_N^t \end{bmatrix} = \begin{bmatrix} w_1 & w_1 x_1^t & w_1 y_1^t \\ w_2 & w_2 x_2^t & w_2 y_2^t \\ \vdots & \vdots & \vdots \\ w_N & w_N x_N^t & w_N y_N^t \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \tag{3.18}$$

solving for $\beta = (\beta_0, \beta_1, \beta_2)^T$ yields

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T \begin{bmatrix} w_1 z_1^t \\ w_2 z_2^t \\ \vdots \\ w_N z_N^t \end{bmatrix} \tag{3.19}$$

with

$$\mathbf{M} = \begin{bmatrix} w_1 & w_1 x_1^t & w_1 y_1^t \\ w_2 & w_2 x_2^t & w_2 y_2^t \\ \vdots & \vdots & \vdots \\ w_N & w_N x_N^t & w_N y_N^t \end{bmatrix}. \tag{3.20}$$

By calculating the Jacobian $\mathbf{F}$ of β, given by

$$\mathbf{F} = \begin{bmatrix} \frac{\partial\beta_0}{\partial x_1} & \frac{\partial\beta_0}{\partial y_1} & \frac{\partial\beta_0}{\partial z_1} & \frac{\partial\beta_0}{\partial x_2} & \frac{\partial\beta_0}{\partial y_2} & \frac{\partial\beta_0}{\partial z_2} & \cdots & \frac{\partial\beta_0}{\partial x_N} & \frac{\partial\beta_0}{\partial y_N} & \frac{\partial\beta_0}{\partial z_N} \\ \frac{\partial\beta_1}{\partial x_1} & \frac{\partial\beta_1}{\partial y_1} & \frac{\partial\beta_1}{\partial z_1} & \frac{\partial\beta_1}{\partial x_2} & \frac{\partial\beta_1}{\partial y_2} & \frac{\partial\beta_0}{\partial z_2} & \cdots & \frac{\partial\beta_1}{\partial x_N} & \frac{\partial\beta_1}{\partial y_N} & \frac{\partial\beta_1}{\partial z_N} \\ \frac{\partial\beta_2}{\partial x_1} & \frac{\partial\beta_2}{\partial y_1} & \frac{\partial\beta_2}{\partial z_1} & \frac{\partial\beta_2}{\partial x_2} & \frac{\partial\beta_2}{\partial y_2} & \frac{\partial\beta_0}{\partial z_2} & \cdots & \frac{\partial\beta_2}{\partial x_N} & \frac{\partial\beta_2}{\partial y_N} & \frac{\partial\beta_2}{\partial z_N} \end{bmatrix}, \tag{3.21}$$

the covariance matrix $\mathbf{C}$ can be calculated as

$$\mathbf{C} = \mathbf{F} \begin{bmatrix} \mathbf{C}_1^t & 0 & \cdots & 0 \\ 0 & \mathbf{C}_2^t & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{C}_N^t \end{bmatrix} \mathbf{F}^T. \tag{3.22}$$

Note that $\mathbf{C}_i^t \in \mathbb{R}^{3\times3}$ with $i = 1 \ldots N$ are the covariance matrices of the input points w.r.t. the local reference frame of the plane. The output covariance matrix $\mathbf{C}$ is the sought-after local covariance matrix of the SP-location describing a planar segment.

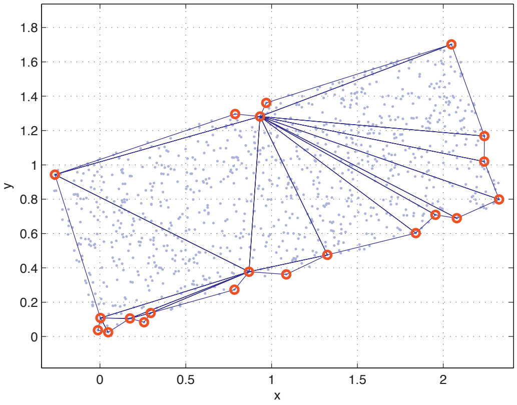


Figure 3.2: The initial planar region composed of a set of points (depicted by gray dots) is transformed into a set of triangles (represented by blue lines) or *alpha*-shape. Note that the input 1000 points are represented by 19 triangles requiring only 21 vertices. The complexity of the initial point cloud was therefore reduced by almost 98% without losing the shape. In a further step, the triangles are converted into polygons (see figure 3.3).

3.5 Planar Segments

For SLAM in simple structured environments, a representation based on infinite planes can be adequate as shown in chapter 5. In most cases however, infinite features are not an ideal choice, firstly as they are not very discriminative and therefore sometimes difficult to compare during data association and secondly as they cannot be rendered in an appealing way. This is the reason why segment information has been added to the infinite plane parameters. As shown below, this leads to highly compressed but still detailed representations.

The segment information of a planar feature essentially is a set of polygons, generated in the following way: At first, a raw 3D scan represented in Cartesian coordinates is segmented as shown below into planar regions defined by a set of supporting points (see figure 3.2). In the next step, each region is approximated by an alpha-shape, a hull also allowing non-convex boundaries defined by a constraint called α. This triangulation is then decimated (see [Schroeder et al., 1992]) and stored as a set of polygons (see figure 3.3). The advantage of such a representation not only lies in the reduced amount of required storage memory, which can easily be fifty times smaller than the raw data point cloud, but also in the topology of the segment which can be grown incrementally in an efficient

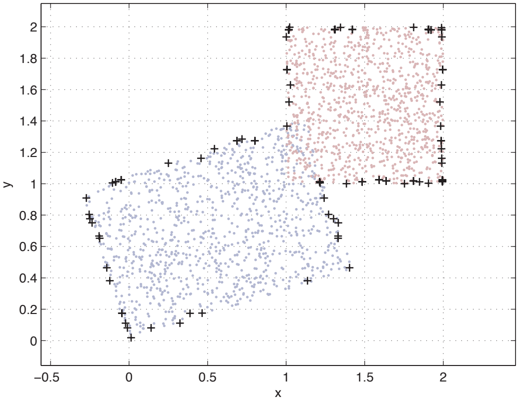


Figure 3.3: Illustration of the merging operation of two planar segments. Each initial point cloud (the two rectangular shapes) is composed of 1000 data points. The resulting contour (black crosses) is composed of 97 vertices or 67 polygons.

way. The latter is important for a SLAM algorithm, as it has to handle monotonically growing data sets. Furthermore, segment information can be exploited positively for data association, as described below.

3.6 Summary

This chapter presented the representation used based on infinite planar features with associated segment information. Plane parameters describing infinite planes are complemented by segment information composed of a set of polygons, allowing to precisely describe the shape of a plane including holes while staying memory efficient at the same time. It also showed how the parameters of the plane are calculated from the input raw data. Two different plane models are used for this purpose, the Hessian Normal Form to obtain the first moments through principal component analysis and the second moments are calculated by error propagation. The resulting fitting method allows to calculate the uncertainty of a plane based on the uncertainty of the raw data, which in turn has been modelled the closest to the reality as possible. It is believed that this carefully developed error model positively affects the performance of the SLAM algorithm.

The next chapter addresses the extraction of the planar features, which uses the fitting methods presented above.

Chapter 4

Feature Extraction

4.1 Introduction

A key issue on the way towards feature-based SLAM is the feature extraction process. In order to be able to build a reliable robot navigation system using a feature-based representation, these features have to be extracted as reliably and precisely as possible. Measurement noise, scene clutter, and dynamic objects make this process difficult.

As mentioned before, in mobile robotics, range finders have become the most popular sensors for navigation and mapping as they directly provide distance measurements at high density and precision. In 3D space, this data can be complex as a single scan can be composed of many ten-thousand data points. Planar feature extraction provides means of reducing this complexity while at the same time keeping the information content.

The crucial part of feature extraction is the segmentation, which is the problem of dividing raw range data into data representing sought-after features and background data. In computer vision and object recognition, this has been an active field of research for decades (see for example [Besl and Jain, 1985], [Fan et al., 1988], [Yokoya and Levine, 1989]). There, typical test images generally consist of known objects composed of planar surfaces scanned under controlled conditions. Some typical example images scanned by a Perceptron laser scanner are depicted in figure 4.1.

In mobile robotics, range data is especially challenging, as it is not known a priori what the scanned scene contains. Therefore, the segmentation process has to be particularly robust with respect to unstructured clutter, dynamic objects and measurement noise. Figure 4.2 shows a typical scene scanned at our lab. It depicts an office corridor with some students sitting at tables, the characteristic ceiling structure of the local buildings is visible as well as some planar structures like the ceiling and the wall on the right. Note that even for the human eye, it is not obvious to find all planar segments.

Besides the segmentation process, another important issue is the fitting process of

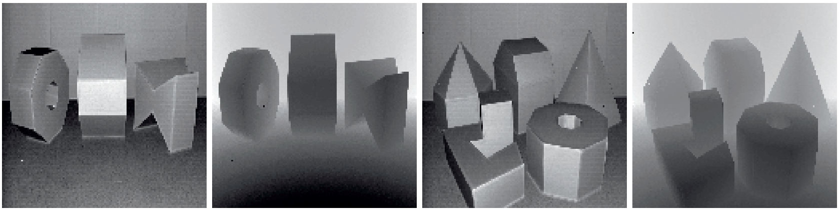


Figure 4.1: Some typical data sets [of Southern Florida] used for planar segmentation in computer vision and object recognition generated by a Perceptron 3D scanner. Note that the first and third image from the left are the corresponding intensity images whereas the other two show depth information.

the (planar) model to the data, which should take into account uncertainties of the raw data and output planar segments with associated uncertainty information. In this way, all information available from the 3D sensor is used and propagated to the model parameters in a consistent way without the need of making additional assumptions.

This chapter is organized as follows. The next section presents the definition of planar segmentation and lists related work. The two subsequent sections present and discuss the two segmentation algorithms developed.

4.2 Planar Segmentation

4.2.1 Definition

Informally, segmenting a range image is the process of labelling pixels so that pixels whose measurements are of the same surface are given the same label [Hoover et al., 1996]. The following formal definition is taken from [Gonzalez and Woods, 1992]:

Let $\mathcal{R}$ represent the entire image region[1] composed of $N_r \times N_c$ range measurements ρ_{rc} with $r \in \{1, 2, \ldots, N_r\}$ and $c \in \{1, 2, \ldots, N_c\}$. The segmentation can be viewed as the process that partitions $\mathcal{R}$ into $N_\mathcal{R}$ subregions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{N_\mathcal{R}}$, such that

(1) $\cup_{i=1}^{N_\mathcal{R}} \mathcal{R}_i = \mathcal{R}$
(2) $\mathcal{R}_i$ is a connected region, $\forall i = 1, 2, \ldots, N_\mathcal{R}$
(3) $\mathcal{R}_i \cap \mathcal{R}_j = \varnothing$ for all i and j, $i \neq j$
(4) $P(\mathcal{R}_i) = TRUE$ for $i = 1, 2, \ldots, N_\mathcal{R}$
(5) $P(\mathcal{R}_i \cup \mathcal{R}_j) = FALSE$ for $i \neq j$

---

[1] here ”image region” is the same as a ”range image” or a ”3d scan”.

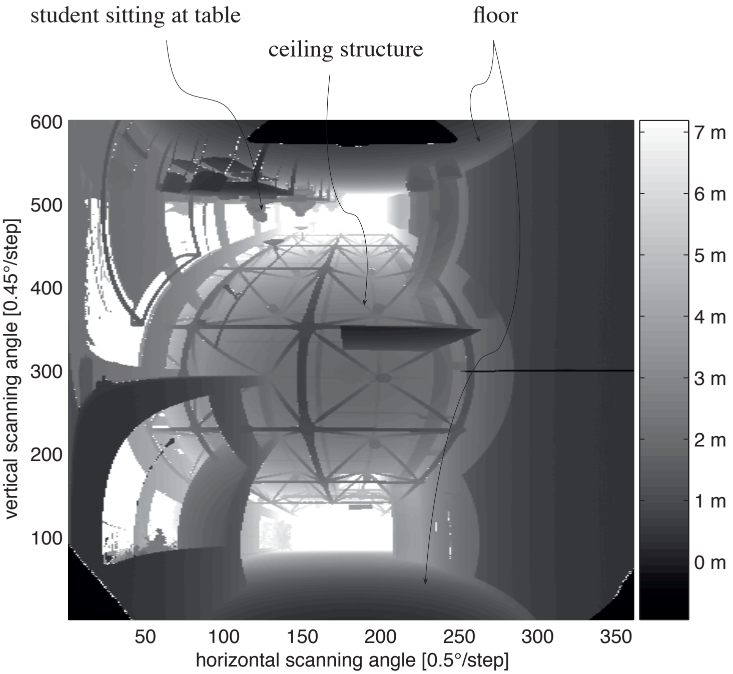


Figure 4.2: A typical 3D scan taken by the rotating 2D laser scanner mounted on the BIBA0 robot visualized in polar coordinates. It consists of 601 2D scans taken at an angular resolution of $0.45°$ per step and 361 data points each, resulting in a total of 216961 data points covering $180°(h) \times 270°(v)$. The scanned scene shows a corridor of the Autonomous Systems Lab, eye-catching features are the students sitting at tables in the top left of the image and the characteristic ceiling structure of the EPFL buildings but also the large planar areas forming the ground and the wall on the right.

where $\varnothing$ is the empty set and $P(\mathcal{R}_i)$ is a predicate over the points in set $\mathcal{R}_i$.

$$P(\mathcal{R}_i) = TRUE \text{ iff } d(\rho_{rc}, p) \leq \tau, \tag{4.1}$$

where d is metric, p a plane model and τ a given constant.

This example of a formal definition for planar range image segmentation shows characteristic weaknesses: Firstly, points representing measurement outliers not necessarily forming a connected region are not considered, as (2) wouldn't be satisfied. Furthermore, (5) doesn't hold in case of two non-bordering regions with the same properties, i.e. if they have the same normal and distance to the origin. In this work, the following less restrictive definition is used:

(1) $(\cup_{i=1}^{N_{\mathcal{R}}} \mathcal{R}_i) \cup \mathcal{S} = \mathcal{R}$, $\mathcal{S}$ are the points not belonging to any region $\mathcal{R}_i$
(2) $\mathcal{R}_i \cap \mathcal{R}_j = \varnothing$ for all i and j, $i \neq j$
(3) $P(\mathcal{R}_i) = TRUE$ for $i = 1, \ldots, N_{\mathcal{R}}$
(4) $P(\mathcal{R}_i \cup \mathcal{R}_j) = FALSE$ for bordering regions $i \neq j$

$\mathcal{S}$ represents points which are not lying on planar structures and therefore not to be included as regions $\mathcal{R}_i$.

4.2.2 Related Work

In computer vision, planar range image segmentation has been an active field of research for the last two decades. Highly detailed geometric models, often represented as complex triangle meshes challenge rendering performance, transmission bandwidth and storage capacities [Hoppe, 1996]. Besides for this so-called *mesh-simplification* [Schroeder et al., 1992], [Hoppe, 1996], recovering of planar segments is used for *reverse-engineering* aiming at reconstructing for example industrial parts as CAD models the most precisely possible [Hoppe et al., 1992], [Curless and Levoy, 1996].

In mobile robotics, planar feature-based representations are used since laser range finders are available on the market. As mentioned in the introduction of this chapter, the data generated by these sensors is very irregular and subject to noise, which justifies the development of planar segmentation algorithms specific to robotic data. [Horn and Schmidt, 1995b] extracted vertical 3D planes for SLAM in 2D space. They extract features using the Hough Transform [Hough, 1959] coupled with an iterative refinement. This work aims at extracting planes at any position and orientation rather than only vertical planes. [Sequeira et al., 1999] extract planes for 3D reconstruction using a hybrid region-based edge-based polynomial surface segmentation method and aligns consecutive scans using an algorithm related to the *iterative closest point algorithm* [Besl and McKay, 1992] referred to as *ICP* in the following. In this work, the Extended Kalman Filter is used to estimate the map and the robot pose which leads to globally consistent maps. [Hähnel et al., 2003] extract planes by using a region-growing algorithm starting at a random point and a plane sweeping method based on normal directions. In this work, this region-growing method is refined by starting the region-growing process at the flattest local area of the scan. Furthermore, inefficient nearest-neighbor search is avoided by exploiting the inherent scan data topology. [Liu et al., 2001] create 3D maps by estimating planar segments using the Expectation Maximization (EM) algorithm. This iterative procedure takes several minutes and is therefore not well-suited for navigation tasks. The method presented in this work is an order of magnitude more efficient. Finally, [Kohlhepp et al., 2004] extract planes in real-time using the scan-line grouping algorithm presented by [Jiang and Bunke, 1994]. This algorithm groups neighboring linear segments approximating single scan lines together in an efficient way. However, it requires that lines of data in the underlying 3D scan can actually be represented by a sequence of linear segments which is not the case for all range sensors (for example the Swiss Ranger).

4.2.3 Taxonomy of Surface Segmentation Algorithms

In order to get a better overview of the numerous existing planar segmentation algorithms, a classification is useful. The following two general classification possibilities were found in the literature.

[Faugeras, 1993] states that there are two ways of segmenting entities in a picture, dominating in the literature. The first is the so-called *split technique*, which starts from the entire object and checks if it is homogeneous according to some criterion. If not, the object is split into a number of subobjects and the process is recursively applied to the subobjects until they become homogeneous or too small. The second method is a merging scheme, also called *region-growing* described below.

[Suk and Bhandarkar, 1992] or [Sequeira et al., 1995] generally classify surface segmentation methods into region-based methods versus edge-based methods. Edge-based methods investigate the characteristics of edges in the image in order to make inferences about the regions they enclose.

Both segmentation methods presented in this work fall into the region-based category. The first one (*Grid-based segmentation*) is optimized for speed of execution aiming at extracting the most important planar areas of a scan rapidly, the second one (*Region-based segmentation*) is slower but more precise and also suited to extract smaller planes. Both methods rely on some well-known concepts which are briefly explained in the following.

Ransac

The Ransac (Random Sample Consensus) algorithm [Fischler and Bolles, 1981] is a popular choice for fitting models to data due to its simplicity and robustness with respect to outliers. Algorithm 1 is a pseudo-code description of the Ransac algorithm for segmenting a single plane from a point cloud. It is mentioned explicitly as it forms a part of the GBS algorithm presented below. For a predefined number of iterations N_I, the segmen-

Algorithm 1 $\hat{\mathbf{p}} = planarSegRansac(V)$

$V = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_V})$	*input* point cloud composed of N_V 3D points
N_C	number of points within the environment of the currently defined plane
N_M	found maximum number points in the defined environment of the plane
N_I	predefined number of Ransac iterations
d	orthogonal distance to plane
τ_r	maximum allowed distance for supporting points
$\mathbf{n}_p$	normal of plane $\mathbf{p}$
$\mathbf{o}_p$	origin of plane $\mathbf{p}$
$\hat{\mathbf{p}}$	*output* best found plane

```
$N_M \leftarrow 0$
for $i = 1$ to $N_I$ do
    randomly select 3 different points $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ of the input point cloud $V$
    $(\mathbf{n}_p, \mathbf{o}_p) \leftarrow createPlane(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$        ▷ create plane $\mathbf{p} = (\mathbf{n}_p, \mathbf{o}_p)$
    for $j = 1$ to $N_V$ do        ▷ count points close to plane
        $d \leftarrow distanceToPlane(\mathbf{v}_j, \mathbf{n}_p, \mathbf{o}_p)$
        if $d < \tau_r$ then
            $N_C \leftarrow N_C + 1$
        end if
    end for
    if $N_C > N_M$ then        ▷ get plane with max. number of supporting points
        $N_M = N_C$
        $\hat{\mathbf{p}} \leftarrow \mathbf{p}$        ▷ write to best plane $\hat{\mathbf{p}} = (\mathbf{n}_{\hat{p}}, \mathbf{o}_{\hat{p}})$
    end if
end for
```

tation performance of a plane defined by three randomly chosen vertices $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ is evaluated by counting the number of points N_C lying within a predefined orthogonal distance τ_r. The plane with the highest number of supporting points N_M is output as the best planar segment $\hat{\mathbf{p}}$ found.

The quality of the resulting segmentation directly depends on the predefined distance threshold τ_r and the chosen number of iterations N_I. The chance to find a correct segmentation increases by augmenting the number of iterations N_I. However, the higher N_I, the slower the algorithm. Hence, a trade-off in speed has to be taken into account to realize good segmentation results. The complexity of the Ransac algorithm can be expressed as $O(N_I \cdot N_V)$.

Let $p_g \in [0, 1]$ be the probability that a randomly chosen data item is part of a good model and $p_f \in [0, 1]$ be the probability that the algorithm exits without finding a good segmentation. p_g and p_f are related by $p_f = (1 - p_g^{N_M})^{N_I}$. Here, $N_M = 3$ as three data items are necessary in order to describe a plane. Hence,

$$N_I = \frac{log(p_f)}{log(1 - p_g^3)} \tag{4.2}$$

Unfortunately, p_f and p_g are generally not known a priori and change from scene to scene. Therefore an empirical analysis is indispensable.

Note that the plane found by this algorithm is not necessarily a connected region as required by (2) of the above-mentioned segmentation definition taken from [Gonzalez and Woods, 1992] which examplifies the short-comings of this definition.

Region-Growing

A region-growing algorithm starts from single entities of an input range image like points or planar patches and grows these into larger regions by merging them with matching neighbors. This process ends, when a certain stopping criteria is reached, e.g. if the approximation error of a planar region exceeds a tolerance threshold. An example of a region-growing algorithm for planar segmentation can be found in [Faugeras, 1993] or [Hähnel et al., 2003]. The pseudo-code of the latter is illustrated by algorithm 2, as it forms the basis of the RGS algorithm described below. The algorithm starts by randomly selecting a point $\mathbf{v}_1$ of the input point cloud V and its closest neighbor $\mathbf{v}_2$. A candidate point $\mathbf{v}'$ is added to the set of planar points π if the minimal distance from π to $\mathbf{v}'$ is less than a threshold δ. The point $\mathbf{v}'$ is accepted if, when added to π, the average residual is less than a threshold ϵ and the distance between the optimal plane and $\mathbf{v}'$ is less than a threshold γ.

Other Methods

Other plane segmentation methods not further considered here are Agglomerative Hierarchical Clustering [Faugeras, 1993], which is not very robust with respect to outliers,

Algorithm 2 growRegion($\mathbf{v}_1, V$)

```
π ← {v1, v2}
for all v' ∈ V do
    if dist(π, v') < δ then
        π' ← merge(π, v')
        if averageError(π') < ε and error(π', v') < γ then
            π ← π'
        end if
    end if
end for
```

the 3D Hough Transform [Hough, 1959], which is not very accurate and slow, scan-line grouping [Jiang and Bunke, 1994], which is efficient but requires that single scan-lines can be approximated by a set of lines.

4.3 The Grid-based Segmentation (GBS) Algorithm

4.3.1 Algorithm Description

Using the Ransac algorithm (see Algorithm 1) for planar segmentation directly on the raw data doesn't lead to satisfactory results especially in terms of efficiency. By using equation (4.2), the number of required Ransac iterations N_I to detect a plane in a reliable[2] way can exemplary be evaluated. For example, to detect a plane composed of 4000 data points in a point cloud of 100000 data points, over 45000 Ransac iterations are needed.

Decomposing the space into grid cells

Therefore, instead of using the Ransac algorithm directly on the input 3D data, a divide-and-conquer mechanism was used leading to a more efficient algorithm (see Algorithm 3 for pseudo-code) in a similar way as described in [Weingarten et al., 2003]. The space occupied by the input point cloud V is decomposed into a set of $N_{\mathcal{C}}$ regular cubes from now on called grid cells $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{N_{\mathcal{C}}}\}$ with a predefined sidelength $s_{\mathcal{C}}$. All N_V points are subsequently associated to their corresponding grid cell $\mathcal{C}_i$. In this way, the rather complex point cloud composed of many ten-thousand data points (see figure 4.5 (b) for an example) is divided into smaller chunks of data of around a thousand data points per cell. The spatial topology defined by the grid cells can be regarded as a region-adjacency graph with the cells as nodes and the neighborhood relations as edges, which will be exploited below to perform a region-growing operation. Before that, a single plane

---

[2] with a probability of 95%

Algorithm 3 $GBS(V)$ - the Grid-Based Segmentation Algorithm

$\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{N_\mathcal{C}}\}$	the grid structure composed of $N_\mathcal{C}$ grid cells
$\mathcal{V}_i.points$	the points contained in grid cell $\mathcal{C}_i$
$\mathcal{C}_i.plane$	the plane approximating the points of grid cell $\mathcal{C}_i$
$s_\mathcal{C}$	the side length of the grid cells
N_I	number of Ransac iterations used (default is 100)
τ_r	Mahalanobis distance used by the Ransac algorithm
τ_m	χ^2-threshold for plane comparison

```
$\mathcal{C} \leftarrow createGridCellStructure(s_\mathcal{C})$    ▷ create the grid
for all $\mathbf{v}_i \in V$ do
    $writeToCorrespondingGridCell(\mathbf{v}_i, \mathcal{C})$ ▷ add $\mathbf{v}_i$ to corresponding grid cell $\mathcal{C}_i$
end for
for all $\mathcal{C}_i \in \mathcal{C}$ do
    $\mathcal{C}_i.plane \leftarrow planarSegRansac(\mathcal{C}_i.points)$    ▷ find plane for data contained in
    every grid cell using the probabilistic Ransac algorithm with parameters $N_I, \tau_r, \tau_m$
end for
while 1 do
    $\mathcal{C}_s \leftarrow getMinError(\mathcal{C})$    ▷ find unprocessed cell with minimum fitting error
    $q \leftarrow merge(q, \mathcal{C}_s)$
    $q \leftarrow growRegionGBS(q, \mathcal{C})$
    while $q \neq \varnothing$ do
        $q \leftarrow growRegionGBS(q, \mathcal{C})$
    end while
end while
```

is found for every cell segmenting the points of the cell into points belonging to the grid cell's plane and points that do not (see figure 4.5 (c)). The segmentation procedure used is the above-mentioned Ransac (see algorithm 1), which in this case operates on the limited amount of data contained in single cells. Using again equation (4.2), the default number of $N_I = 100$ leads to a probability of finding the correct plane $p_f = 0.9987$ assuming that $p_g = 0.4$, which is reasonable for environments composed of many planar walls like our office environment. Note that this number N_I represents the maximum number of Ransac iterations, the actual performed number of iterations can fall considerably below, because it doesn't make sense to continue searching for a better segmentation if all points of the cell are already included in the current model. This heuristic makes the algorithm substantially faster in environments composed of many planar surfaces.

Ransac considering uncertainties

As uncertainty information is available in the raw data, the metric used in the Ransac algorithm is the Mahalanobis distance instead of the Euclidean distance. As mentioned in algorithm 1, line 7, a threshold τ_r decides whether a point $\mathbf{v}_j$ belongs to the planar region or not. There, the used metric is the orthogonal distance

$$d = |\mathbf{n}_p \mathbf{v}_j - \mathbf{n}_p \mathbf{o}_p| \tag{4.3}$$

to the current plane $\mathbf{p}$. By considering the uncertainties of the raw data points, this Euclidean distance can be replaced by the squared Mahalanobis distance

$$D^2 = (\mathbf{v}_{j_p} - \mathbf{v}_j)^T \mathbf{P}_j^{-1} (\mathbf{v}_{j_p} - \mathbf{v}_j) \tag{4.4}$$

where $\mathbf{v}_{j_p}$ is the projection of $\mathbf{v}_j$ onto plane $\mathbf{p}$ and $\mathbf{P}_j$ is the covariance matrix associated to $\mathbf{v}_{j_p}$.

Region-Growing

Finally, a region-growing operation leads to the final segmentation (see Figure 4.5 (d)). It starts at the cell $\mathcal{C}_s$ containing the plane with the smallest fitting error, representing the flattest surface. The above mentioned region-adjacency graph provides topological information and is searched using a breadth-first-search strategy (see algorithm 4) which in comparison to depth-first-search strategies leads to smoother region-growing, as the region is grown in every direction simultaneously. A neighboring plane candidate P_n is compared to the current processed region P_r by means of a hypothesis test based on the Mahalanobis distance, described in the following.

Comparing two planes in a probabilistic way

According to the SPmodel, a plane is represented by a quadruple

$$\mathbf{L}_{WE} = (\hat{\mathbf{x}}_{WE}, \hat{\mathbf{p}}_E, \mathbf{C}_E, \mathbf{B}_E) \tag{4.5}$$

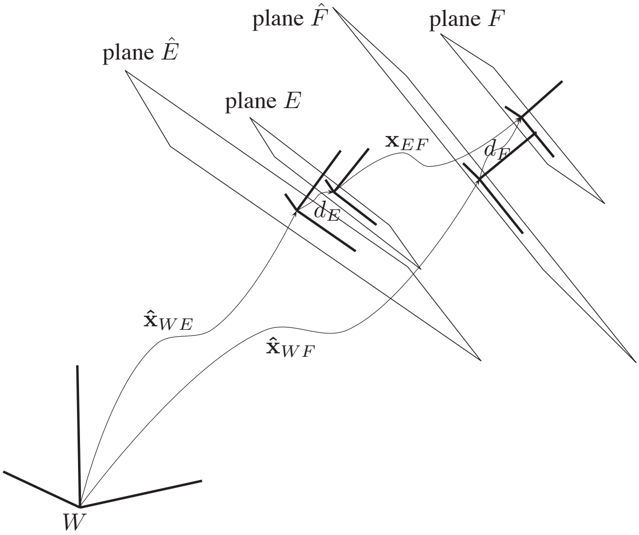


Figure 4.3: $\mathbf{x}_{WE}$ and $\mathbf{x}_{WF}$ are location vectors from the base reference W into the local reference frames of plane E and F. Planes E and F are compared probabilistically by means of a χ^2-hypothesis test based on the squared Mahalanobis distance.

with the location vector $\hat{\mathbf{x}}_{WE}$ defining the transform from the world coordinate system W into the local reference system E, the binding matrix

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

accounting for symmetries and the perturbation vector $\hat{\mathbf{p}}_E$ with associated covariance matrix $\mathbf{C}_E$ accounting for errors. Let a second plane be described by

$$\mathbf{L}_{WF} = (\hat{\mathbf{x}}_{WF}, \hat{\mathbf{p}}_F, \mathbf{C}_F, \mathbf{B}_F), \tag{4.7}$$

see figure 4.3 for an illustration. Comparing these two planes can be carried out in a probabilistic way by means of a χ^2-hypothesis test based on the squared Mahalanobis distance as follows.

The two planes coincide if their relative location vector

$$\mathbf{B}_{EF}\mathbf{x}_{EF} = \mathbf{0}. \tag{4.8}$$

In this case, the binding matrix $\mathbf{B}_{EF}$ of the pairing corresponds to $\mathbf{B}_E$ or $\mathbf{B}_F$ as the feature types correspond. $\mathbf{x}_{EF} = (x, y, z, \phi, \theta, \psi)^T$, hence $\mathbf{B}_{EF}\mathbf{x}_{EF} = (z, \theta, \psi)^T = \mathbf{0}$, meaning that the two planes coincide if their three relative parameters z, θ and ψ equal zero.

This can be represented by the implicit non-linear measurement equation

$$\begin{aligned}\mathbf{f}_m(\mathbf{p}_E, \mathbf{p}_F) &= \mathbf{B}_{EF}\mathbf{x}_{EF} \\ &= \mathbf{B}_{EF}(\ominus\mathbf{x}_{WE} \oplus \mathbf{x}_{WF}) \\ &= \mathbf{B}_{EF}(\ominus(\hat{\mathbf{x}}_{WE} \oplus \mathbf{B}_E^T\mathbf{p}_E) \oplus (\hat{\mathbf{x}}_{WF} \oplus \mathbf{B}_F^T\mathbf{p}_F)) \\ &= \mathbf{B}_{EF}(\ominus\mathbf{B}_E^T\mathbf{p}_E \ominus \hat{\mathbf{x}}_{WE} \oplus \hat{\mathbf{x}}_{WF} \oplus \mathbf{B}_F^T\mathbf{p}_F) \\ &= \mathbf{B}_{EF}(\ominus\mathbf{B}_E^T\mathbf{p}_E \ominus \hat{\mathbf{x}}_{EF} \oplus \mathbf{B}_F^T\mathbf{p}_F) \\ &= \mathbf{0}\end{aligned} \tag{4.9}$$

relating the two planes. After linearization and assuming the estimations are centered (see chapter 5 for details), these expressions reduce to

$$\begin{aligned}\mathbf{h}_m &= \mathbf{f}_m(\hat{\mathbf{p}}_E, \hat{\mathbf{p}}_F) = \mathbf{B}_{EF}\hat{\mathbf{x}}_{EF} \\ \mathbf{H}_m &= \left.\frac{\partial \mathbf{f}_m}{\partial \mathbf{p}_E}\right|_{(\hat{\mathbf{p}}_E, \hat{\mathbf{p}}_F)} = -\mathbf{B}_{EF}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{EF}\}\mathbf{B}_F^T \\ \mathbf{G}_m &= \left.\frac{\partial \mathbf{f}_m}{\partial \mathbf{p}_F}\right|_{(\hat{\mathbf{p}}_E, \hat{\mathbf{p}}_F)} = \mathbf{B}_{EF}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{EF}, \mathbf{0}\}\mathbf{B}_E^T\end{aligned} \tag{4.10}$$

The squared Mahalanobis distance is then expressed by:

$$D^2 = \mathbf{h}_m^T \left(\mathbf{H}_m\mathbf{C}_E\mathbf{H}_m^T + \mathbf{G}_m\mathbf{C}_F\mathbf{G}_m^T\right)^{-1} \mathbf{h}_m \tag{4.11}$$

The hypothesis based on the square Mahalanobis distance D^2 is considered true with a significance level α, if

$$D^2 \leq \chi^2_{r,\alpha}. \tag{4.12}$$

$\chi^2_{r,\alpha}$ is a threshold value that can be obtained from the χ^2-distribution with rank 3, α is the probability of rejecting a good match. If the candidate pairing is accepted, the two planes E and F are fused, which is realized by merging supporting point clouds and calculating a new probabilistic fit as described in the previous chapter.

The time complexity of the algorithm is $O(N_V(N_\mathcal{C} + N_E))$ where $N_\mathcal{C}$ is the number of grid cells, N_V is the number of input points and N_E is the number of edges in the region-adjacency graph. Analogously, the space complexity is $O(N_V(N_\mathcal{C} + N_E))$.

Algorithm 4 $growRegionGBS(q,\mathcal{C})$

```
$\mathcal{C} = \{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_{N_\mathcal{C}}\}$ the grid structure $\mathcal{C}$ composed of $N_\mathcal{C}$ grid cells
$\mathcal{C}_i.points$ the points contained in grid cell $\mathcal{C}_i$

$\mathcal{C}_f \leftarrow getFirstCellFromQueue()$    ▷ get the first cell from the queue
$\mathcal{N} = (\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_N) \leftarrow getValidNeighborsOfCell(\mathcal{C}_f)$
for all $\mathcal{C}_i \in \mathcal{N}$ do
    if $planesFit(\mathcal{C}_f,\mathcal{C}_i)$ then    ▷ comparison of the planes of grid cell $\mathcal{C}_f$ and $\mathcal{C}_i$
        $q \leftarrow addToEnd(q,\mathcal{C}_i)$
    end if
end for
```

4.3.2 Results

Figure 4.4 shows the result of applying the GBS algorithm to a simulated test scene containing 10 planes of a size of around 10×6 meters. The chosen grid cell size is $s_\mathcal{C} = 0.25$m, the Ransac distance $\tau_r = 0.01$m, the (maximum) number of Ransac iterations $N_I = 100$.

Another result is shown in figure 4.5. The top left image (a) shows a photograph of the scene which is scanned by the 3D scanner resulting in the 3D scan shown in the top right image (b). Decomposing the space into cubic cells and approximating the data of every cube by a plane leads to (c). After fusing similar neighboring regions together and filtering too small regions, the final image (d) is obtained. Figure 4.6 shows how the segmentation times scale with increasing number of input points. As expected, a clear linear dependency can be observed between the number of input points and the computation time required for segmentation.

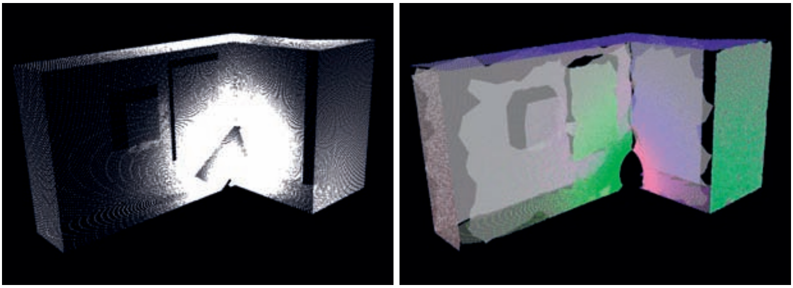


Figure 4.4: The segmentation result of an artificial 3D scan using the grid-based algorithm (GBS). All 10 planar segments have been found successfully. As every grid cell is allowed to contain only a single plane, some contour information is lost leading to artefacts.

4.3.3 Discussion

The GBS algorithm decomposes the space into a regular grid and extracts a plane for every grid cell. In a second step, neighboring planes are compared using a probabilistic measure and fused if found to be corresponding. The main advantage of this algorithms are its robustness w.r.t. outliers and its flexibility, as it doesn't require any structure in the raw data. This means that it is applicable to any type of range data, coming from all kinds of sensors. Stereo vision data for example tends to be irregular, as distance information is available only at corresponding pixel pairs which are heterogeneously distributed over the image.

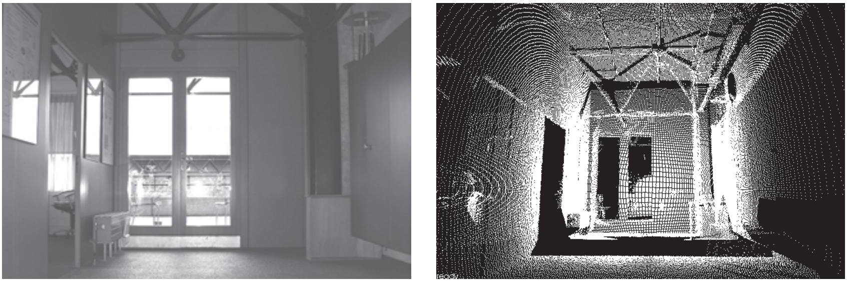
Furthermore, the algorithm can process uncertainty information which is believed to improve the segmentation performance depending on the quality of the input raw data and the noise model. In this case, the input raw data is already of high quality, and hence the improvement gained by using uncertainty information is practically negligible. The region-growing algorithm presented subsequently is therefore non-probabilistic, as it is believed that the gain in performance achieved when leaving out uncertainties matters more that the marginal segmentation quality improvement.

The main drawback of this algorithm is that only a single plane is extracted for each cell, other potential planar segments within the cell are lost. Depending on the application, this is not necessarily unfavorable. In this work, it is aimed at extracting the most important planes of a scene only, hence it is believed that this algorithm is appropriate. On the other hand, it could be imagined to allow several planes per grid cell. However, this idea is not further followed as the algorithm thereby loses its simple structure and efficient behavior.

4.4 A Region-Growing Segmentation (RGS) Algorithm

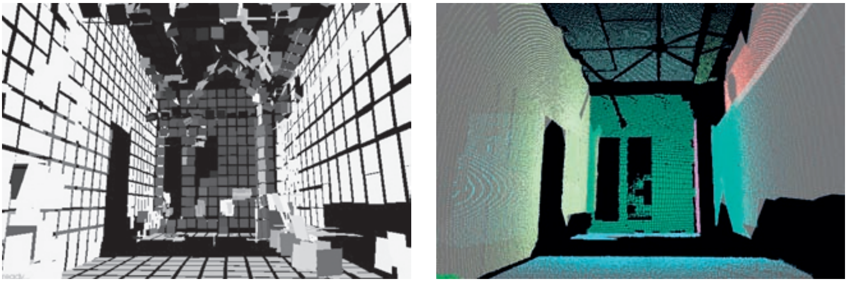
4.4.1 Algorithm Description

The second algorithm developed is based on the region-growing paradigm. It starts by calculating the normals $N = \{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_{N_V}\}$ and the mean-square orthogonal fitting errors $E = \{e_1, e_2, \dots, e_{N_E}\}$ for every input data point $V = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{N_V}\}$. This is done by taking points into consideration lying in a square-shaped odd number-sized (see w_s below) window around $\mathbf{v}_i$. In the main loop (see Algorithm 5 line 6), another breadth-first search strategy is used to grow regions starting at points $\mathbf{v}_{min}$ with best local planar fits. This ensures that region-growing actually starts in a planar area, which is not guaranteed by using a random approach like in [Hähnel et al., 2003]. The breadth-first search strategy is implemented by using a FIFO (”first in first out”) queue q which is initialized by the starting point $\mathbf{v}_{min}$. After initialization, the local environment of $\mathbf{v}_{min}$ is analyzed (see Algorithm 6) and if a neighboring point $\mathbf{v}_i$ satisfies the following 3 constraints, it is added to the queue q.



(a) The example scene shows a corridor of our lab with an open door on the left and the glass door in front leading to the roof terrace.

(b) The same scene visualized as a point cloud measured by the rotating laser scanner.



(c) In every cube of a side length of 0.25m the point data is approximated by a planar patch.

(d) Similar planar patches are fused by region-growing resulting in the final segmentation.

Figure 4.5: This figure shows how the grid-based algorithm (GBS) extracts planes from a point cloud. The real scene (a) is scanned by the rotating laser scanner generating point cloud (b), composed of 216961 points. After associating the points to their corresponding cells, a plane is found for every cell using the Ransac algorithm (c). The final segmentation (d) results from subsequent region-growing and is composed of 12 planar segments.

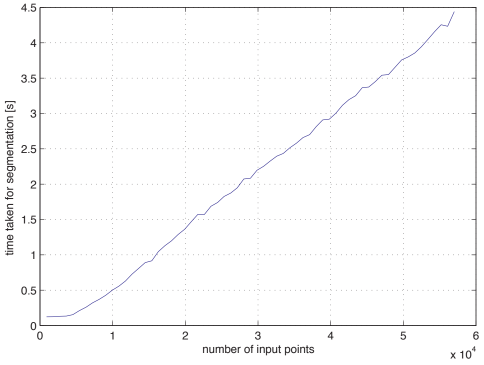


Figure 4.6: The evolution of segmentation times on a Pentium M (1.4 GHz) using the GBS algorithm with increasing number of input points. The x-axis represents the number of input points and ranges from 1000 to 57920. The curve shows that the segmentation times are proportional to the number of input points.

- constraint Γ_1 defines the maximum allowed (Euclidean) distance τ_d between $\mathbf{v}_f$ and $\mathbf{v}_i$.
- constraint Γ_2 defines the maximum allowed distance between $\mathbf{v}_i$ and the current approximative plane defined by normal $\mathbf{n}$ and orthogonal distance d to the origin.
- constraint Γ_3 defines the maximum allowed angle between the normal $\mathbf{n}$ of the current region and local normal $\mathbf{n}_i$ of point $\mathbf{v}_i$.

The breadth-first strategy consists of successively processing the queue elements until it is empty. To find several regions, these steps are repeated for the remaining points.

Note that the neighborhood relations defined by the inherent topology of the 3D scan are used for local neighbor search. Thus, no time-consuming nearest-neighbor search has to be carried out. The complexity of the algorithm is proportional to $O(N_V + N_E)$, where N_V is the number of input points and N_E the number of edges defined by the regular topology of the scan.

4.4.2 Results

Figure 4.8 shows the same artifical scan as in figure 4.4, this time segmented using the RGS algorithm. It can be observed, that the quality of the RGS segmentation is superior to the GBS segmentation as the contours are sharper and less artefacts are visible. However, it also shows that data points lying close to an edge remain unsegmented. This is caused by constraint Γ_3 which ensures, that the local normal of the current candidate point $\mathbf{v}_i$ is similar to the current region's normal. This similarity decreases when approaching an edge.

Figure 4.7 shows how the algorithm scales with increasing number of input points. After some non-linear growth at the beginning which is believed to be related to memory

Algorithm 5 Planar Segmentation by Region-Growing

Parameters:

$V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_V}\}$	all input N_V data points
$N = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_{N_V}\}$	the normals of the local planes
$E = \{e_1, e_2, \ldots, e_{N_E}\}$	the plane fitting orthogonal mean square errors at vertices $\mathbf{v}_i$
$S = \{s_1, s_2, \ldots, S_{N_S}\}$	set of output planar segments
w_s	the size of the sliding window used for local plane estimation

```
$N = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_{N_V}\} \leftarrow calculateNormalsAtVertices(V)$
$E = \{e_1, e_2, \ldots, e_{N_E}\} \leftarrow calculateLocalFitAtVertices(V, s_w)$
$S \leftarrow \varnothing$
$q \leftarrow \varnothing$
$N_{segmented} \leftarrow 0$
while $N_{segmented} < N_V$ do
    $\mathbf{v}_{min} \leftarrow getPointWithMinimumError(V, E)$
    $q \leftarrow merge(q, \mathbf{v}_{min})$
    $q \leftarrow growRegionRGS(q, V, N)$
    while $q \neq \emptyset$ do
        $q \leftarrow growRegionRGS(q, V, N)$
    end while
    $S \leftarrow addToPlanarSegments(S, q)$
end while
```

Algorithm 6 $growRegionRGS(q, V, N, \mathbf{n}, d)$

N_E size of neighborhood (4 or 8-connectivity)
$r_{\mathbf{v}_f}$ row index of first vertex in q
$c_{\mathbf{v}_f}$ columns index of first vertex in q
τ_d predefined threshold defining translational constraint Γ_1
τ_m predefined threshold defining maximum translational constraint Γ_2
τ_α predefined threshold defining rotational constraint Γ_3

```
$\mathbf{v}_f \leftarrow getFirst(q)$
$(r_{\mathbf{v}_f}, c_{\mathbf{v}_f}) \leftarrow getIndicesRI(\mathbf{v}_f)$
$V_n = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_E}\} \leftarrow getValidNeighbors(V, r_{\mathbf{v}_f}, c_{\mathbf{v}_f}, N_E)$
for all $\mathbf{v}_i \in V_n$ do
    $\delta_a \leftarrow calcPointDistance(\mathbf{v}_f, \mathbf{v}_i)$
    if $\delta_a > \tau_d$ then                                   ▷ constraint $\Gamma_1$
        $continue$
    end if
    $\delta_p \leftarrow distanceToPlane(\mathbf{v}_i, \mathbf{n}, d)$
    if $\delta_p > \tau_m$ then                                   ▷ constraint $\Gamma_2$
        $continue$
    end if
    $\mathbf{n}_i \leftarrow getCorrespondingNormal(N, \mathbf{v}_i)$
    $\delta_\alpha \leftarrow calcAngleBetweenNormals(\mathbf{n}, \mathbf{n}_i)$
    if $\delta_\alpha > \tau_\alpha$ then                         ▷ constraint $\Gamma_3$
        $continue$
    end if
    $q \leftarrow addAtEnd(q, \mathbf{v}_i)$
end for
if $length(q) > 2$ then
    $(\mathbf{n}, d) \leftarrow recalcPlaneParameters(q)$
end if
$q \leftarrow removeFirst(q)$
```

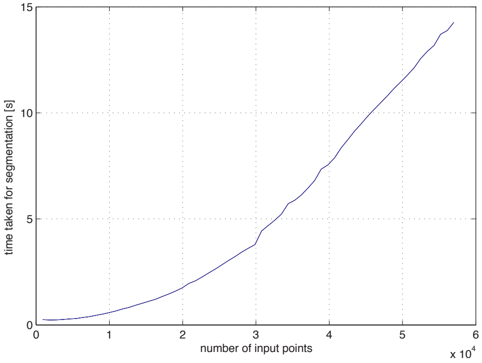


Figure 4.7: The evolution of segmentation times on a Pentium M (1.4 GHz) using the RGS algorithm with increasing number of input points. The x-axis represents the number of input points and ranges from 1000 to 57920. Note that above 30000 points, the curve shows its expected linear shape.

allocation times of involved data structures the curve features a linear shape above 30000 input data points.

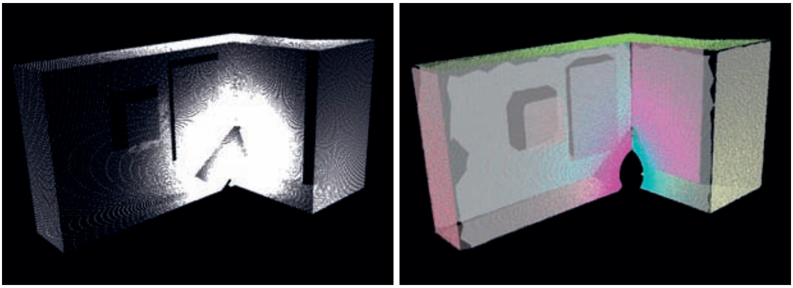


Figure 4.8: An example result showing the planar segmentation of a simulated data set composed of 10 planes. Note that the RGS algorithm was able to extract all planes successfully. The process took several seconds.

4.4.3 Discussion

The region-growing algorithm (RGS) exploits the underlying regular structure of the 3D scan by assuming that points lying next to each other in the range image actually represent neighbors in reality. This holds for the majority of cases and leads to a drastically faster algorithm in comparison to an algorithm that has to carry out the nearest-neighbor search additionally. The local evaluation of plane normals allows to initialize the algorithm properly, asserting that the region-growing operation doesn't start at an edge which could lead to unexpected results. In the current version of the algorithm, uncertainty data is not processed explicitly, as the input data is highly precise. But it should be mentioned that it can easily be extended to processing probabilistic data in a similar way as the GBS algorithm, where in loose words, the Euclidean distance is replaced by the Mahalanobis distance and wherever comparisons are carried out, statistical tests are used. The results of the region-growing algorithm are satisfying in terms of quality, but the number of segmented planes is generally too high, which is addressed in the next section.

4.5 Post-Processing

The two planar segmentation algorithms presented generally lead to a so-called over-segmentation, which means that the number of output planar segments is larger than the number of segments existing in the physical reality. This mainly happens due to non-planar structures breaking up a single planar segment into multiple subsegments (see Figure 4.9 for an example). In terms of the formal definition of planar segmentation, this can be expressed by

$$\begin{aligned}\mathcal{P}(\mathcal{R}_i \cup \mathcal{R}_j) = TRUE \text{ for non-bordering regions } i \neq j \\ \text{belonging to the same physical feature.}\end{aligned} \tag{4.13}$$

This means that non-adjacent planar segments can very well belong to the same physical entity. This over-segmentation can be addressed at different levels. It can make sense to introduce multi-segment features (see [Arras, 2003] for an example in 2D space) that allow to group several subfeatures together forming a single multi-segment feature. A long corridor wall interrupted by doorways or other corridor openings could for example be represented by a single multi-segment feature consisting of several planar (sub)segments. In this work, the over-segmentation is not explicitly addressed on a global level but on a local level of a single 3D scan.

As the number of extracted planar segments per 3D scan stays bounded (see figure 4.10), a basic method comparing all planar segments among themselves has been implemented, merging coinciding segments together. An example result of this post-processing operation is illustrated by figure 4.9. The left image shows the raw scan of a part of the jagged ceiling at EPFL. The middle image shows the raw segmentation result using the region-growing algorithm. Different planar segments have different (random) shades. After fusing similar regions together, the segmentation shown in the right image is obtained where all the subelements of the middle image representing the actual ceiling have been fused together.

Figure 4.11 shows further results of the two algorithms applied to a number of example scans. The left column contains the raw data visualized as point cloud, the middle column is the result obtained using the GBS algorithm and the column on the right holds the results of the RGS algorithm. Note that the arbitrary colors should help to differentiate different planar segments.

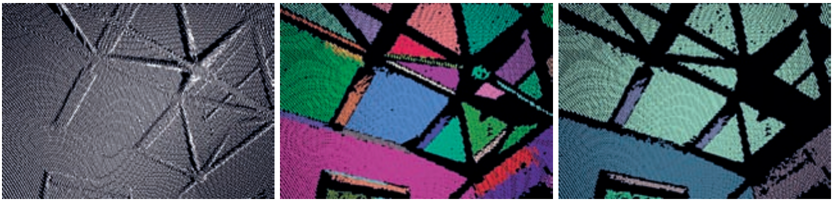


Figure 4.9: A part of a scanned scene showing the ceiling structure at our lab. Note that the cross beams shown in the raw scan on the left break up the single planar segment describing the ceiling into several subsegments depicted in different colors in the middle image. The right image shows the result of the post-processing step, where the several segments of the ceiling (see middle image) have been fused to a single region.

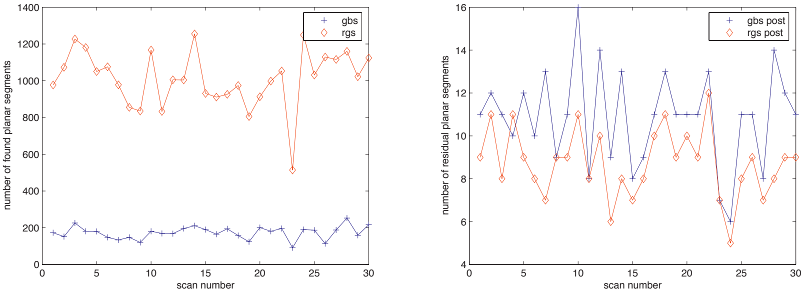


Figure 4.10: The evolution of the number of extracted features in a sequence of scans taken in a typical office environment. Note that the region-based segmentation (RGS) algorithm extracts around 5 times as many planes as the grid-cell based (GBS) algorithm (left), as it is also suited to find small planes. The graph on the right shows the residual number of planar segments after post-processing and filtering segments containing less than 2000 data points. Note that the number of residual features based on the region-growing algorithm (RGS) is often below but close to the number of features based on the grid-based segmentation (GBS) algorithm.

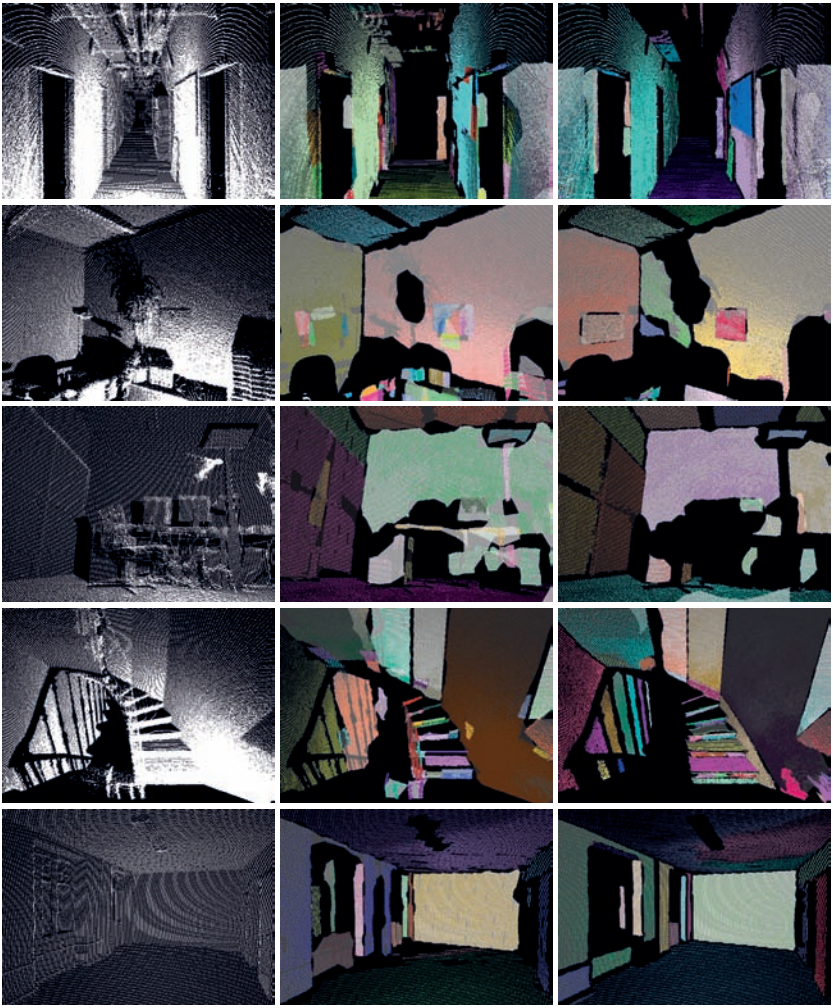


Figure 4.11: Comparison of the final segmentation quality of the two presented algorithms. The left column shows raw data scans, the column in the center the results using the GBS algorithm, the right column the results using the region-growing algorithm (RGS). It can be observed that the quality of the latter is superior in all test scenes. The number of input scan data points is 216961 for each scan and the resulting extracted feature number stays below 30 at all times for both algorithms.

4.6 Summary

This chapter presented two algorithms for planar segmentation of range images, both capable of processing probabilistic data. The grid-based segmentation (GBS) algorithm aims at efficient processing and doesn't require a regular structure in the raw data points, whereas the region-based segmentation (RGS) algorithm benefits of the inherent data topology to become more efficient. The former algorithm is well-suited to extract the most important planes from a 3D scan, but it is inferior to the latter algorithm in terms of segmentation quality when it comes to smaller planes. As both algorithms lead to an over-segmentation caused by structures which break up regions into smaller subregions, a post-processing step has been developed which fuses corresponding subregions back together. After filtering out the smallest regions, this leads to considerably smaller feature numbers of the order of magnitude of around 10 per 3D scan, which is believed to be reasonably low for efficient localization and mapping, which is addressed in the next chapter.

Note that due to the high quality of the input raw data and the related small uncertainty in every data point, it is not of significant importance to use the uncertainty information for the segmentation step. In other cases however, where the uncertainty of the raw data is greater, it should lead to significantly better segmentation results, justifying the associated computational overhead.

Chapter 5

SLAM

5.1 Introduction

SLAM (Simultaneous Localization And Mapping) or CML (Concurrent Mapping and Localization) is the process of incrementally building a map of the robot's environment while tracking its pose at the same time. As the above terms imply, it is composed of two subproblems, localization and mapping. Localization addresses the problem of estimating the pose of the mobile robot given an a priori map and a sequence of sensor measurements. Mapping is the problem of building a map given known poses of the mobile robot and a sequence of sensor measurements.

Solving the problem of localization requires an a priori map of the environment. Unfortunately, such a map is not always available. Blue prints may not exist or be out of date, furniture in an office environment for example may have shifted over time, making the map obsolete. A possible solution is to measure the environment by hand, which is feasible for small-scale environments, but a tedious and time-consuming process for larger environments.

Mapping on the other hand requires a known robot pose. Outdoors, this is possible since GPS (Global Positioning System) is available providing an absolute position around the globe with centimeter range precision in ideal conditions. However, in less good conditions like in forests, valleys or cities, the GPS signal can be disturbed or simply cut off like in tunnels, caves or other indoor environments. In these cases, reliable mapping using GPS cannot be carried out. Indoors, the robot pose is provided by internal sensors like wheel encoders or accelerometers for example. However, these sensors accumulate errors and therefore can only be used reliably over a short period of time / short distances.

A better solution is to fuse both subproblems together and to perform SLAM. By assuming the initial robot pose is known, the robot can start the incremental mapping process by translating its external sensor readings into its map representation. In a next step, the robot moves, predicts its pose using its internal sensors and fuses newly arrived exteroceptive information into the (predicted) map, updating the map and the robot pose at the same time. This cycle is repeated for the whole SLAM experiment.

First methods building the basis of SLAM were presented by [Smith et al., 1990], who established a statistical basis for describing geometric uncertainty and relationships between features or landmarks [Christensen, 2002]. In the beginning of the nineties, it was understood that SLAM is a ”chicken and egg” problem, which cannot be decoupled. This was also the time when the first working solution to the SLAM problem (in two dimensions) were presented, see for example [Smith et al., 1990], [Moutarlier and Chatila, 1989], [Leonard and Durrand-Whyte, 1991]. They were based on the Extended Kalman Filter used in conjunction with a so-called stochastic map presented below.

As already mentioned, the SLAM algorithm follows a repeated two-step procedure. After a movement of the robot, its new pose is first estimated by using the odometry data only, coming from the wheel encoders and in a second step corrected by considering exteroceptive data. This step is often called the prediction whilst the second step consisting of updating the map and the robot pose by integrating newly arrived sensor measurements is called correction or fusion. The SPmap, which is the above-mentioned stochastic map formulated within the SPmodel, and the prediction and correction step of a SLAM cycle are detailed below.

Note that the presented SLAM algorithm (see figure 5.2 for an overview) matches closely the algorithm described in [Castellanos and Tardós, 1999], which is the standard Extended Kalman Filter adapted to the SPmodel. This work differs in the sense that it implements the algorithm in full 3D space using planar segments as underlying features.

5.2 Related Work

Several other research groups use a 2D laser scanner on a rotating support (see [Newman et al., 2006], [Nüchter and Surmann, 2004], [Kohlhepp et al., 2004], [Hähnel et al., 2003]) to produce 3D data. The majority of the SLAM approaches are scan-alignment approaches related to the Iterative Closest Point (ICP) algorithm. It has the advantage that it solves the data association problem automatically as it directly tries to find corresponding points in the raw data. However, due to its iterative nature it can be slow and generally leads to inconsistent maps, when no off-line global alignment is performed as for example presented in [Surmann et al., 2003]. A second drawback is that resulting maps are very complex as they can be composed of up to several million data points, limiting their applicability to higher-level robotic tasks like global localization, scene understanding or path-planning. The approach presented in this work in comparison outputs compact and consistent maps in a more efficient way and also provides useful information to address the latter issues.

[Horn and Schmidt, 1995a] presented early work on using 3D data for robot navigation, extracting vertical planar features to correct the vehicle pose in 2D. [Sequeira et al., 1999] use a single point laser mounted on a pan-tilt unit to create 3D models of indoor scenes. They use an ICP-like algorithm for scan alignment and focus on environment reconstruction and next-view planning rather than 3D navigation. As already mentioned, [Surmann et al., 2003] presented a 3D SLAM approach, based on the ICP algorithm minimizing a global error measure to keep consistency. [Andreasson et al., 2005] also use the ICP to create a 3D map and exploit added intensity information to improve subsequent planar segmentation. [Kohlhepp et al., 2004] presented a 3D navigation approach using a laser scanner continuously rotating around its central optical axis. They match scans using bounded planes in an iterative way, track the robot pose with an EKF and build local submaps rather than a single stochastic map as it is done in this work. This work extends [Weingarten and Siegwart, 2005] to process planar segments.

5.3 The SPmap

A unified way of representing the pose of the mobile robot along with the features of the map of its environment is to use the concept of the so-called *stochastic map*. It was introduced by [Smith et al., 1990] and simply consists of a state vector including all feature locations, the robot pose and the associated covariance matrix. Within the scope of the SPmodel, it is given by

$$\mathbf{SPmap} = (\hat{\mathbf{x}}^W, \hat{\mathbf{p}}^W, \mathbf{C}^W, \mathbf{B}^W) \tag{5.1}$$

with

$$\hat{\mathbf{x}}^W = \begin{bmatrix} \hat{\mathbf{x}}_{WR} \\ \hat{\mathbf{x}}_{WF_1} \\ \hat{\mathbf{x}}_{WF_2} \\ \vdots \\ \hat{\mathbf{x}}_{WF_{N_F}} \end{bmatrix}, \tag{5.2}$$

$$\hat{\mathbf{p}}^W = \begin{bmatrix} \hat{\mathbf{p}}_{R} \\ \hat{\mathbf{p}}_{F_1} \\ \hat{\mathbf{p}}_{F_2} \\ \vdots \\ \hat{\mathbf{p}}_{F_{N_F}} \end{bmatrix}, \tag{5.3}$$

$$\mathbf{C}^W = \begin{bmatrix} \mathbf{C}_R & \mathbf{C}_{RF_1} & \mathbf{C}_{RF_2} & \cdots & \mathbf{C}_{RF_{N_F}} \\ \mathbf{C}_{RF_1}^T & \mathbf{C}_{F_1} & \mathbf{C}_{F_1F_2} & \cdots & \mathbf{C}_{F_1F_{N_F}} \\ \mathbf{C}_{RF_2}^T & \mathbf{C}_{F_1F_2}^T & \mathbf{C}_{F_2} & \cdots & \mathbf{C}_{F_2F_{N_F}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{RF_{N_F}}^T & \mathbf{C}_{F_1F_{N_F}}^T & \mathbf{C}_{F_2F_{N_F}}^T & \cdots & \mathbf{C}_{F_{N_F}} \end{bmatrix} \tag{5.4}$$

and

$$\mathbf{B}^W = diag(\mathbf{I}_6, \mathbf{B}_{F_1}, \mathbf{B}_{F_2}, \ldots, \mathbf{B}_{F_{N_F}}). \tag{5.5}$$

$\mathbf{x}^W \in \mathbb{R}^{6\cdot(1+N_F)\times 1}$ is the estimated location vector of the SPmap, which represents the estimated location of the mobile robot and the map features w.r.t. the base reference W. $\hat{\mathbf{p}}^W \in \mathbb{R}^{rank(\mathbf{B}^W)\times 1}$ is the perturbation vector of the SPmap, $\mathbf{C}^W \in \mathbb{R}^{\left(rank(\mathbf{B}^W)\times rank(\mathbf{B}^W)\right)}$ is the covariance matrix of the SPmap and $\mathbf{B}^W \in \mathbb{R}^{\left(6+\sum_{i=1}^{N_F} rank(\mathbf{B}_{F_i})\right)\times 6}$ its binding matrix. Note that N_F is the number of features in the map.

The cross correlation between the robot and the features $\mathbf{C}_{RF_i}$, $i = 1 \ldots N_F$ relate the error of the robot pose to the error in the feature locations. This way, a correction of the robot pose can update the location of a far-off feature, even if the robot cannot see the latter from its current position.

5.4 Displacement of the Mobile Robot

In this work, the robot moves in a stop-and-go manner, solely taking 3D scans while standing still. Between two 3D scans taken at two consecutive points[1] $\mathbf{x}_{WR_{k-1}}$ and $\mathbf{x}_{WR_k}$ of the robot trajectory, an initial estimation of the displacement $\mathbf{x}_{R_{k-1}R_{k^-}}$ is given by the wheel encoders. As the robot has differential-drive kinematics, the odometry relies on the commonly used piecewise approximation of the displacement of the robot wheels in

---

[1] in this case a point refers to a point in time on a continuous robot trajectory. It is therefore equivalent to a certain robot pose

Cartesian space.

$$\begin{aligned}
\hat{\mathbf{x}}_{WR_{k^-}} = \begin{bmatrix} x_{k^-} \\ y_{k^-} \\ z_{k^-} \\ \phi_{k^-} \\ \theta_{k^-} \\ \psi_{k^-} \end{bmatrix} &= f(\hat{\mathbf{x}}_{WR_{k-1}}, \mathbf{u}_k) \\
&= \hat{\mathbf{x}}_{WR_{k-1}} + \hat{\mathbf{x}}_{R_{k-1}R_{k^-}} \qquad (5.6) \\
&= \hat{\mathbf{x}}_{WR_{k-1}} + \begin{bmatrix} (s_l + s_r)/2 \cdot \cos(\phi_{k-1} + \frac{s_l - s_r}{2b}) \\ (s_l + s_r)/2 \cdot \sin(\phi_{k-1} + \frac{s_l - s_r}{2b}) \\ 0 \\ \frac{s_l - s_r}{2b} \\ 0 \\ 0 \end{bmatrix},
\end{aligned}$$

with wheelbase[2] b and $\mathbf{u}_k = (s_l, s_r)$ being the displacement of the left and right wheel, respectively. Note that index k^- is associated to the predicted state at time step k using the odometry information. This is equivalent to the alternative notation used in the literature, where the above $\hat{\mathbf{x}}_{k^-}$ would be written as $\hat{\mathbf{x}}_{k|k-1}$.

Within the SPmodel, the robot displacement is represented by an uncertain location

$$\mathbf{L}_{R_{k-1}R_{k^-}} = (\hat{\mathbf{x}}_{R_{k-1}R_{k^-}}, \hat{\mathbf{d}}_{R_{k^-}}, \mathbf{C}_{R_{k^-}}, \mathbf{I}_6) \qquad (5.7)$$

with the estimated relative displacement vector $\hat{\mathbf{x}}_{R_{k-1}R_{k^-}}$, the error vector $\hat{\mathbf{d}}_{R_{k^-}}$ with associated full-rank covariance matrix $\mathbf{C}_{R_{k^-}}$ representing the uncertainty and the binding matrix which in this case is the identity $\mathbf{I}_6$. Obviously, this kind of odometry doesn't detect slope changes in undulated terrain. However, if the components of the covariance matrix $\mathbf{C}_{R_{k^-}}$ which are not related to the odometry readings are set sufficiently high, the robot pose can still be tracked successfully in slightly inclined terrain. Otherwise, 3D scan alignment using the Iterative Closest Point (ICP) algorithm can be used to simulate a three-dimensional odometry reading. Note that the error of the robot displacement $\mathbf{x}_{R_{k-1}R_{k^-}}$ is modelled as Gaussian distribution with mean $\mathbf{0}$ and covariance $\mathbf{C}_{R_{k^-}}$ and its magnitude is set to be proportional to the travelled distance.

After having read the data coming from odometry, it has to be used to update the

---

[2] the distance between the two actuated wheels

SPmap in the following way: The predicted state vector is given by

$$\hat{\mathbf{x}}^W_{k-} = \begin{bmatrix} \hat{\mathbf{x}}_{WR_{k-}} \\ \hat{\mathbf{x}}_{WF_{1,k-}} \\ \hat{\mathbf{x}}_{WF_{2,k-}} \\ \vdots \\ \hat{\mathbf{x}}_{WF_{N_F,k-}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{WR_{k-1}} \oplus \hat{\mathbf{x}}_{R_{k-1}R_{k-}} \\ \hat{\mathbf{x}}_{WF_{1,k-1}} \\ \hat{\mathbf{x}}_{WF_{2,k-1}} \\ \vdots \\ \hat{\mathbf{x}}_{WF_{N_F,k-1}} \end{bmatrix}, \tag{5.8}$$

the estimated perturbation vector becomes

$$\hat{\mathbf{p}}^W_{k-} = \begin{bmatrix} \mathbf{J}_{R_{k-}R_{k-1}} \hat{\mathbf{d}}_{R_{k-1}} \\ \hat{\mathbf{p}}_{M_{k-}} \end{bmatrix} \tag{5.9}$$

and the new covariance matrix is given by

$$\mathbf{C}^W_{k-} = \begin{bmatrix} \mathbf{J}_{R_{k-}R_{k-1}} \mathbf{C}_{k-1} \mathbf{J}^T_{R_{k-}R_{k-1}} + \mathbf{C}_{R_{k-}} & \mathbf{J}_{R_{k-}R_{k-1}} \mathbf{C}_{RM_{k-1}} \\ \mathbf{C}^T_{RM_{k-1}} \mathbf{J}^T_{R_{k-}R_{k-1}} & \mathbf{C}_{M_{k-1}} \end{bmatrix} \tag{5.10}$$

with

$$\mathbf{C}^W = \begin{bmatrix} \mathbf{C}_R & \mathbf{C}_{RM} \\ \mathbf{C}^T_{RM} & \mathbf{C}_M \end{bmatrix}, \tag{5.11}$$

where M represents all features included in the **SPmap** and R stands for the robot pose. $\mathbf{J}_{R_{k-}R_{k-1}}$ is the Jacobian of the relative transform $\mathbf{x}_{R_{k-}R_{k-1}}$ (see Appendix for details).

5.5 Update

After a robot movement from $\mathbf{x}_{WR_{k-1}}$ to $\mathbf{x}_{WR_{k-}}$, a 3D scan is taken and converted into the feature-based representation as described in chapter 4. The goal of the update step is to find correspondences between these newly observed features and the features within the **SPmap** and update the latter appropriately. The update is carried out by using the Extended Kalman Filter [Welch and Bishop, 2001], [Grewal and Andrews, 1993], [Bar-Shalom and Li, 1993], which requires to solve the data association problem described in the following.

5.5.1 Data Association

One of the most critical problems in feature-based SLAM is the data association or correspondence problem [Thrun, 2002]. It is the problem of finding features in scans taken from different locations that correspond to the same physical entity. The higher the differentiability of the used features, the better is the obtained data association performance. Abstraction levels range from geometric features like points, lines or planes to semantically more significant features combining laser and vision information for high distinctiveness [Lamon et al., 2003]. To solve data association, extensive use of statistical decision theory is required [Bar-Shalom and Li, 1993]. In loose words, this means a metric is needed to compare different features quantitatively, taking into account uncertainty information. The Mahalanobis distance [Mahalanobis, 1936] d is such a metric and is defined by

$$d(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{C}^{-1} (\mathbf{x} - \mu)}. \tag{5.12}$$

d is the Mahalanobis distance of a random vector x to a multivariate normal distribution with mean μ and covariance matrix $\mathbf{C}$. It can also be defined as dissimilarity measure between two random vectors $\mathbf{x}$ and $\mathbf{y}$ of the same distribution with covariance matrix $\mathbf{C}_M = \mathbf{C}_x + \mathbf{C}_y$, yielding

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{C}_M^{-1} (\mathbf{x} - \mathbf{y})}. \tag{5.13}$$

If $\mathbf{x}$ and $\mathbf{y}$ are randomly chosen, $d_M(\mathbf{x}, \mathbf{y})^2$ is a χ^2-variable with r degrees of freedom. To test whether a random vector $\mathbf{x} = (x_1, \ldots, x_{N_D})^T$ matches a vector $\mathbf{y} = (y_1, \ldots, y_{N_D})^T$ with N_D being the number of components of $\mathbf{x}$ or the degrees of freedom, the χ^2-hypothesis test can be carried out by evaluating $d_M(\mathbf{x}, \mathbf{y})^2$. The hypothesis is that $\mathbf{x}$ belongs to the distribution defined by $\mathbf{y}$ and $\mathbf{C}_M$ or vice versa $\mathbf{y}$ belongs to the distribution defined by $\mathbf{x}$ and $\mathbf{C}_M$. It is rejected with significance level $a\%$ (or with confidence level $(100 - a)\%$) if $d_M(\mathbf{x}, \mathbf{y})$ falls into the rejection region $(\chi^2_{r,a}, \infty)$ and is regarded as acceptable otherwise. The threshold value $\chi^2_{r,a}$ is called the $a\%$ significance value of χ^2 with r degrees of freedom[3] and defined in such a way that

$$\int_{\chi^2_{r,a}}^{\infty} \frac{1}{2^{r/2}\Gamma(r/2)} R^{r/2-1} e^{-R/2} dR = \frac{a}{100}, \tag{5.14}$$

where $\Gamma(n) = \int_0^\infty t^{n-1} e^{-t} dt$ is the Gamma function and $R = d_M(\mathbf{x}, \mathbf{y})^2$. Fortunately, the values of the above integral have been precalculated for various degrees of freedom r and significance levels a and can be looked up. The hypothesis is rejected with significance level $a\%$ if

$$R > \chi^2_{r,a}. \tag{5.15}$$

See for example [Kanatani, 1996] for details. To use the χ^2-hypothesis test to compare global map features with new features observed from the robot coordinate frame, the

---

[3] $r = N_D$

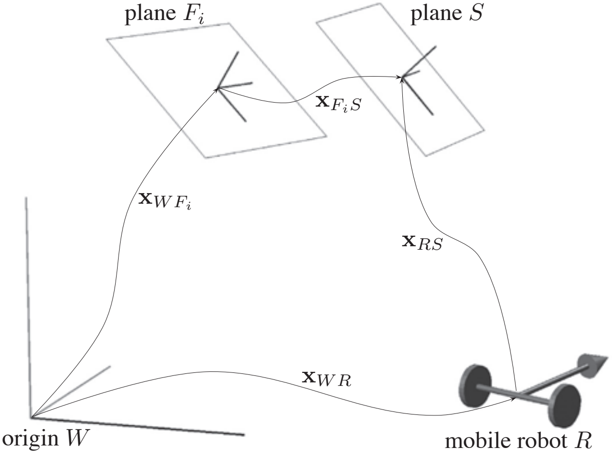


Figure 5.1: Exemplary geometric relations between world origin W, mobile robot R and global map feature F_i which is matched to locally observed feature S. During data association, the relative transform $\mathbf{x}_{F_iS}$ is evaluated for different map features F_i ($i = 1 \dots N_F$) by means of a χ^2-hypothesis test based on the squared Mahalanobis distance metric.

involved covariance matrices have to be transformed into a common reference frame as described below.

5.5.2 Relating geometric entities

The implicit measurement equation

Often in this work, it is aimed at estimating a state vector $\mathbf{x} \in \mathbb{R}^m$ given an observation $\hat{\mathbf{y}} \in \mathbb{R}^n$ of $\mathbf{x}$ subject to noise modelled as additive white Gaussian distribution, yielding

$$\hat{\mathbf{y}} = \mathbf{y} + \mathbf{u}, \qquad \mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}). \tag{5.16}$$

Following [Ayache and Faugeras, 1988], $\mathbf{x}$ and $\mathbf{y}$ can be related by an implicit function

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0}, \tag{5.17}$$

where $\mathbf{f}$ is generally nonlinear due to rotational terms. Using the Taylor expansion series and assuming a ”good” estimate of $\mathbf{x}$ denoted by $\hat{\mathbf{x}}$ is available, $\mathbf{f}$ can be linearized at

$\mathbf{x} = \hat{\mathbf{x}}$ and $\mathbf{y} = \hat{\mathbf{y}}$ as follows:

$$\mathbf{f}(\mathbf{x},\mathbf{y}) = \mathbf{f}(\hat{\mathbf{x}},\hat{\mathbf{y}}) + \underbrace{\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}},\mathbf{y}=\hat{\mathbf{y}}}}_{\mathbf{H}} (\mathbf{x} - \hat{\mathbf{x}}) + \underbrace{\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{x}=\hat{\mathbf{x}},\mathbf{y}=\hat{\mathbf{y}}}}_{\mathbf{G}} (\mathbf{y} - \hat{\mathbf{y}}) \tag{5.18}$$

By rearranging the terms and appropriate substitution, the following linear measurement equation is obtained, which can directly be used for the Extended Kalman Filter as described below:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v}, \qquad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{5.19a}$$

with

$$\mathbf{z} = -\mathbf{f}(\hat{\mathbf{x}},\hat{\mathbf{y}}) + \mathbf{H}\hat{\mathbf{x}}, \tag{5.19b}$$

$$\mathbf{v} = \mathbf{G}(\mathbf{y} - \hat{\mathbf{y}}), \tag{5.19c}$$

$$\mathbf{R} = \mathbf{G}\mathbf{S}\mathbf{G}^T. \tag{5.19d}$$

Paring geometric features within the SPmodel

Let the current robot pose be denoted by location $\mathbf{L}_{WR} = (\hat{\mathbf{x}}_{WR}, \hat{\mathbf{d}}_R, \mathbf{C}, \mathbf{I}_6)$, the newly extracted plane by $\mathbf{L}_{RS} = (\hat{\mathbf{x}}_{RS}, \hat{\mathbf{p}}_S, \mathbf{C}, \mathbf{B})$ and the planes contained in the $\mathbf{SPmap}$ by $\mathbf{L}_{WF_i} = (\hat{\mathbf{x}}_{WF_i}, \hat{\mathbf{p}}_{F_i}, \mathbf{C}, \mathbf{B})$, with $i = 1 \dots N_F$ (see figure 5.1 for an illustration). The problem of data association is to find all corresponding pairings between the newly observed planes $S_j, j = 1 \dots N_S$ and the features of the map. The two planes coincide if

$$\mathbf{B}_{F_iS}\mathbf{x}_{F_iS} = \mathbf{0}. \tag{5.20}$$

In this case, the binding matrix $\mathbf{B}_{F_iS}$ of the pairing corresponds to $\mathbf{B}_{F_i}$ or $\mathbf{B}_S$ as the feature types correspond. $\mathbf{x}_{F_iS} = (x, y, z, \phi, \theta, \psi)^T$, hence $\mathbf{B}_{F_iS}\mathbf{x}_{F_iS} = (z, \theta, \psi)^T = \mathbf{0}$, meaning that the two planes coincide if their three relative parameters z, θ and ψ equal zero.

In the case of simultaneous localization and mapping (SLAM), global features included in the $\mathbf{SPmap}$ are matched to locally observed features in order to correct the map features and track the robot pose at the same time. Hence, the pairing consists of vector $\mathbf{p}^W$, which is to be estimated based on information contained in the current observation denoted by $\mathbf{p}_S$. The related nonlinear implicit measurement equation aligning the two coordinate frames then takes the following form:

$$
\begin{aligned}
\mathbf{f}(\mathbf{p}_i^W, \mathbf{p}_{\mathbf{S}}) &= \mathbf{B}_{F_iS}\mathbf{x}_{F_iS} \\
&= \mathbf{B}_{F_iS}\left(\ominus \mathbf{x}_{WF_i} \oplus \mathbf{x}_{WR} \oplus \mathbf{x}_{WS}\right) \\
&= \mathbf{B}_{F_iS}\left(\ominus(\hat{\mathbf{x}}_{WF_i} \oplus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i}) \oplus \hat{\mathbf{x}}_{WR} \oplus \mathbf{d}_R \oplus \hat{\mathbf{x}}_{RS} \oplus \mathbf{B}_S^T \mathbf{p}_S\right) \\
&= \mathbf{B}_{F_iS}\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iR} \oplus \mathbf{d}_R \oplus \hat{\mathbf{x}}_{RS} \oplus \mathbf{B}_S^T \mathbf{p}_S\right) \\
&= \mathbf{B}_{F_iS}\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right) \\
&= \mathbf{0}
\end{aligned}
\tag{5.21}
$$

Linearization of (5.21) is performed as follows:

$$
\mathbf{h}_i = \mathbf{f}_i(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S) \tag{5.22}
$$

$$
\begin{aligned}
\mathbf{H}_i &= \begin{bmatrix} \mathbf{H}_i^R & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_i^{F_i} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \\
\mathbf{H}_i^R &= \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{d}_R}\right|_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\left[\frac{\partial\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right)}{\partial \mathbf{d}_R}\right]_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\left[\frac{\partial\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right)}{\partial(\mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S)}\right. \\
&\qquad\qquad \left.\cdot \frac{\partial(\mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S)}{\partial(\mathbf{J}_{SR}\mathbf{d}_R)} \cdot \frac{\partial(\mathbf{J}_{SR}\mathbf{d}_R)}{\partial \mathbf{d}_R}\right]_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\mathbf{J}_{2\oplus}\{\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS}, \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\} \\
&\qquad\qquad \cdot \mathbf{J}_{1\oplus}\{\mathbf{J}_{SR}\mathbf{d}_R, \mathbf{B}_S^T \mathbf{p}_S\} \cdot \mathbf{J}_{SR}
\end{aligned}
\tag{5.23}
$$

$$
\begin{aligned}
\mathbf{H}_i^{F_i} &= \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_{F_i}}\right|_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\left[\frac{\partial\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right)}{\partial \mathbf{p}_{F_i}}\right]_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\left[\frac{\partial\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right)}{\partial(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i})}\right. \\
&\qquad\qquad \left.\cdot \frac{\partial(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i})}{\partial(\mathbf{B}_{F_i}^T \mathbf{p}_{F_i})} \cdot \frac{\partial(\mathbf{B}_{F_i}^T \mathbf{p}_{F_i})}{\partial \mathbf{p}_{F_i}}\right]_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\mathbf{J}_{1\oplus}\{\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i}, \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\} \cdot \mathbf{J}_{\ominus}\{\mathbf{B}_{F_i}^T \mathbf{p}_{F_i}\}\mathbf{B}_{F_i}^T
\end{aligned}
\tag{5.24}
$$

$$
\begin{aligned}
\mathbf{G}_i &= \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_S}\right|_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\left[\frac{\partial\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right)}{\partial \mathbf{p}_S}\right]_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\left[\frac{\partial\left(\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R \oplus \mathbf{B}_S^T \mathbf{p}_S\right)}{\partial (\mathbf{B}_S^T \mathbf{p}_S)} \cdot \frac{\partial(\mathbf{B}_S^T \mathbf{p}_S)}{\partial \mathbf{p}_S}\right]_{(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)} \\
&= \mathbf{B}_{F_iS}\mathbf{J}_{2\oplus}\{\ominus \mathbf{B}_{F_i}^T \mathbf{p}_{F_i} \oplus \hat{\mathbf{x}}_{F_iS} \oplus \mathbf{J}_{SR}\mathbf{d}_R, \mathbf{B}_S^T \mathbf{p}_S\}\mathbf{B}_S^T
\end{aligned} \tag{5.25}
$$

Assuming the involved perturbation vectors are centered, i.e. $\hat{\mathbf{p}}_{F_i} = \mathbf{0}$, $\hat{\mathbf{p}}_S = \mathbf{0}$, $\hat{\mathbf{d}}_R = \mathbf{0}$ and the features are all planes, these expressions reduce to:

$$\mathbf{h}_i = \mathbf{f}_i(\hat{\mathbf{p}}^W, \hat{\mathbf{p}}_S)$$

$$\mathbf{H}_i = \begin{bmatrix}\mathbf{H}_i^R & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_i^{F_i} & \mathbf{0} & \dots & \mathbf{0}\end{bmatrix} \tag{5.26}$$

$$\mathbf{H}_i^R = \mathbf{B}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{F_iS}, \mathbf{0}\}\mathbf{J}_{SR} \tag{5.27}$$

$$\mathbf{H}_i^{F_i} = -\mathbf{B}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{F_iS}\}\mathbf{B} \tag{5.28}$$

$$\mathbf{G}_i = \mathbf{B}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{F_iS}, \mathbf{0}\}\mathbf{B} \tag{5.29}$$

with

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.30}$$

The squared Mahalanobis distance between global plane F_i and newly observed plane S is then given by

$$D^2 = \mathbf{h}_i^T\left(\mathbf{H}_i^R \mathbf{C}_R (\mathbf{H}_i^R)^T + \mathbf{H}_i^{F_i}\mathbf{C}_{F_i}(\mathbf{H}_i^{F_i})^T + \mathbf{G}_i \mathbf{C}_S \mathbf{G}_i^T\right)^{-1}\mathbf{h}_i. \tag{5.31}$$

The hypothesis based on the square Mahalanobis distance D^2 is considered true with a significance level α, if

$$D^2 \leq \chi^2_{r,\alpha}. \tag{5.32}$$

$\chi^2_{r,\alpha}$ is a threshold value that can be obtained from the χ^2-distribution with rank 3, α is the probability of rejecting a good match. If the candidate pairing is accepted, the two planes F_i and S are fused. Whenever multiple matching candidates are available, a nearest-neighbor strategy is applied.

Note that in the above terms, the time index k has been omitted for clarity. In the following it has to be included. The above $\mathbf{H}_i^R$ is for example equivalent to the $\mathbf{H}_{i,k}^R$ below.

5.5.3 Fusion

The data association step divides newly observed features into unpaired features that couldn't be matched to any map feature and successfully paired features. The goal of the fusion step is to estimate the new robot pose using these successful pairings as well as the predicted robot pose. This fusion is carried out in a probabilistic way using the Extended Kalman Filter. There are different possibilities of fusing the new information into the $\mathbf{SPmap}$. Either include information of all feature pairings at the same time or in an iterative way. In this work, the iterative way of fusing information has been implemented, as this allows to refine the data association after every iteration.

For every found pairing i between a newly observed feature and an existing map feature, the perturbation vector of the $\mathbf{SPmap}$ is updated in the following way:

$$\mathbf{p}_{i,k}^{W} = \mathbf{p}_{i,k^-}^{W} + \underbrace{\mathbf{K}_{i,k}}_{gain} \underbrace{\left(\mathbf{z}_{i,k} - \mathbf{H}_{i,k}\hat{\mathbf{p}}_{i,k^-}^{W}\right)}_{innovation} \tag{5.33}$$

$$= \hat{\mathbf{p}}_{i,k^-}^{W} - \mathbf{K}_{i,k}\mathbf{h}_{i,k}, \tag{5.34}$$

This is the fusion step of the Extended Kalman Filter, where the predicted state vector $\mathbf{p}_{i,k^-}^{W}$ is corrected by adding the innovation weighted by the so-called Kalman gain $\mathbf{K}_{i,k}$. The covariance matrix is updated as follows:

$$\mathbf{C}_{i,k}^{W} = \left(\mathbf{I} - \mathbf{K}_{i,k}\mathbf{H}_{i,k}\right)\mathbf{C}_{i,k^-}^{W}. \tag{5.35}$$

The Kalman gain defines the weighting of the innovation in relation to the underlying uncertainties encoded by the covariance matrices $\mathbf{C}_{i,k}^{W}$ and $\mathbf{S}_{i,k}^{W}$. A small uncertainty in the newly observed feature defined by covariance matrix $\mathbf{S}_{i,k}$ will lead to a Kalman gain close to the identity matrix and hence strongly take into account the new observation. On the other hand, a less certain newly observation described by larger components of the covariance matrix $\mathbf{S}_{i,k}$ leads to a smaller Kalman gain and hence gives the predicted state the higher weight. The Kalman gain $\mathbf{K}_{i,k}$ is defined by

$$\mathbf{K}_{i,k} = \mathbf{C}_{i,k^-}^{W}\mathbf{H}_{i,k}^{T}\left(\mathbf{H}_{i,k}\mathbf{C}_{i,k^-}^{W}\mathbf{H}_{i,k}^{T} + \mathbf{G}_{i,k}\mathbf{S}_{i,k}\mathbf{G}_{i,k}^{T}\right)^{-1} \tag{5.36}$$

$\mathbf{C}_{i,k}^{F_i}$ is the uncertainty covariance matrix of the plane w.r.t. its local reference frame. Note that the state vector of the $\mathbf{SPmap}$ is centered after each iteration as explained below .

5.5.4 Adding Non-Matched Features to the Map

All newly observed features that couldn't be matched to any global map feature contained in the $\mathbf{SPmap}$ have to be added to the latter. It is assumed that the $\mathbf{SPmap}$ has the following form:

$$\hat{\mathbf{x}}^W = \begin{bmatrix}\hat{\mathbf{x}}_{WR}\\ \hat{\mathbf{x}}_{WM}\end{bmatrix}, \qquad \hat{\mathbf{p}}^W = \begin{bmatrix}\hat{\mathbf{d}}_R\\ \hat{\mathbf{p}}_M\end{bmatrix}, \qquad \mathbf{C}^W = \begin{bmatrix}\mathbf{C}_R & \mathbf{C}_{RM}\\ \mathbf{C}_{RM}^T & \mathbf{C}_M\end{bmatrix} \tag{5.37}$$

An unpaired feature F is added to the $\mathbf{SPmap}$, using the following terms:

$$(\hat{\mathbf{x}}^W)' = \begin{bmatrix}\hat{\mathbf{x}}_{WR}\\ \hat{\mathbf{x}}_{WM}\\ \hat{\mathbf{x}}_{WR} \oplus \hat{\mathbf{x}}_{RF}\end{bmatrix} \tag{5.38}$$

$$(\mathbf{p}^W)' = \begin{bmatrix}\mathbf{d}_R\\ \mathbf{p}_M\\ \mathbf{B}_F\mathbf{J}_{FR}\mathbf{d}_R + \mathbf{p}_E\end{bmatrix} \tag{5.39}$$

$$(\mathbf{C}^W)' = \begin{bmatrix}\mathbf{C}_R & \mathbf{C}_{RM} & \mathbf{C}_R\mathbf{J}_{FR}^T\mathbf{B}_F^T\\ \mathbf{C}_{RM}^T & \mathbf{C}_M & \mathbf{C}_{RM}^T\mathbf{J}_{FR}^T\mathbf{B}_F^T\\ \mathbf{B}_F\mathbf{J}_{FR}\mathbf{C}_R & \mathbf{B}_F\mathbf{J}_{FR}\mathbf{C}_{RM} & \mathbf{B}_F\mathbf{J}_{FR}\mathbf{C}_R\mathbf{J}_{FR}^T\mathbf{B}_F^T + \mathbf{C}_F\end{bmatrix} \tag{5.40}$$

5.5.5 Centering

In order to keep consistency, after integrating each feature into the $\mathbf{SPmap}$, it is centered which simplifies related mathematical terms [Castellanos and Tardós, 1999]. Informally, centering the $\mathbf{SPmap}$ means transforming all features and the robot pose in a way that their perturbation vectors become zero. This is achieved by compounding the location vector $\hat{\mathbf{x}}^W$ with the perturbation vector $\hat{\mathbf{p}}^W$ and propagating the uncertainty encoded by $\mathbf{C}^W$. The centered map is given by

$$(\hat{\mathbf{x}}^W)' = \begin{bmatrix}\hat{\mathbf{x}}'_{WR}\\ \hat{\mathbf{x}}'_{WF_1}\\ \hat{\mathbf{x}}'_{WF_2}\\ \vdots\\ \hat{\mathbf{x}}'_{WF_{N_F}}\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{x}}_{WR} \oplus \hat{\mathbf{d}}_R\\ \hat{\mathbf{x}}_{WF_1} \oplus \mathbf{B}_{F_1}^T\hat{\mathbf{p}}_{F_1}\\ \hat{\mathbf{x}}_{WF_2} \oplus \mathbf{B}_{F_2}^T\hat{\mathbf{p}}_{F_2}\\ \vdots\\ \hat{\mathbf{x}}_{WF_{N_F}} \oplus \mathbf{B}_{F_{N_F}}^T\hat{\mathbf{p}}_{F_{N_F}}\end{bmatrix} \tag{5.41}$$

$$(\mathbf{p}^W)' = \mathbf{0} \tag{5.42}$$

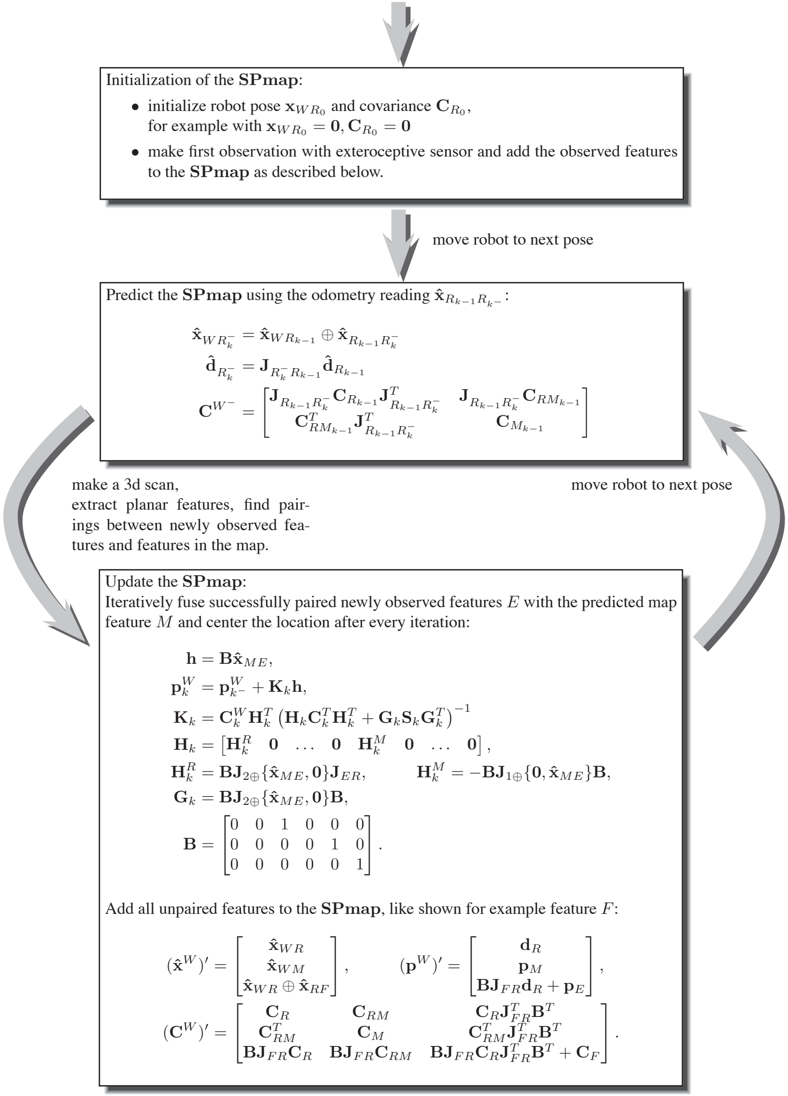
$$(\mathbf{C}^W)' = \mathbf{Q}^W\mathbf{C}^W(\mathbf{Q}^W)^T \tag{5.43}$$

with

$$\begin{aligned}\mathbf{Q}^W = diag(&\mathbf{J}_{2\oplus}^{-1}\{\mathbf{d}_R, \mathbf{0}\}, \mathbf{B}_{F_1}\mathbf{J}_{2\oplus}^{-1}\{\mathbf{B}_{F_1}^T\mathbf{p}_{F_1}, \mathbf{0}\}\mathbf{B}_{F_1}^T, \ldots,\\ &\mathbf{B}_{F_{N_F}}\mathbf{J}_{2\oplus}^{-1}\{\mathbf{B}_{F_{N_F}}^T\mathbf{p}_{F_{N_F}}, \mathbf{0}\}\mathbf{B}_{F_{N_F}}^T).\end{aligned} \tag{5.44}$$

5.6 Summary

This chapter presented the EKF-based SLAM algorithm in detail. It is an extension of the algorithm presented by Castellanos et al. [**?**] to three dimensions leading to six degrees of freedom in the robot pose. After initialization of the stochastic map called the SPmap, the robot gathers new knowledge by moving in a stop-and-go manner. The robot displacement is first estimated by the data collected by the wheel encoders and then fused to the 3D data extracted as planar features from the point cloud generated during standstill. This process is governed by the Extended Kalman Filter (EKF) which estimated the displacement of the robot and the reconstructed environment features in a suboptimal manner considering involved uncertainties modelled by Gaussian distributions. The next chapter presents real-world results using this algorithm.



Initialization of the **SPmap**:

- initialize robot pose $\mathbf{x}_{WR_0}$ and covariance $\mathbf{C}_{R_0}$, for example with $\mathbf{x}_{WR_0} = \mathbf{0}, \mathbf{C}_{R_0} = \mathbf{0}$
- make first observation with exteroceptive sensor and add the observed features to the **SPmap** as described below.

move robot to next pose

Predict the **SPmap** using the odometry reading $\hat{\mathbf{x}}_{R_{k-1}R_{k-}}$:

$$\begin{aligned}
\hat{\mathbf{x}}_{WR_k^-} &= \hat{\mathbf{x}}_{WR_{k-1}} \oplus \hat{\mathbf{x}}_{R_{k-1}R_k^-} \\
\hat{\mathbf{d}}_{R_k^-} &= \mathbf{J}_{R_k^- R_{k-1}} \hat{\mathbf{d}}_{R_{k-1}} \\
\mathbf{C}^{W^-} &= \begin{bmatrix} \mathbf{J}_{R_{k-1}R_k^-} \mathbf{C}_{R_{k-1}} \mathbf{J}^T_{R_{k-1}R_k^-} & \mathbf{J}_{R_{k-1}R_k^-} \mathbf{C}_{RM_{k-1}} \\ \mathbf{C}^T_{RM_{k-1}} \mathbf{J}^T_{R_{k-1}R_k^-} & \mathbf{C}_{M_{k-1}} \end{bmatrix}
\end{aligned}$$

make a 3d scan, extract planar features, find pairings between newly observed features and features in the map.

move robot to next pose

Update the **SPmap**:
Iteratively fuse successfully paired newly observed features E with the predicted map feature M and center the location after every iteration:

$$\begin{aligned}
\mathbf{h} &= \mathbf{B}\hat{\mathbf{x}}_{ME}, \\
\mathbf{p}_k^W &= \mathbf{p}_{k^-}^W + \mathbf{K}_k \mathbf{h}, \\
\mathbf{K}_k &= \mathbf{C}_k^W \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{C}_k^T \mathbf{H}_k^T + \mathbf{G}_k \mathbf{S}_k \mathbf{G}_k^T \right)^{-1} \\
\mathbf{H}_k &= \begin{bmatrix} \mathbf{H}_k^R & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_k^M & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}, \\
\mathbf{H}_k^R &= \mathbf{B}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{ME}, \mathbf{0}\}\mathbf{J}_{ER}, \qquad \mathbf{H}_k^M = -\mathbf{B}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{ME}\}\mathbf{B}, \\
\mathbf{G}_k &= \mathbf{B}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{ME}, \mathbf{0}\}\mathbf{B}, \\
\mathbf{B} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

Add all unpaired features to the **SPmap**, like shown for example feature F:

$$(\hat{\mathbf{x}}^W)' = \begin{bmatrix} \hat{\mathbf{x}}_{WR} \\ \hat{\mathbf{x}}_{WM} \\ \hat{\mathbf{x}}_{WR} \oplus \hat{\mathbf{x}}_{RF} \end{bmatrix}, \qquad (\mathbf{p}^W)' = \begin{bmatrix} \mathbf{d}_R \\ \mathbf{p}_M \\ \mathbf{B}\mathbf{J}_{FR}\mathbf{d}_R + \mathbf{p}_E \end{bmatrix},$$

$$(\mathbf{C}^W)' = \begin{bmatrix} \mathbf{C}_R & \mathbf{C}_{RM} & \mathbf{C}_R \mathbf{J}_{FR}^T \mathbf{B}^T \\ \mathbf{C}_{RM}^T & \mathbf{C}_M & \mathbf{C}_{RM}^T \mathbf{J}_{FR}^T \mathbf{B}^T \\ \mathbf{B}\mathbf{J}_{FR}\mathbf{C}_R & \mathbf{B}\mathbf{J}_{FR}\mathbf{C}_{RM} & \mathbf{B}\mathbf{J}_{FR}\mathbf{C}_R \mathbf{J}_{FR}^T \mathbf{B}^T + \mathbf{C}_F \end{bmatrix}.$$

Figure 5.2: Overview of the Extended Kalman Filter equations implementing the SLAM algorithm used in this work. Note that after the initialization of the SPmap, the prediction and update cycle is repeated until the robot stops. Refer to the the text for details.

Chapter 6

Experimental Results

6.1 Introduction

The experiments have been carried out with a differential-drive wheeled robot (see figure 6.1) in the office environment of our lab. The only used sensors are the robot's 1 kHz wheel encoders and the rotating laser scanner presented in chapter 2 mounted on the top. During the different experiments, the robot moves in a stop-and-go manner to allow the generation of consistent 3D scans while the robot stands still. A continuously moving robot would unnecessarily complicate the 3D point registration process. After first tests using a simulation, the performance of the SLAM algorithm is analyzed based on infinite planes in a qualitative way and in a second step compared to the result using planar segment information.

Even though the test environment is flat almost everywhere with the exception of several small ramps, it is believed that it is suitable to validate a 3D SLAM algorithm, if all six degrees of freedom of the robot are considered during the estimation. Furthermore, a flat environment has the advantage that the z-component of the reconstructed map can quickly be compared to the ground truth which is known to be constant at almost all places (except at the ramps). In the experiment with the loop, the z-component was used to evaluate the improvement of using planar segment information for the data association step of the SLAM algorithm. An experimental validation in a non-flat environment would require three-dimensional ground truth information, which may be difficult to obtain and, even more importantly, also require a three-dimensional odometry sensor. As presented below in the experiment with the ramp, the ICP algorithm can be used to simulate a full three-dimensional odometry.

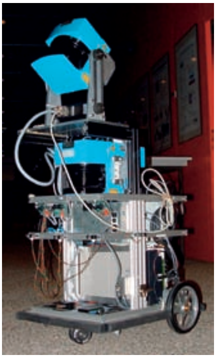


Figure 6.1: The robot used for the experiments is a differential-drive mobile robot equipped with a multitude of sensors: two opposing horizontal Sick LMS 200 laser range scanners, four ultrasound, five infrared distance sensors and a 1000 Hz odometry controller. Only the latter is used for this work as well as the added 3D scanning system mounted on the top. As mentioned earlier in the text, it is composed of another Sick LMS 200 laser scanner mounted on a rotating support. It generates up to 216961 data points (601 2D scans) per scan at an angular range of 270 degrees.

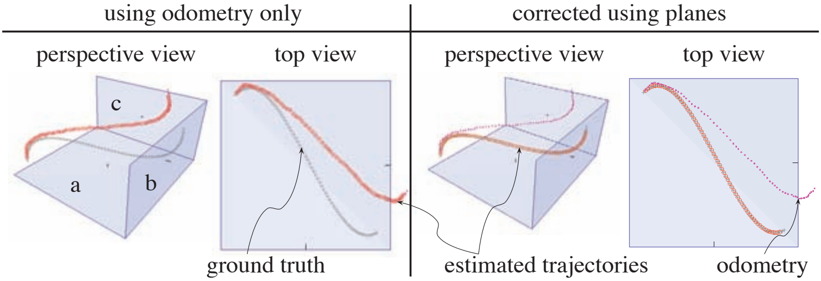
6.2 Simulation

A simulation provides means to compare the result of the SLAM algorithm to ground-truth, which is rarely available in the real world. A simple idealized environment composed of three orthogonal planes was used representing the minimal environment necessary to update all six degrees of freedom of the robot state vector $\mathbf{x}_{WR} = (x, y, z, \phi, \theta, \psi)^T$. The robot performs a sequence of 60 movements with a simulated odometry perturbed by statistic and systematic errors. Figure 6.2 shows a visualization of the above-mentioned environment as well as the ground-truth path depicted by gray crosses, the odometry data (magenta dots) and the corrected path of the robot (red circles). It can be seen that if the exteroceptive data is used (right side), the robot follows closely the ground truth trajectory. To show the functioning of the algorithm, a table was included allowing to compare the reconstructed planes with the ground-truth in a quantitative[1] way. Note that the performance of the algorithms depends on the modelled motion and observation errors. The quantitative error analysis 6.3 is therefore by no means to be viewed as absolute. It merely provides a concrete example of a quantitative analysis of the localization errors with respect to ground-truth. It can be observed that all six components of the robot pose stay close to the ground-truth and the estimated error stays bounded.

This experiment shows the performance of the Extended Kalman Filter in 3D in an example case with greatly simplified data association as the 3 involved planes are perpendicular w.r.t. each other and therefore simple to discern. In a real-world case, the data association is a more critical issue as dozens of planar features have to be discerned correctly. Nevertheless, after many other simulated robot experiments, it can be concluded, that the presented feature-based SLAM algorithm based on (infinite) planes works in 3D space.

---

[1] note that the performance of the algorithm depends on the error models and the step size used, hence the numerical values should merely be seen as an example



plane	rec. normal	rec. distance	normal	distance
a	$(-0.0010, -0.0003, 1.0000)^T$	-0.00059107	$(0, 0, 1)^T$	0
b	$(-0.0006, 1.0000, 0.0006)^T$	-2.9982	$(0, 1, 0)^T$	-3
c	$(-1.0000, -0.0001, -0.0004)^T$	-2.9988	$(-1, 0, 0)^T$	-3

Figure 6.2: The idealized simulation environment used to perform initial tests. The robot moves 60 steps along a three-dimensional simulated trajectory. The estimated trajectory is marked by red circles. Note that the robot closely follows the ground truth on the right side, when considering the information coming from the extracted planes. See figure 6.3 for the associated error analysis. The table shows the values of the 3 reconstructed planes (a,b,c) in comparison to the ground-truth.

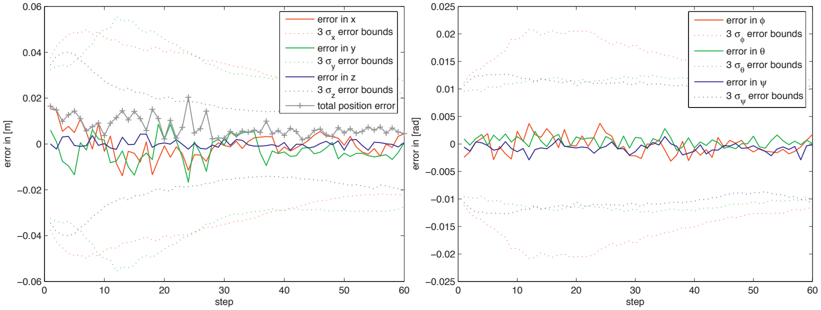


Figure 6.3: The evolution of localization errors during the simulated SLAM experiment (see also figure 6.2). Note that all components of the estimated robot pose stay close to the ground truth and the uncertainty stays bounded.

6.3 Using Infinite Planes

6.3.1 Corridor

In a first set of real experiments, the mobile robot moved along a corridor covering a distance of 34m entering two rooms on its way. Forty 3D scans were taken in a stop-and-go manner, one approximatively every meter. The scans are composed of 601 2D scans of 361 data points each, covering an angular field of view of $270^\circ \times 180^\circ$ and an angular resolution of around 0.5° in both horizontal and vertical direction. Infinite planes are extracted using the grid-based algorithm (GBS) described in chapter 4.

Figure 6.4 shows the reconstruction of the environment using odometry information only. The raw 3D scans were subsampled to around 3000 data points each for faster rendering. As expected, the robot accumulates an error during dead-reckoning leading to an inconsistent map. Especially the accumulated rotational error is easily visible as the reconstructed map is bent to the left.

A possibility of correcting the dead-reckoning error is to use scan alignment based on the Iterative Closest Point (ICP) algorithm (see [Besl and McKay, 1992]. It proceeds by iteratively finding closest point pairs in different point clouds and minimizes their relative rigid body transform. It therefore solves the data association problem in an iterative way until it converges. Figure 6.5 shows the result of using ICP with the same data set as above. It is applied in a pairwise manner, not in a global way, which is generally done off-line [Surmann et al., 2003]. Hence, an accumulated error persists, illustrated by the offset between the dashed vertical line depicting the wall of the corridor in an ideal case and the reconstructed corridor. Methods addressing the global consistency using ICP are not further considered here.

Figure 6.6 shows the result of using the EKF-SLAM algorithm presented above. The features used are infinite planes. In comparison to figure 6.5, the remaining accumulated error seems to be negligible. However, in the upper part of the image, the plotted scans are less accurately aligned. Figure 6.7 shows the associated evolution of the robot pose components with their uncertainty (as 10σ error bounds). It can be observed that the error doesn't grow monotonically, but slightly decreases at two points in time, around step 15 and step 29. This represents the moment when the robot moves out of a room back into a known environment. The robot therefore relocalizes itself, accommodating the uncertainty level it had before entering the room. Figure 6.8 shows the evolution of pairings between newly observed features and existing map features. Note that when entering a room or a new area at steps 11 or 26 for example, the number of newly observed features is relatively high, whereas the number of successfully paired features is low. On the other hand, after leaving the room and reobserving a known area of the map, for example at step 15 and 29, the number of successfully paired features is high, whereas the number of new features is relatively low. Figure 6.9 shows the evolution of the trace of the covariance of a subset of features. The graphs are characteristic for the EKF-SLAM algorithm and reflect that as the robot gathers more information on its way, the uncertainty of the reobserved features decreases monotonically.

6.3.2 A long corridor

In a second experiment, the robot travelled through the complete corridor of our lab, covering a distance of 140m. The goal of this experiment was to check the consistency of the built map in a larger scale. Figure 6.11 shows the reconstructed map using the EKF-SLAM approach overlayed onto a building plan. It can be observed, that the reconstructed map composed of 244 planes matches closely the reality.

6.3.3 A ramp

In this experiment, partial data of the above experiment with the long corridor has been reused and analyzed in more detail. It shows a small ramp of a height of around 0.2m and a length of 1.3m. As shown in figure 6.13, the EKF-SLAM algorithm is able to estimate the resulting three-dimensional robot trajectory. However, as the wheel encoders don't provide information about slope changes, the ICP algorithm [Besl and McKay, 1992] was used to generate a ”simulated” 3D odometry.

6.3.4 A Loop

In the third experiment, a loop was chosen of a size of approximatively 10×12 meters. The robot was moved almost three times around the inner structure, covering a total distance of 99m. Figure 6.14 shows several results. Image a) shows the estimated map using odometry information only. As expected, the accumulated odometry error leads to inconsistent maps. Image b) shows the result of using the (pairwise) ICP algorithm to improve odometry. As mentioned above, this provides means of generating a true 3D odometry with six degrees of freedom which can be useful when the appropriate proprioceptive 3D sensor is unavailable. The resulting map is significantly more consistent than the map in image b), but still shows some misalignments due to the persisting accumulated error. Image c) shows the resulting map using the EKF-SLAM approach overlayed onto a building plan. It can be observed, that in comparison to the map represented in image b), the misalignments disappeared.

6.3.5 Discussion

These results represent a first validation of the feature-based 3D EKF-SLAM algorithm using infinite planes. It shows similar behavior to its two-dimensional counterpart based on infinite lines (see for example [Arras, 2003], [Castellanos et al., 1999]), that are its existential dependence on a working data association, its ability to build globally consistent maps in an incremental way and its high performance, as a single SLAM cycle takes

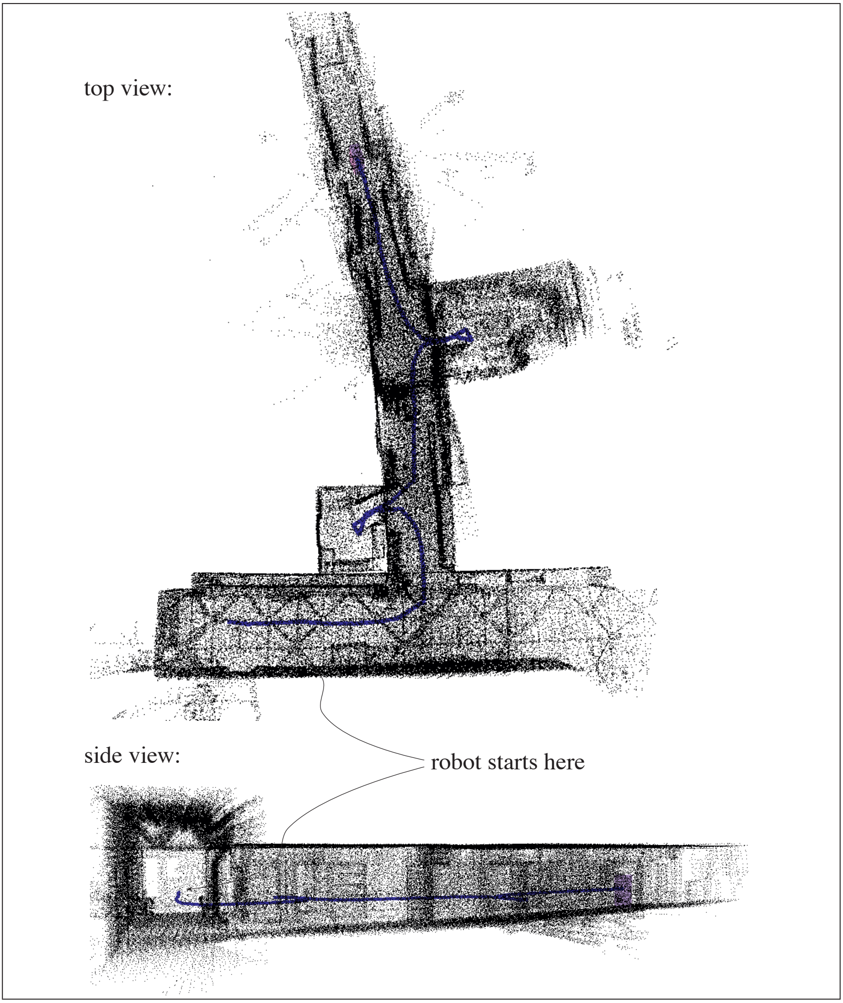


Figure 6.4: Estimating the robot pose and the map using odometry information only. The robot starts at the end of a corridor on the left and and moves 40 steps covering a distance of 34m entering two rooms on its way. The solid line represents the robot trajectory, while the black dots are subsampled raw scans generated by the 3D laser scanner. Note the characteristic odometry error accumulation leading to a curved map.

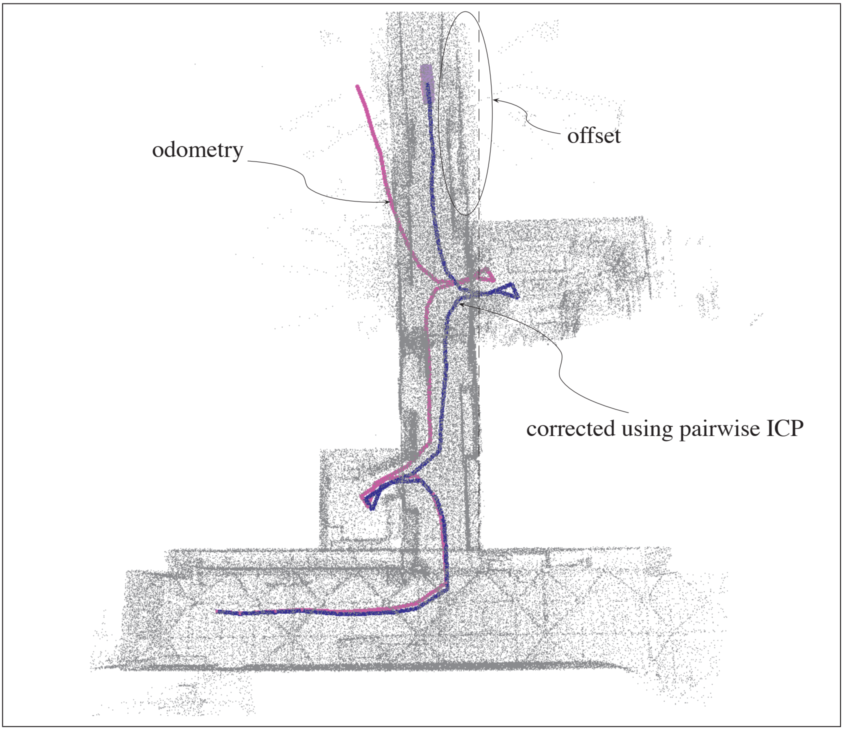


Figure 6.5: The same experiment as in figure 6.4, but additionally using the Iterative Closest Point (ICP) algorithm for pairwise scan alignment. Note that the resulting map (depicted in blue) is more consistent than the map using odometry information (depicted in magenta) only, however an accumulated error remains, which leads to a slightly curved corridor (see indicated offset between dashed line and reconstructed map)

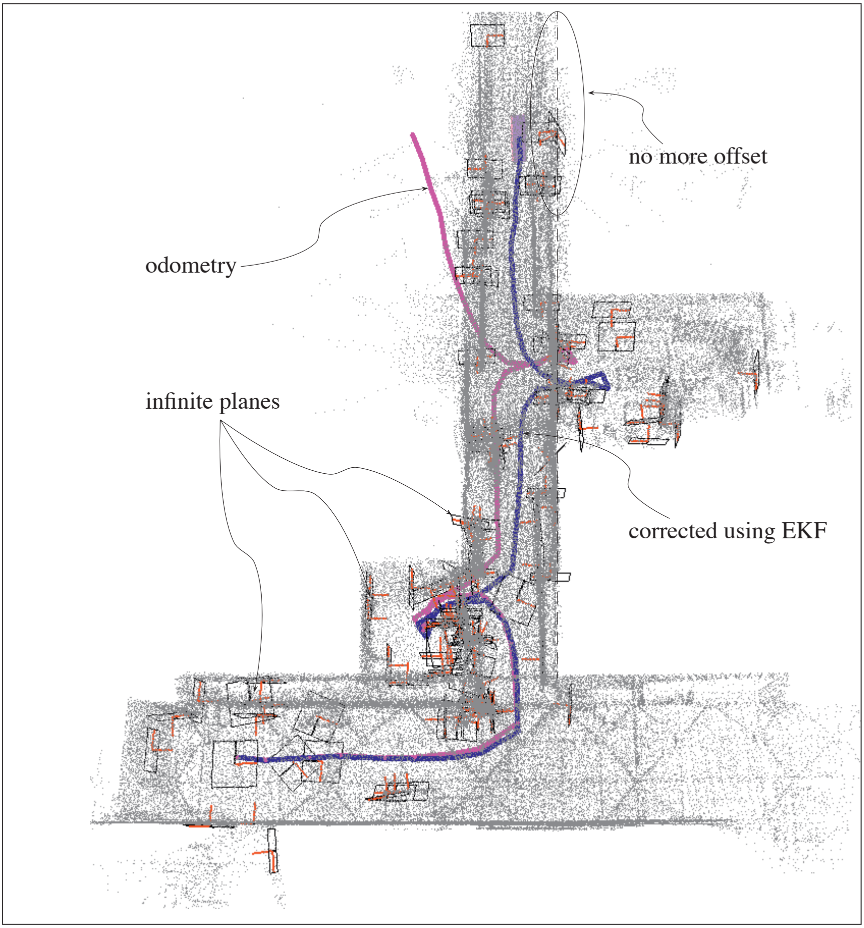


Figure 6.6: The same experiment as in figure 6.4 and 6.5, but using the EKF-based SLAM algorithm. The underlying features used are infinite planes extracted probabilistically from the raw 3D scans. The latter have been plotted in grey for better visualization at the estimated poses output by the EKF. Note that the offset of figure 6.5 has disappeared, however scans in the upper image seem to be less exactly aligned resulting in a somewhat less sharp point cloud.

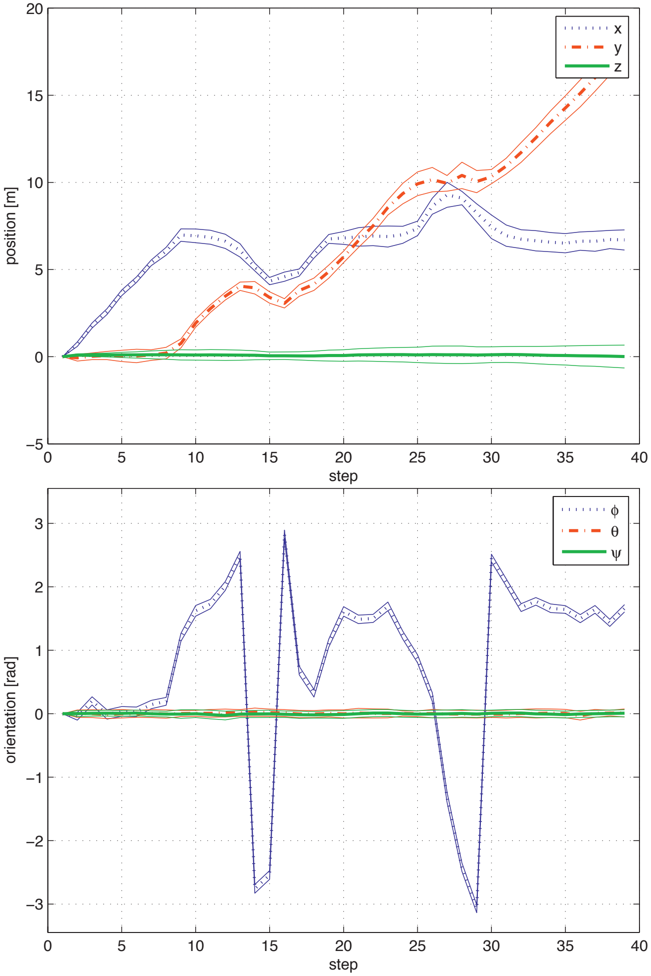


Figure 6.7: The evolution of the components of the robot pose $\mathbf{x}_{WR} = (x, y, z, \phi, \theta, \psi)^T$ with their $\pm 10\sigma$ error bounds. The error grows as new areas of the environment are explored. At the end of the experiment, the translational error $\sigma_t < 0.1$m, with $t \in \{x, y, z\}$, whilst the rotational error $\sigma_r < \frac{1}{100}$rad ($< 1^\circ$), with $r \in \{\phi, \theta, \psi\}$.

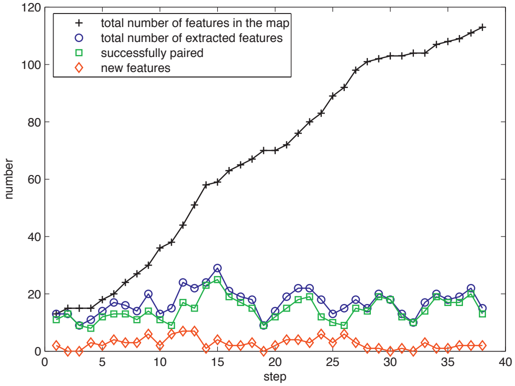


Figure 6.8: The evolution of feature numbers pairings during the first experiment in an office corridor. The black curve represents the total number of features in the map, the blue the total number of features in the current scan, the green and red are the successfully paired and new features, respectively. Note that throughout the experiment, the number of successfully paired features stays above 8 at all times, resulting in a reasonably large overlap between consecutive scans.

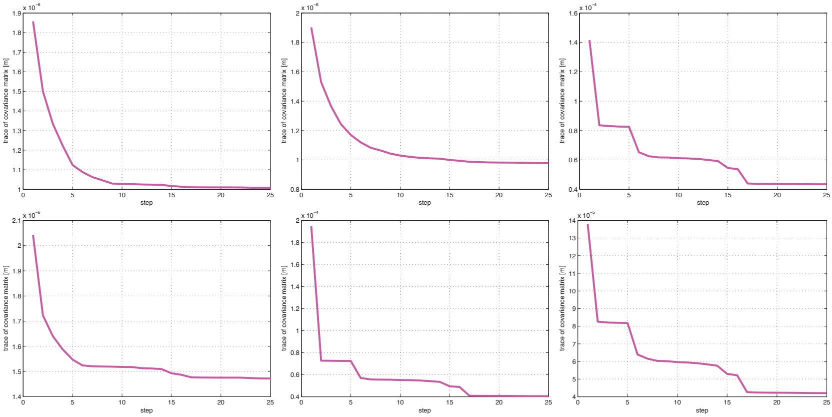


Figure 6.9: The evolution of the feature uncertainty of a subset of features over 25 steps. The curve represents the trace of the covariance matrix of the plane. It can be observed that the uncertainty decreases monotonically while new information is gathered.

a fraction of a second[2] while the number of features is relatively low ($N_F < 100$), not counting the data association procedure.

However, up to this point, it also brings along the following problems: Firstly, the data association is a critical step which can, if not working correctly, make the algorithm diverge. In order to obtain a reliable data association, the error models used have to carefully chosen. The modelled odometry error has to be set sufficiently large to enclose systematic errors perturbing robot motion caused by not precisely known wheel diameters and wheel slippage. In this work it is further set to be proportional w.r.t. the travelled distance, which is always reasonable as long as the robot velocity and scanning frequency of the exteroceptive sensor are not constant. The uncertainty of the extracted planes is directly propagated from the uncertainty of the raw data. Note that when the number of points N constituting the plane increases, the modelled error represented by a Gaussian distribution with mean $\hat{\mathbf{p}}_{\mathbf{P}}$ and covariance matrix $\mathbf{C}_P$ decreases. If the planar segment is composed of for example 10000 random data points lying in a plane with covariance

---

[2] in Matlab on a Pentium M 1.4 GHz

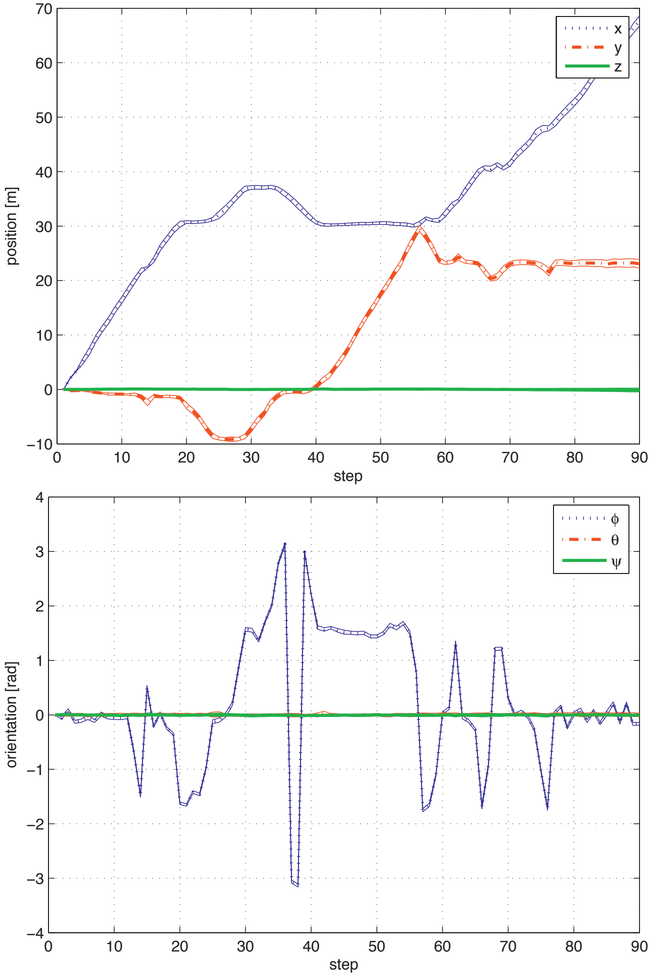


Figure 6.10: The evolution of the components of the robot pose $\mathbf{x}_{WR} = (x, y, z, \phi, \theta, \psi)^T$ with their $\pm 10\sigma$ error bounds during the experiment in the long corridor. The error grows as new areas of the environment are explored. At the end of the experiment, the translational error is $\sigma_t < 0.1$m, with $t \in \{x, y, z\}$, whilst the rotational error is $\sigma_r < \frac{1}{100}$rad ($< 1^\circ$), with $r \in \{\phi, \theta, \psi\}$.

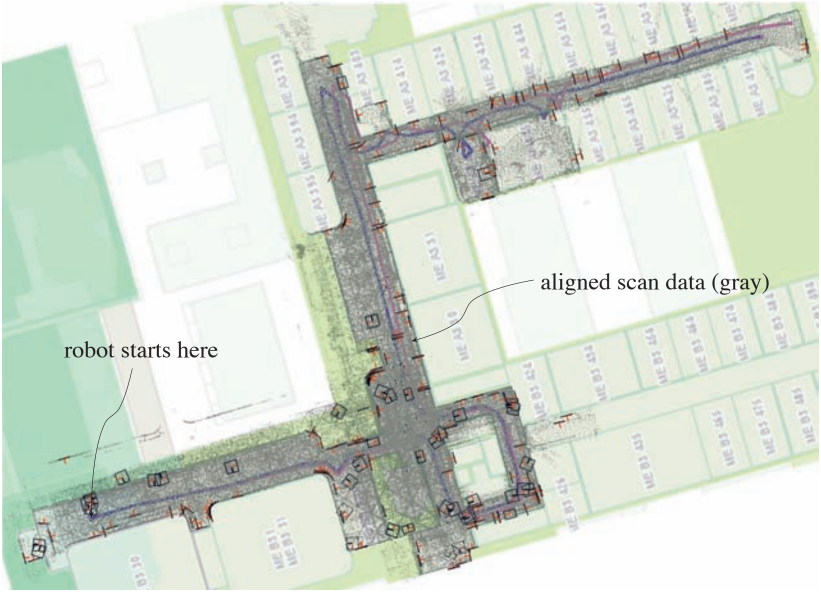


Figure 6.11: In this experiment, the complete corridor of our lab was reconstructed. The robot travelled 140m and took 90 3D scans in a stop-and-go manner. The resulting map is visualized as aligned raw data (gray) overlayed onto a building map for visual comparison. The total number of found planar features is 244, see also figure 6.12.

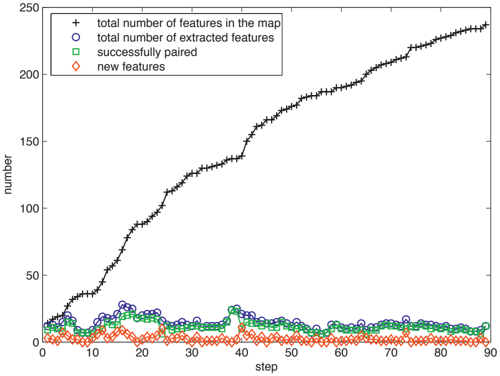


Figure 6.12: The evolution of feature numbers pairings during the long corridor experiment. The black curve represents the total number of features in the map, the blue the total number of features in the current scan, the green and red are the successfully paired and unpaired, respectively.

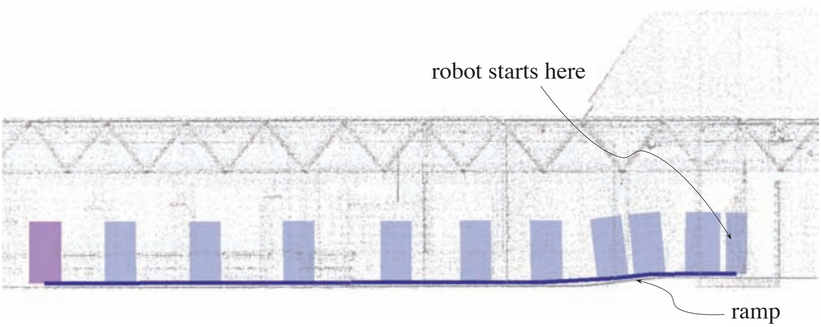


Figure 6.13: Side view of a reconstructed corridor including a small ramp on the right. The robot moves from the right to left, descending a small ramp on its way. The change in height is approximately 20cm and can be successfully reconstructed. As the wheel encoders don't provide information about changes in slope, the ICP algorithm is used to simulate a three-dimensional odometry by aligning consecutive scans.

matrix $\mathbf{C}_i = 0.01^2 \cdot \mathbf{I}_3$, the resulting covariance matrix[3] of the plane is

$$\mathbf{C}_P = \begin{bmatrix} 0.0000000689 & 0.0000000589 & 0.0000000591 \\ 0.0000000589 & 0.00000001194 & 0.0000000018 \\ 0.0000000591 & 0.0000000018 & 0.0000001203 \end{bmatrix}. \qquad (6.1)$$

In this exemplary case, the standard deviation σ_d of the orthogonal distance d is $\sigma_d = 0.0000000689^{0.5} = 0.00026249$ (see first element of first row of matrix $\mathbf{C}_P$). This corresponds to a quarter of a millimeter, which in practice is not realistic, as systematic errors inherent to the 3D sensor cannot be compensated completely. Hence, with increasing number of supporting points, the residual systematic errors exceed the decreasing statistical errors. In this work, this issue is tackled by introducing a heuristic consisting of a constant error e_c which is simply added to the diagonal elements of the plane's covariance matrix $\mathbf{C}_P$.

A further improvement is obtained by augmenting the information content and therefore the discernibility of the used features. In the loop experiment presented below, planar segments are used enhancing the infinite planes, which leads to visually more appealing maps and better data association performance.

---

[3] calculated by the method described in chapter 3

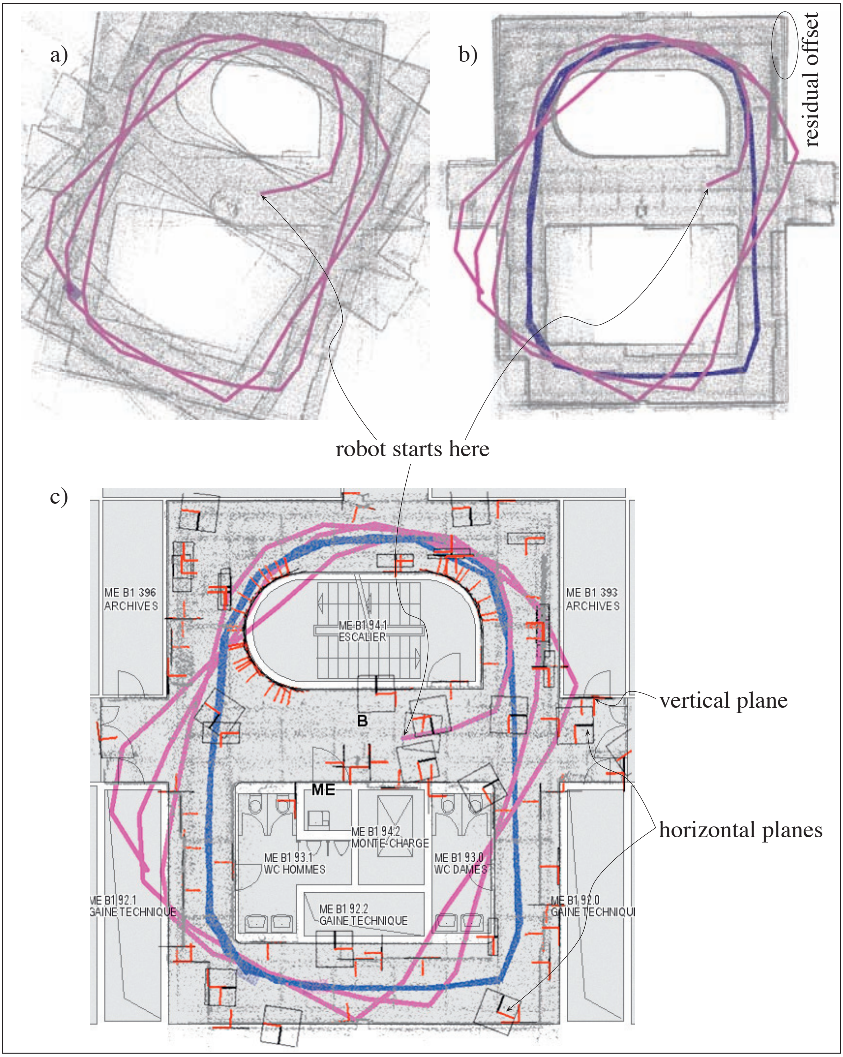


Figure 6.14: Experiment with a loop of the size of 14 × 11m. The robot starts in the center and makes almost three rounds in counterclockwise direction around the inner structures. Image a) shows aligned 3D scans using odometry only leading to an inconsistent map. Image b) shows the result using the ICP algorithm in a pairwise manner. The accumulated odometry error can be drastically reduced, but a small residual error remains (see offset). Image c) is the result using the EKF-based approach. The map built has been superimposed on a building plan for visual comparison.

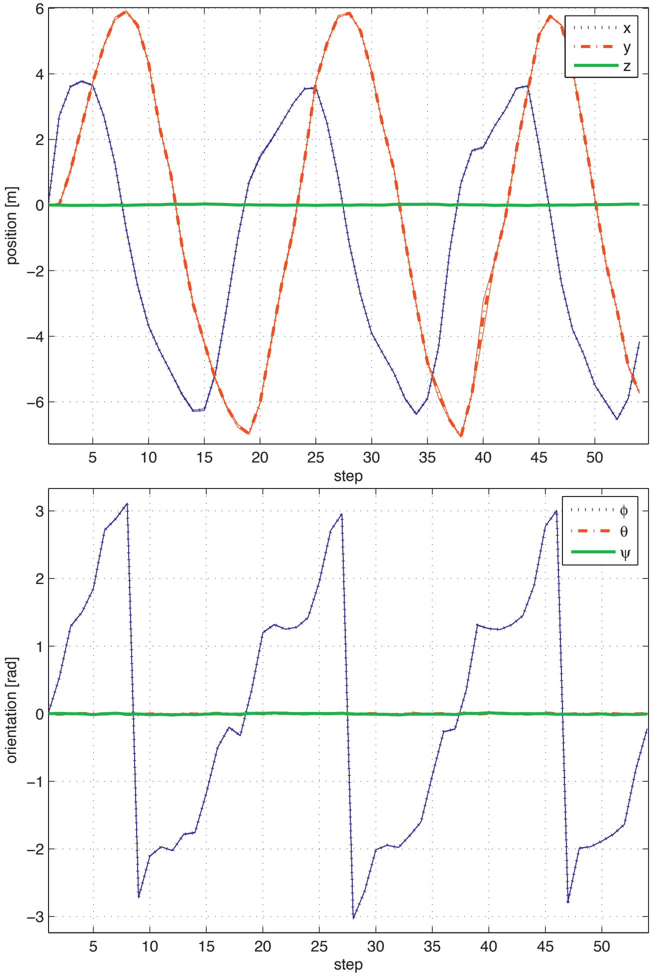


Figure 6.15: The evolution of the components of the robot pose $\mathbf{x}_{WR} = (x, y, z, \phi, \theta, \psi)^T$ with their $\pm 10\sigma$ error bounds during the experiment with the loop. The error grows as new areas of the environment are explored. See figure 6.23 for a closer analysis.

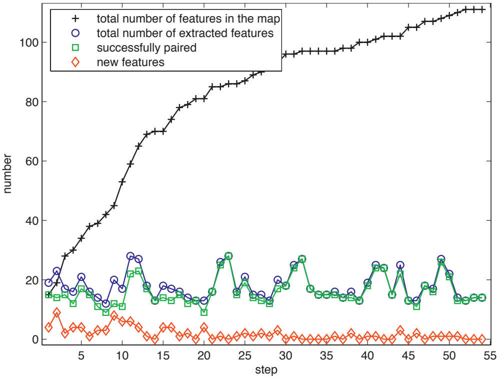


Figure 6.16: The evolution of feature numbers and pairings during the experiment with the loop. The black curve represents the total number of features in the map, the blue the total number of features in the current scan, the green and red are the successfully paired and unpaired, respectively. As from around step 15, the robot travels through known environment, the number of newly observed features (red) decreases and the total number (black) doesn't grow boundless in contrast to figure 6.8

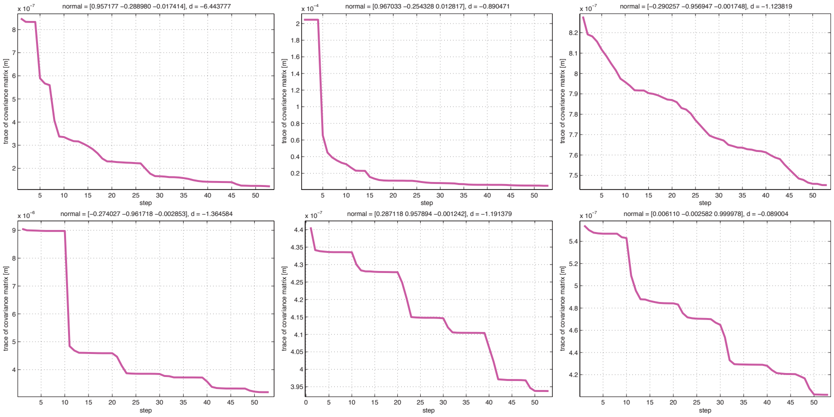


Figure 6.17: The evolution of the feature uncertainty of a subset of features over 54 steps in the loop. The curves represents the trace of the covariance matrix of the plane. Note that for example the step-like shape of the central graph in the lower row can be explained by the location of the feature between the inner structures (see figure 6.14). The robot reobserves the feature twice per round and therefore the information is added piecewise, leading to the step-like shape.

6.4 Using Planar Segments

6.4.1 Introduction

A map based on infinite planar features (see figure 6.19 left) is almost impossible to understand for a human. A robot however can perform SLAM using such a representation and benefit from its compactness, requiring a very limited amount of memory and computation power. The shown exemplary map requires less than 3 KB of memory. In real world applications, this could be appropriate for small-scale environments and robots with limited onboard computing and memory resources. The main problem however when using infinite planes or other infinite features is that they can be confused during data association. Two features with similar model parameters could be successfully paired even though they are far-off from each other in the physical reality and hence not necessarily corresponding.

Three-dimensional point clouds generated by aligning consecutive 3D scans are an interesting alternative (see figure 6.19 right) to feature-based representations. They are easy to decode, show high detail, but on the other hand require a fairly large amount of memory, are more time-consuming to build and don't provide higher-level information about the environment.

In the following, a representation is described lying in-between these two extremes. On one side it is memory-efficient requiring the least amount of memory necessary and on the other hand it provides dense information easily readable by humans and computers. It is based on the planar segments presented in chapter 3.

6.4.2 Incremental map building using planar segments

In order to build a map based on planar segments in an incremental way, not only the infinite plane parameters of corresponding features have to be estimated but also the segment information has to be updated. As the segment information is defined in two dimensions as a set of polygons w.r.t. the local reference frame of the plane, a mechanism has been developed which allows to fuse segment information of paired features together. This is carried out by firstly calculating the new plane parameters as with infinite planes but requiring that corresponding features overlap, secondly projecting the segment information of both features onto the updated infinite plane. In a third step, this projected segment information is projected back into the local coordinate frame of the plane and finally fused. Figure 6.18 shows how the planar segment representing the floor of the loop environment evolves during the SLAM experiment. Similar information is available for all features of the map and can be used to create statistics about the reconstructed environment, extract higher-level features like for example doors, etc.

Figure 6.20 shows the result of reconstructing the above-mentioned loop environment as a set of planar segments. The created map not only shows high detail which is further underlined by figure 6.21, but is also small in size compared to the number of input data

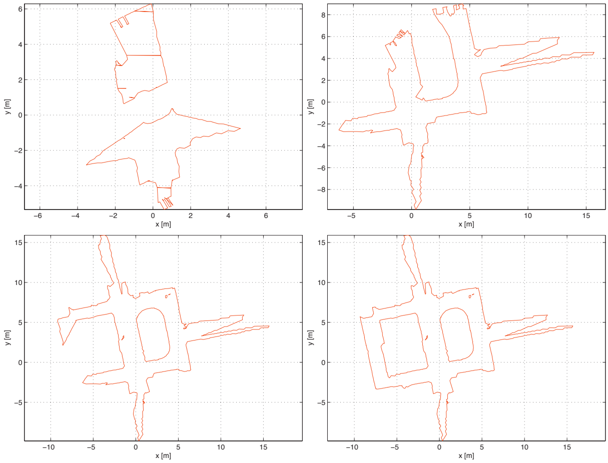


Figure 6.18: The evolution of the planar segment representing the floor of the environment used for the experiment with the loop. The top left graph is captured at step 1, the top right at step 7, the bottom left at step 14 and the bottom right at step 20. Note that the incrementally grown planar segment provides useful information for path-planning, exploration, building statistics etc.

points. After 20 steps, the robot took 20 scans of 57920 data points each, which adds up to over 1 million raw data points. The reconstructed map is composed of 89 features represented by 167 polygons with a total of 17548 underlying data points. This represents less than 2% of the original amount of data. Figure 6.22 is another example demonstrating the strong compression ratio of the representation developed. It shows the long corridor experiment composed of ninety 3D scans which corresponds to an input total number of points of $90 \times 57920 = 5212800$. The reconstructed map is composed of 244 planar segments, represented as 299 polygons with 44696 supporting data points. This represents less than 1% of the amount of data input.

6.5 Discussion

Using the segment information is favorable for the SLAM algorithm, as it not only provides a clearer visualization but also improves the data association. By requiring that the planar segments to be matched have to overlap, the number of false positive matches can be reduced. This overlap is evaluated by polygon clipping operations, which are common in the field of computer graphics and not further detailed here. It should be underlined, that the current implementation is rather a proof of concept than highly optimized code. Especially the search for corresponding features is momentarily implemented in a naive

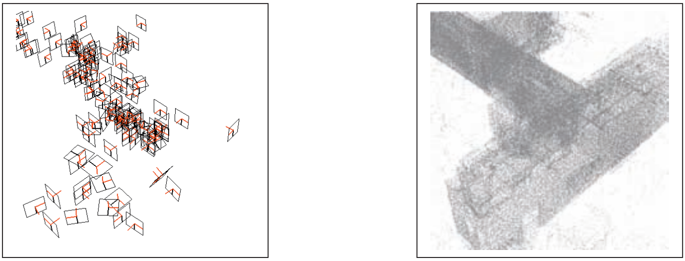


Figure 6.19: The left image shows a map based on infinite planes depicted as coordinate frames and rectangles. The position of each plane in space is defined by the center of gravity of the underlying point cloud, leading to a very compact map using less than 3 KB. Even though this map is almost unreadable for a human being, the robot can use it to perform SLAM in 3D space as shown below. The right image shows the same map as a three-dimensional point cloud ($\sim$ 100000 points) composed of 40 aligned 3D scans. Note that the original scans containing over 8 million data points have been subsampled for faster visualization. It is much easier to read, but on the other hand needs more memory space and cannot be fed into the EKF algorithm directly.

way, as all features are compared among themselves. Hence, besides the covariance matrix of the stochastic map, which scales quadratically with the number of features, another procedure scaling quadratically is included. The latter can be improved by using more elaborate nearest-neighbor search methods similar to KD-trees [Samet, 1990], for example.

6.6 Problematic Issues

As already mentioned above, problematic issues are firstly the unfavorable scaling of the algorithm performance with increasing number of features. The covariance matrix of the stochastic map is square and has to be inverted in every Kalman Filter cycle, which becomes a bottleneck with an increasing number of features. Possibilities to overcome this issue is to use specialized optimized versions of the Kalman Filter like the *compressed* EKF (CEKF) [Guivant and Nebot, 2001b], which leads to a cost proportional to $O(N_a^2)$, where N_a is the number of landmarks in a local environment.

A second issue is that the Jacobians of the composition of two 3D locations contain singularities like for example due to the first term in the second column of matrix $\mathbf{K}_1$ of eq. (A.18). If $\cos_{\theta_3}$ approaches $\pm\frac{\pi}{2}$, the accuracy of the associated covariance decreases drastically [Smith et al., 1990]. In the experimental setup used for this work, this cannot

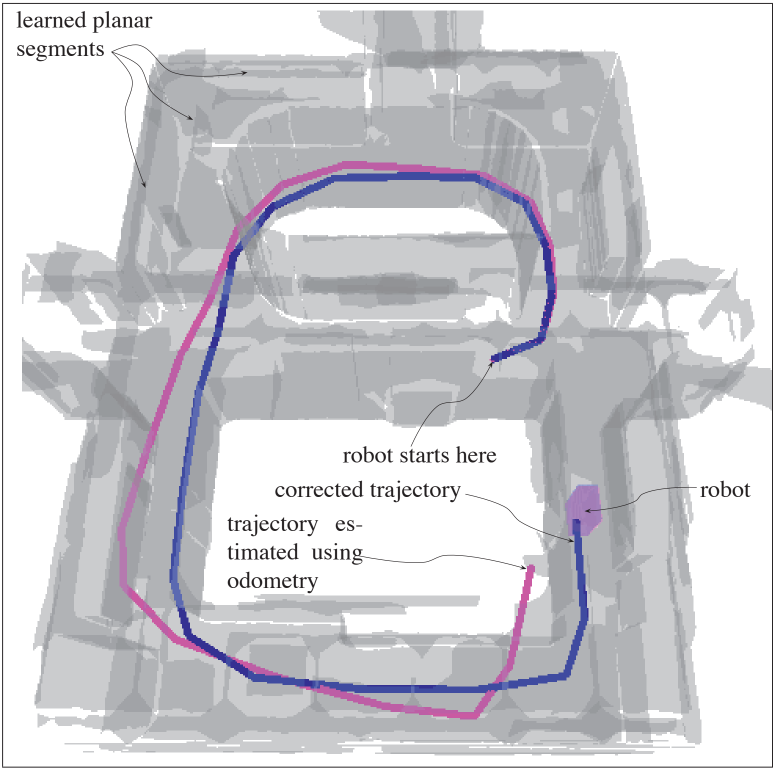


Figure 6.20: Resulting map of the same experiment as in figure 6.14 using planar segments visualized as transparent α-shapes. Note that the robot performed only 20 steps. The trajectory estimated using odometry information only is depicted in magenta, the corrected trajectory in blue. The number of learned planar segments is 89, with a total number of supporting points of 17548 visualized as 167 polygons. This represents less than 2% of the raw 1158400 (20×57920) data points.

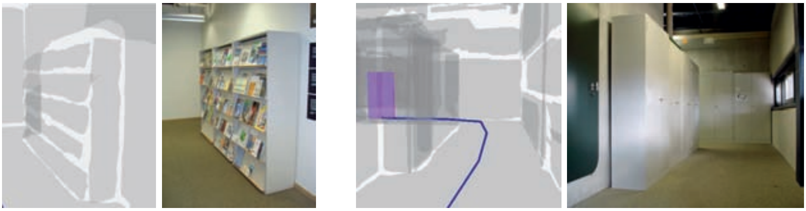


Figure 6.21: Close-ups of some reconstructed areas of the map shown in figure 6.20 with photos for comparison. The left view represents a rack with magazines, the right scene shows a part of a corridor and a the robot displayed by a pink cuboidal having moved around a corner, as well as its corrected trajectory (blue).

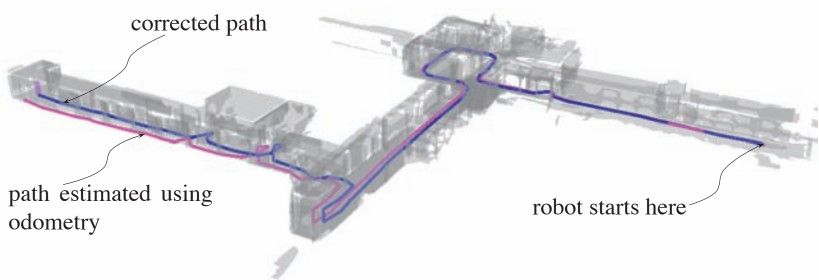


Figure 6.22: Perspective view of the long corridor experiment reconstructed using planar segments. The robot moves from the right to the left making 90 3D scans in a stop-and-go manner. The number of planar segments found is 244 composed of a total of 44696 data points / 299 polygons. This corresponds to a compression ration of less than 1% with respect to the raw data gathered by the 3D sensor ($90 \times 57920 = 5212800$). Note that the path estimated using the wheel encoder data in this case lies surprisingly close to the corrected path.

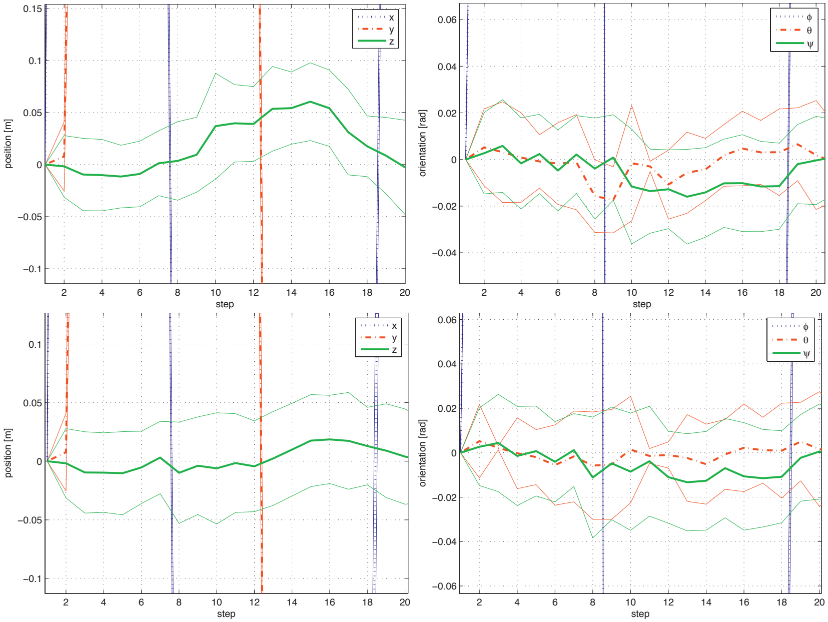


Figure 6.23: The top row shows the evolution of the estimated robot pose and its associated error during the first 20 steps of the experiment with the loop environment. It represents a magnification of figure 6.15. Note that the z-component varies substantially, taking values between -1 and $+6$ cm. This doesn't reflect well the reality as the environment is flat. It is believed that this error arises from false pairings, which in turn are caused by the limited discernibility of infinite planar features. The bottom row shows the evolution of the robot pose, this time considering segment information for data association. Pairings are only accepted if their planar segments overlap. It can be observed, that the z-component of the estimated robot pose stays much closer to zero.

happen, as θ_3 stays close to zero, because the robot rolls on the floor. However, there are several other singularities in the above-mentioned Jacobians and it is not always clear whether the critical values are reached or not. A way to avoid this issue is to replace the standard EKF by the so-called *Unscented Kalman Filter* (UKF), which doesn't require the sometimes complex calculation of Jacobians. Instead, it uses a specific set of samples approximating the underlying Gaussian distributions. See for example [Julier and Uhlmann, 1997], [Van der Merwe et al., 2000] for an introduction or [Kraft, 2003] for a concrete implementation.

A third issue is the data association or correspondence problem. In case of a simple environment as used for the presented simulation, infinite planes prove adequately discernable. Hence, it is simple to find working parameters for the SLAM algorithm, which are the χ^2-threshold for plane comparison and the modelled uncertainty of the odometry. In real-world situations however, environments can be composed of several hundred features which have to be correctly discerned. A way to improve data association performance is to increase the complexity of the involved features as done in [Lamon et al., 2003]. Herein, this is achieved by using segment information, which is considered for data association by requiring an overlap of the pairing candidates.

6.7 Summary

This chapter presented the experimental validation of the 3D SLAM algorithm presented above. At first, a simulation was used to validate the three-dimensional version of the algorithm. It showed that the algorithm is able to reconstruct a simple environment and the 6-DOF[4] trajectory of the robot, while its error w.r.t. ground-truth stays bounded. The next set of experiments was carried out in a mainly flat office environment, which the algorithm proved capable to reconstruct in a consistent manner. With the aid of the ICP algorithm it was even able to correctly estimate slightly inclined areas of the environment exemplified by a ramp.

The algorithm shows the same characteristic drawbacks as its two-dimensional counterpart which are its quadratically scaling complexity with an increasing number of features and its sensitivity to a working data association. Furthermore, it is sensitive to linearization errors, sometimes leading to inconsistencies and to systematic odometry errors, which have to be considered by a sufficiently large assumed motion error. This can be addressed by increasing the scanning frequency of the external sensor or equivalently reducing the displacement between two scanning poses. It was finally shown how the data association can be improved by using segment information which also leads to more useful and appealing maps. The generated maps are shown to be detailed, as even bookshelves are reconstructed (see figure 6.21), and very compact at the same time, requiring

---

[4] degrees of freedom

around 1% of the amount of memory of the input raw data.

Chapter 7

Conclusions and Outlook

The goal of this thesis was to develop a complete feature-based 3D SLAM approach for a mobile robot and validate it experimentally. This firstly required to find 3D sensors suitable for a mobile robot. Two have been taken into consideration, the CSEM Swiss Ranger, a time-of-flight camera generating dense range images in real-time, and a custom-built 3D scanner based on a commercially available 2D laser scanner. Both sensors are active, rely on the time-of-flight principle and are capable of delivering dense 3D data composed of many thousand data points.

Chapter 2 presented the modelling of the two sensors which requires their calibration. This includes compensating the systematic errors and analyzing the residual statistical errors, which are modelled by Gaussian distributions. The rotating laser scanner is shown to lead to an accuracy within centimeter range whereas the Swiss Ranger shows persisting systematic errors even after a simple calibration using a flat wall at different distances. This is believed to be caused by several error sources like the weak optics leading to distortions, the inhomogeneous illumination provided by the array of LEDs, the strong temperature dependency, etc. Further taking into account its restricted field of view of around $45°$, which is unfavorable for SLAM, it is concluded that the Swiss Ranger is less suited for precise localization and mapping than the rotating laser scanner. Even though the latter's scanning frequency is far below the former's, its wide angle of view (up to $180° \times 330°$) and its precision of around a centimeter are the decisive factor. The Swiss Ranger is therefore not further considered subsequently.

In the following chapter, the representation based on planar segments was presented. Each of these is composed of an infinite plane represented with its associated uncertainty covariance matrix and a set of supporting polygons extracted from the supporting (segmented) raw data points. It is shown how the plane uncertainty can directly be calculated from the raw data uncertainty which is modelled with the aid of the findings of the last chapter. The first moments of the plane parameters are found by weighted principal component analysis (PCA) and the second moments using common error propagation techniques. Finally, the planes are described in compliance with the SPmodel (Symmetries and Perturbations Model), which is a framework allowing to represent and process uncertain geometric data in a consistent way. The SPmodel was successfully used for feature-based SLAM in 2D and also forms the basis for the three-dimensional SLAM algorithm presented in this work.

Chapter 4 showed how planes are segmented from raw point clouds. Two planar segmentation algorithms were presented in detail, one is called grid-based segmentation (GBS) algorithm and the other region-growing segmentation (RGS) algorithm. The former algorithm (GBS) aims at extracting the most important planes in a flexible and robust way by decomposing the space into regular cells, finding a single plane for every cell and fusing similar neighboring planes together. This leads to an efficient algorithm capable of reconstructing the most important planes of the scene without requiring a regular structure in the raw data. The second algorithm developed (RGS) is based on the region-growing paradigm. It first calculates the normal at every data point by exploiting the inherent regular structure of the 3D scan. Then it incrementally grows regions by starting at the flattest point of the scan which is found by evaluating the residual least-square plane fitting error. With several additional constraints, it is shown that it leads to highly precise segmentation results. As both algorithms tend to output an over-segmentation, a post-processing step was implemented to merge corresponding planar regions. It is further shown how both algorithms can be adapted to process uncertainty information which can improve segmentation quality but has to be paid with a heavier computational burden.

Finally, the 3D SLAM algorithm was presented in chapter 5. It is based on the Extended Kalman Filter (EKF) SLAM approach used in conjunction with the SPmodel. At first, an experimental validation was carried out using a simulation which provides means to compare the estimated trajectory and map to ground-truth. It showed that the algorithm is capable of estimating a trajectory in full (simulated) 3D space with six degrees of freedom and its surrounding environment in a precise way. A number of real experiments was carried out with the BIBA0 robot, a differential-drive mobile robot of our lab equipped with the rotating laser scanner described above. It showed that the system was able to reconstruct all test environments in an accurate way leading to compact and detailed maps, composed of planar segments. However, the approach also showed the same characteristic drawbacks as its two-dimensional counterpart. These are its unfavorable scaling performance with increasing number of features, its sensitivity to a working data association and to the underlying motion error model, as well as its sometimes inconsistent results caused by linearization errors. Its advantages on the other hand are its efficiency with a limited number of features, its accuracy and the resulting output map, which due to the segment information is compact in size and detailed at the same time and therefore interesting for higher-level tasks like scene understanding, manipulation or simply visualization.

Recapitulating, this thesis presents the following main contributions: Firstly, a way of modelling uncertain 3D data and fitting a plane to it in a probabilistic way. This allows to keep the modelled uncertainty as realistic as possible without the need to make additional assumptions. Secondly, two planar segmentation algorithms were developed, both capable of processing uncertain data and robust with respect to noise and clutter typically appearing in robotic environment scans. Thirdly, the experimental validation of the three-dimensional EKF-SLAM algorithm is new, which is concluded to be applicable in environments of limited size. Finally, the extension to using planar segments is presented, which not only improves the SLAM algorithm but also leads to visually appealing, very compact and useful maps.

In future work, issues arising from singularities in the Jacobians of the transformations of three-dimensional location vectors have to be further investigated, which can lead to unexpected algorithm behavior. Alternative representations for rotations like quaternions should be evaluated and compared to the representation based on location vectors and homogenous matrices used in this work. Another way to tackle this issue could be to use an unscented Kalman Filter (UKF) instead of the Extended Kalman Filter (EKF). It avoids the use of Jacobians completely and generally leads to superior estimation performance.

A second issue to be addressed is the complexity of the EKF-SLAM algorithm, which in the current implementation scales quadratically with the number of map features. In a larger environment composed of several hundred features, this leads to unacceptably low performance. Hence, techniques like local map sequencing could be used, decomposing the global map into smaller local maps which are fused together when necessary. Or a combination of particle filters with the used representation based on planar segments could be considered, in a similar way as it is done with points in two dimensions already (FastSLAM). It would be interesting to find out if this is possible and how many particles are necessary to efficiently create consistent maps in 3D.

Finally, another track to follow would be to improve the discernibility of the planar features used with the goal to simplify data association. If, for example, intensity information was included, this could be exploited for data association and at the same time lead to more appealing and useful maps.

Appendix A

Some Mathematical Background

A.1 Error Propagation

Let $\mathbf{f} = \mathbf{f}(\mathbf{x})$ be m functions $f_i = f_i(x_1, \ldots, x_n), \;\; (i = 1 \ldots m)$ of n variables. Now the question is how the expectation and the variance of $\mathbf{f}(\mathbf{x})$ look like supposing $\mathbf{x}$ is a random variable. Applying Taylor leads to

$$f_i(\mathbf{x}) = f_i(\langle \mathbf{x} \rangle) + \sum_{k=1}^{n} (x_k - \langle x_k \rangle) \left. \frac{\partial f_i}{\partial x_k} \right|_{\mathbf{x}=\langle \mathbf{x} \rangle} + O(x_k^2). \tag{A.1}$$

Assuming errors are small, terms of second order and higher are generally omitted. The (approximate) expectation of every component of $\mathbf{f}$ can then be calculated yielding

$$\langle f_i(\mathbf{x}) \rangle \approx f_i(\langle \mathbf{x} \rangle). \tag{A.2}$$

To calculate the covariance matrix $\mathbf{C}[\mathbf{f}(\mathbf{x})]$, the matrix notation of (A.1) is used:

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\langle \mathbf{x} \rangle) + \mathbf{F}(\mathbf{x} - \langle \mathbf{x} \rangle), \tag{A.3a}$$

where

$$\mathbf{F} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{\mathbf{x}=\langle \mathbf{x} \rangle} \tag{A.3b}$$

is the *Jacobian* of $\mathbf{f}$ at $\mathbf{x} = \langle \mathbf{x} \rangle$. Therewith, the following equations hold for the covariance matrix:

$$\begin{aligned}
\mathbf{C}[\mathbf{x}] &= (\mathbf{x} - \langle \mathbf{x} \rangle)(\mathbf{x} - \langle \mathbf{x} \rangle)^T \\
\mathbf{C}[\mathbf{f}(\mathbf{x})] &\approx (\mathbf{f}(\mathbf{x}) - \langle \mathbf{f}(\mathbf{x}) \rangle)(\mathbf{f}(\mathbf{x}) - \langle \mathbf{f}(\mathbf{x}) \rangle)^T \\
&= (\mathbf{f}(\langle \mathbf{x} \rangle) + \mathbf{F}(\mathbf{x} - \langle \mathbf{x} \rangle) - \langle \mathbf{f}(\mathbf{x}) \rangle)(\mathbf{f}(\langle \mathbf{x} \rangle) + \mathbf{F}(\mathbf{x} - \langle \mathbf{x} \rangle) - \langle \mathbf{f}(\mathbf{x}) \rangle)^T \qquad \text{(A.4)} \\
&= (\mathbf{F}(\mathbf{x} - \langle \mathbf{x} \rangle))(\mathbf{F}(\mathbf{x} - \langle \mathbf{x} \rangle))^T \\
&= \mathbf{F}\mathbf{C}[\mathbf{x}]\mathbf{F}^T
\end{aligned}$$

Note that the Jacobian $\mathbf{F}$ represents a linearization of $\mathbf{f}$ at $\mathbf{x} = \langle \mathbf{x} \rangle$. With non-linear $\mathbf{f}$, this always represents and approximation. Also note that (A.4) can be rewritten as the Gaussian law of error propagation:

$$\begin{aligned}
cov(f_i, f_j) &= \sum_{k=1}^{n} \sum_{l=1}^{n} \frac{\partial f_i}{\partial x_k} \frac{\partial f_j}{\partial x_l} cov(x_k, x_l), \quad (i \neq l) \\
\sigma_{f_i}^2 &= \sum_{k=1}^{n} \left(\frac{\partial f_i}{\partial x_k} \sigma_{x_k} \right)^2 \qquad\qquad\quad (i = l)
\end{aligned} \qquad \text{(A.5)}$$

The following section describes how 3D points with associated uncertainty can be transformed using the error propagation method described above.

A.2 Transforming 3D points with Uncertainty

This appendix describes how a point $(x_p, y_p, z_p)^T \in \mathbb{R}^3$ represented in Cartesian three-dimensional space with uncertainty covariance matrix $\mathbf{C}_p$ can be transformed using a rigid-body motion written as location vector $\mathbf{x} = (x, y, z, \phi, \theta, \psi)^T$. This equals a change of coordinate systems.

First of all, this location vector $\mathbf{x}$ has to be converted into a homogeneous matrix $\mathbf{H}$, yielding[1]:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \qquad \text{(A.6a)}$$

with

$$\mathbf{R} = \begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \qquad \text{(A.6b)}$$

---

[1] c_ϕ stands for $\cos(\phi)$, s_θ for $\sin(\theta)$ etc.

and

$$\mathbf{T} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.6c}$$

The transformed point $(x_t, y_t, z_t)^T$ can then be found by

$$\begin{aligned} \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} &= \mathbf{R} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} + \mathbf{T} \\ &= \underbrace{\begin{bmatrix} c_\phi c_\theta x_p + (c_\phi s_\theta s_\psi - s_\phi c_\psi) y_p + (c_\phi s_\theta c_\psi + s_\phi s_\psi) z_p + x \\ s_\phi c_\theta x_p + (s_\phi s_\theta s_\psi + c_\phi c_\psi) y_p + (s_\phi s_\theta c_\psi - c_\phi s_\psi) z_p + y \\ -s_\theta x_p + (c_\theta s_\psi) y_p + (c_\theta c_\psi) z_p + z \end{bmatrix}}_{\mathbf{f}=(f_1, f_2, f_3)^T} \end{aligned} \tag{A.7}$$

In order to propagate the uncertainty of input point $(x_p, y_p, z_p)^T$ which is encoded in $\mathbf{C}_p \in \mathbb{R}^{3\times3}$, the Jacobian of $\mathbf{f}$ has to be calculated:

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial(x_p, y_p, z_p, x, y, z, \phi, \theta, \psi)} \tag{A.8a}$$

$$= \begin{bmatrix} \frac{\partial f_1}{\partial x_p} & \frac{\partial f_1}{\partial y_p} & \frac{\partial f_1}{\partial z_p} & \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial \phi} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \psi} \\ \frac{\partial f_2}{\partial x_p} & \frac{\partial f_2}{\partial y_p} & \frac{\partial f_2}{\partial z_p} & \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} & \frac{\partial f_2}{\partial \phi} & \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial \psi} \\ \frac{\partial f_3}{\partial x_p} & \frac{\partial f_3}{\partial y_p} & \frac{\partial f_3}{\partial z_p} & \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} & \frac{\partial f_3}{\partial \phi} & \frac{\partial f_3}{\partial \theta} & \frac{\partial f_3}{\partial \psi} \end{bmatrix} \tag{A.8b}$$

$$=: \begin{bmatrix} \mathbf{U} & \mathbf{V} & \mathbf{W} \end{bmatrix} \tag{A.8c}$$

with

$$\mathbf{U} = \begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}, \tag{A.8d}$$

$$\mathbf{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.8e}$$

$$\mathbf{W}_{(1:3,1)} = \begin{bmatrix} -x_p s_\phi c_\theta - y_p(s_\phi s_\theta s_\psi + c_\phi c_\psi) - z_p(s_\phi s_\theta c_\psi - c_\phi s_\psi) \\ x_p c_\phi c_\theta + y_p(c_\phi s_\theta s_\psi - s_\phi c_\psi) + z_p(c_\phi s_\theta c_\psi + c_\phi s_\psi) \\ 0 \end{bmatrix}, \tag{A.8f}$$

$$\mathbf{W}_{(1:3,2)} = \begin{bmatrix} -x_p c_\phi s_\theta + y_p c_\phi c_\theta s_\psi + z_p c_\phi c_\theta c_\psi \\ -x_p s_\phi s_\theta + y_p s_\phi c_\theta s_\psi + z_p s_\phi c_\theta c_\psi \\ -x_p c_\theta - y_p s_\theta s_\psi - z_p s_\theta c_\psi \end{bmatrix}, \tag{A.8g}$$

$$\mathbf{W}_{(1:3,3)} = \begin{bmatrix} y_p(c_\phi s_\theta c_\psi + s_\phi s_\psi) - z_p(c_\phi s_\theta s_\psi - s_\phi c_\psi) \\ y_p(s_\phi s_\theta c_\psi - c_\phi s_\psi) - z_p(s_\phi s_\theta s_\psi + c_\phi c_\psi) \\ y_p c_\theta c_\psi - z_p c_\theta s_\psi \end{bmatrix}. \tag{A.8h}$$

Using the findings of the latter section, the covariance matrix $\mathbf{C}_t$ of the transformed point $(x_t, y_t, z_t)^T$ can be found by

$$\mathbf{C}_t = \mathbf{J}\mathbf{C}_\mathbf{p}\mathbf{J}^T. \tag{A.9}$$

A.3 The Symmetries and Perturbations Model (SPmodel)

A.3.1 Introduction to Location Vectors and Homogeneous Matrices

This work specifically addresses the problem of SLAM in 3D space. In three dimensions, the robot's pose $\mathbf{x}_{WR}$ is described by six components $\mathbf{x}_{WR}^{3D} = (x, y, z, \phi, \theta, \psi)^T$, where the three rotation angles ϕ, θ and ψ are represented as yaw-pitch-roll angles, in comparison to three components $\mathbf{x}_{WR}^{2D} = (x, y, \theta)^T$ with only one angle θ in two dimensions. In the following, a *location vector* $\mathbf{x}_{WR}$ describes the location of a reference frame R w.r.t. the world reference frame W.

Especially due to the two added rotational degrees of freedom, the related equations for transforming such location vectors between different coordinate frames get quite complex. Hence, by introducing homogeneous matrices, this can be carried out in a convenient way requiring a single 4×4 matrix multiplication for any rigid body motion[2] in 3D. The

---

[2] a rigid body motion is equivalent to a transform in 3D space composed of a rotation and a translation.

conversion from a location vector $\mathbf{x}_{WR}$ to a homogenous matrix $\mathbf{H}_{\mathbf{W}}^{\mathbf{R}}$ is given by

$$\mathbf{H}_{\mathbf{W}}^{\mathbf{R}} = \mathbf{Hom}(\mathbf{x}_{WR}) = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & t_x \\ n_y & o_y & a_y & t_y \\ n_z & o_z & a_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.10a}$$

with

$$\mathbf{R} = \begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi - s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \tag{A.10b}$$

and

$$\mathbf{t} = \begin{bmatrix} x & y & z \end{bmatrix}^T. \tag{A.10c}$$

The back conversion from a homogeneous matrix $\mathbf{H}_W^R$ to a location vector $\mathbf{x}_{WR}$ is given by:

$$\mathbf{x}_{WR} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T = \mathbf{Loc}(\mathbf{H}_W^R) \tag{A.11a}$$

with

$$x = t_x, \tag{A.11b}$$
$$y = t_y, \tag{A.11c}$$
$$z = t_z, \tag{A.11d}$$
$$\phi = \arctan 2(n_y, n_x), \tag{A.11e}$$
$$\theta = \arctan 2(-n_z, \cos(\phi) n_x + \sin(\phi) n_y), \tag{A.11f}$$
$$\psi = \arctan 2(\sin(\phi) a_x - \cos(\phi) a_y, -\sin(\phi) o_x + \cos(\phi) o_y. \tag{A.11g}$$

A.3.2 Operations in 3D

Mainly two operations are needed to transform location vectors in 3D space: The compounding operation denoted by $\oplus$ and the inversion denoted by $\ominus$. Explicitly, the compounding operation is given by:

$$\begin{aligned} \mathbf{x}_{WB} &= \mathbf{x}_{WA} \oplus \mathbf{x}_{AB} \\ &= Loc(Hom(\mathbf{x}_{WA}) Hom(\mathbf{x}_{AB})) \end{aligned} \tag{A.12}$$

The inversion is given by:

$$\begin{aligned}\mathbf{x}_{AW} &= \ominus\mathbf{x}_{WA} \\ &= Loc(Hom(\mathbf{x}_{WA})^{-1})\end{aligned} \tag{A.13}$$

When for example a robot moves from pose $\mathbf{x}_{WR_i}$ to the pose $\mathbf{x}_{WR_{i+1}}$ and the transition is defined by an odometry reading $\mathbf{x}_{R_iR_{i+1}}$, the new pose of the robot w.r.t. the world reference W is given by $\mathbf{x}_{WR_{i+1}} = \mathbf{x}_{WR_i} \oplus \mathbf{x}_{R_iR_{i+1}}$.

A.3.3 Jacobians of the Composition

The Jacobians of the composition $\mathbf{x}_3 = (x_3, y_3, z_3, \phi_3, \theta_3, \psi_3)^T = \mathbf{x}_1 \oplus \mathbf{x}_2$ are given by

$$\mathbf{J}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} = \left.\frac{\partial(\mathbf{y} \oplus \mathbf{z})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}_1,\mathbf{z}=\mathbf{x}_2}, \tag{A.14}$$

$$\mathbf{J}_{2\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} = \left.\frac{\partial(\mathbf{y} \oplus \mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{y}=\mathbf{x}_1,\mathbf{z}=\mathbf{x}_2}, \tag{A.15}$$

with $\mathbf{J}_{\oplus} = \{\mathbf{J}_{1\oplus}; \mathbf{J}_{2\oplus}\}$. Their values can be calculated by

$$\mathbf{J}_{1\oplus} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{M} \\ \mathbf{0} & \mathbf{K}_1 \end{bmatrix}, \qquad \mathbf{J}_{2\oplus} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix} \tag{A.16}$$

with

$$\mathbf{M} = \begin{bmatrix} -(y_3 - y_1) & (z_3 - z_1)c_{\phi_1} & a_{x_1}y_2 - o_{x_1}z_2 \\ x_3 - x_1 & (z_3 - z_1)s_{\phi_1} & a_{y_1}y_2 - o_{y_1}z_2 \\ 0 & -x_2c_{\theta_1} - y_2s_{\theta_1}s_{\psi_1} - z_2s_{\theta_1}c_{\psi_1} & a_{z_1}y_2 - o_{z_1}z_2 \end{bmatrix}, \tag{A.17}$$

$$\mathbf{K}_1 = \begin{bmatrix} 1 & [s_{\theta_3}s_{(\phi_3-\phi_1)}]/c_{\theta_3} & [o_{x_2}s_{\psi_3} + a_{x_2}c_{\psi_3}]/c_{\theta_3} \\ 0 & c_{(\phi_3-\phi_1)} & -c_{\theta_1}s_{(\phi_3-\phi_1)} \\ 0 & [s_{(\phi_3-\phi_1)}]/c_{\theta_3} & [c_{\theta_1}c_{(\phi_3-\phi_1)}]/c_{\theta_3} \end{bmatrix}, \tag{A.18}$$

$$\mathbf{R}_1 = \begin{bmatrix} c_{\phi_1}c_{\theta_1} & c_{\phi_1}s_{\theta_1}s_{\psi_1} - s_{\phi_1}c_{\psi_1} & c_{\phi_1}s_{\theta_1}c_{\psi_1} + s_{\phi_1}s_{\psi_1} \\ s_{\phi_1}c_{\theta_1} & s_{\phi_1}s_{\theta_1}s_{\psi_1} + c_{\phi_1}c_{\psi_1} & s_{\phi_1}s_{\theta_1}c_{\psi_1} - c_{\phi_1}s_{\psi_1} \\ -s_{\theta_1} & c_{\theta_1}s_{\psi_1} & c_{\theta_1}c_{\psi_1} \end{bmatrix} \tag{A.19}$$

and

$$\mathbf{K}_2 = \begin{bmatrix} (c_{\theta_2}c_{(\psi_3-\psi_2)})/c_{\theta_3} & (s_{(\psi_3-\psi_2)} & 0 \\ -c_{\theta_2}s_{(\psi_3-\psi_2)} & c_{(\psi_3-\psi_2)} & 0 \\ (a_{x_1}c_{\phi_3} + a_{y_1}s_{\phi_3})/c_{\theta_3} & (s_{\theta_3}s_{(\psi_3-\psi_2)})/c_{\theta_3} & 1 \end{bmatrix}. \tag{A.20}$$

Note that a_{x_1} and a_{y_1} are elements of the homogeneous matrix equivalent to location vector $\mathbf{x}_1$ similar to eq. (A.10a).

A.3.4 Jacobians of the Inversion

The Jacobian of the inversion of a location vector $\ominus\mathbf{x} = (x', y', z', \phi', \theta', \psi')^T$ is given by

$$\mathbf{J}_{\ominus}\{\mathbf{x}\} = \left.\frac{\partial(\ominus\mathbf{y})}{\partial\mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}} \tag{A.21}$$

$$= \begin{bmatrix} -\mathbf{R}^T & \mathbf{N} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}, \tag{A.22}$$

where

$$\mathbf{N} = \begin{bmatrix} n_y x - n_x y & -n_z x c_\phi - n_z y s_\phi & 0 \\ o_y x - o_x y & -o_z x c_\phi - o_z y s_\phi + z s_\theta s_\psi & z' \\ a_y c - a_x y & -a_z x c_\phi - a_z y s_\phi + z s_\theta c_\psi & -y' \end{bmatrix}, \tag{A.23}$$

$$\mathbf{Q} = \begin{bmatrix} -a_z/(1-a_x^2) & -a_y c_\phi/(1-a_x^2) & n_x a_x/(1-a_x^2) \\ a_y/(1-a_x^2)^{0.5} & -a_z c_\phi/(1-a_x^2)^{0.5} & o_x/(1-a_x^2)^{0.5} \\ a_z a_x/(1-a_x^2) & -o_x c_\psi/(1-a_x^2) & -n_x/(1-a_x^2) \end{bmatrix}. \tag{A.24}$$

Note that like above, the components a_x, a_y and so on are taken from the homogeneous matrix corresponding to the input location vector $\mathbf{x}$ (see eq. (A.10a)).

A.3.5 Differential Transforms

Let A and B be two references, whose relative location is given by location vector $\mathbf{x}_{AB}$. A differential change to B which takes it to B' can be represented as a differential location vector $\mathbf{d} = (dx, dy, dz, d\phi, d\theta, d\psi)^T$. There are two ways of representing the transform from A to B':

$$\mathbf{x}_{AB'} = \mathbf{x}_{AB} \oplus \mathbf{d}_B \tag{A.25}$$

$$\mathbf{x}_{AB'} = \mathbf{d}_A \oplus \mathbf{x}_{AB} \tag{A.26}$$

Using the Jacobian of the composition, this can be transformed into

$$\mathbf{x}_{AB'} = \mathbf{x}_{AB} + \mathbf{J}_{2\oplus}\{\mathbf{x}_{AB}, \mathbf{0}\}\mathbf{d}_B \tag{A.27}$$

$$\mathbf{x}_{AB'} = \mathbf{x}_{AB} + \mathbf{J}_{1\oplus}\{\mathbf{0}, \mathbf{x}_{AB}\}\mathbf{d}_A. \tag{A.28}$$

Hence, $\mathbf{J}_{2\oplus}\{\mathbf{x}_{AB}, \mathbf{0}\}\mathbf{d}_B = \mathbf{J}_{1\oplus}\{\mathbf{0}, \mathbf{x}_{AB}\}\mathbf{d}_A$, and therefore

$$\mathbf{d}_A = \underbrace{\mathbf{J}_{1\oplus}^{-1}\{\mathbf{0}, \mathbf{x}_{AB}\}\mathbf{J}_{2\oplus}\{\mathbf{x}_{AB}, \mathbf{0}\}}_{\mathbf{J}_{AB}}\mathbf{d}_B \tag{A.29}$$

$$\mathbf{d}_B = \underbrace{\mathbf{J}_{2\oplus}^{-1}\{\mathbf{x}_{AB}, \mathbf{0}\}\mathbf{J}_{1\oplus}\{\mathbf{0}, \mathbf{x}_{AB}\}}_{\mathbf{J}_{BA}}\mathbf{d}_A \tag{A.30}$$

A.3.6 Operations with SP-Locations

Inversion

Given an uncertain location $\mathbf{L}_{AB} = (\hat{\mathbf{x}}_{AB}, \hat{\mathbf{p}}_B, \mathbf{C}_B, \mathbf{B}_B)$, its inverse $\mathbf{L}_{BA}$ is given by:

$$\begin{array}{ll}
\begin{aligned}
\mathbf{x}_{BA} &= \ominus \mathbf{x}_{AB} \\
&= \ominus(\hat{\mathbf{x}}_{AB} \oplus \mathbf{B}_B^T \mathbf{p}_B) \\
&= \ominus \mathbf{B}_B^T \mathbf{p}_B \ominus \hat{\mathbf{x}}_{AB} \\
&= \ominus \mathbf{B}_B^T \mathbf{p}_B \oplus \hat{\mathbf{x}}_{BA} \\
&= \hat{\mathbf{x}}_{BA} \ominus \mathbf{J}_{AB} \mathbf{B}_B^T \mathbf{p}_B
\end{aligned}
&
\begin{aligned}
\mathbf{p}_A &= -\mathbf{B}_B \mathbf{J}_{AB} \mathbf{B}_B^T \mathbf{p}_B \\
\mathbf{B}_A &= \mathbf{B}_B \\
\mathbf{C}_A &= \mathbf{B}_B \mathbf{J}_{AB} \mathbf{B}_B^T \mathbf{C}_B \mathbf{B}_B \mathbf{J}_{AB}^T \mathbf{B}^T
\end{aligned}
\end{array} \tag{A.31}$$

Composition

The composition of two uncertain locations $\mathbf{L}_{WF} = (\hat{\mathbf{x}}_{WF}, \hat{\mathbf{p}}_F, \mathbf{B}_F, \mathbf{C}_F)$ and $\mathbf{L}_{FE} = (\hat{\mathbf{x}}_{FE}, \hat{\mathbf{p}}_E, \mathbf{B}_E, \mathbf{C}_E)$ is given by $\mathbf{L}_{WE}$ with

$$\begin{aligned}
\hat{\mathbf{x}}_{WE} &= \hat{\mathbf{x}}_{WF} \oplus \hat{\mathbf{x}}_{FE} \\
\hat{\mathbf{p}}_E^W &= \mathbf{B}_E \mathbf{J}_{EF} \mathbf{B}_F^T \mathbf{p}_F + \mathbf{p}_E \\
\mathbf{C}_E^W &= \mathbf{B}_E \mathbf{J}_{EF} \mathbf{B}_F^T \mathbf{C}_F \mathbf{B}_F \mathbf{J}_{EF}^T \mathbf{B}_E^T + \mathbf{C}_E
\end{aligned}$$

Centering

A location $\mathbf{L}_{WE} = (\hat{\mathbf{x}}_{WE}, \hat{\mathbf{p}}_E, \mathbf{C}_E, \mathbf{B}_E)$ with $\hat{\mathbf{p}}_E \neq \mathbf{0}$ is transformed into location $\mathbf{L}_{WE'} = (\hat{\mathbf{x}}_{WE'}, \hat{\mathbf{p}}'_E, \mathbf{C}'_E, \mathbf{B}'_E)$ with $\hat{\mathbf{p}}_E = \mathbf{0}$, called *centered*, in the following way:

$$\begin{aligned}
\hat{\mathbf{x}}_{WE'} &= \hat{\mathbf{x}}_{WE} \oplus \mathbf{B}_E^T \hat{\mathbf{p}}_E \\
\mathbf{C}_{E'} &= (\mathbf{B}_E \mathbf{J}_{2\oplus}^{-1}\{\mathbf{B}_E^T \hat{\mathbf{p}}_E, \mathbf{0}\} \mathbf{B}_E^T) \mathbf{C}_E (\mathbf{B}_E \mathbf{J}_{2\oplus}^{-1}\{\mathbf{B}_E^T \hat{\mathbf{p}}_E, \mathbf{0}\} \mathbf{B}_E^T)^T
\end{aligned}$$

Bibliography

H. Andreasson, R. Triebel, and W. Burgard. Improving plane extraction from 3d data by fusing laser data and vision. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, pages 2656–2661, Edmonton, Canada, August 2005.

K. O. Arras, J. Castellanos, M. Schilt, and R. Siegwart. Feature-based multi-hypothesis localization and tracking using geometric constraints. *Robotics and Autonomous Systems*, 1056:1–13, 2003.

K.O. Arras. *Feature-Based Robot Navigation in Known and Unknown Environments*. PhD thesis, EPFL, 2003.

N. Ayache and O.D. Faugeras. Building, registrating, and fusing noisy visual maps. *International Journal on Robotics Research*, 7(6):45–65, 1988. ISSN 0278-3649.

Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House, 1993.

Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *ACM Comput. Surv.*, 17(1):75–145, 1985. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/4078.4081.

Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.121791.

J.-Y. Bouguet. Complete camera calibration toolbox for matlab, 2000. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`. last visited on April 12th, 2006.

W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114:3–55, 1999.

J. A. Castellanos and J.D. Tardós. *Mobile Robot Localization And Map Building*. Kluwer Academic Publishers, 1999.

J. A. Castellanos, M. M. Montiel, J. Neira, and J. D. Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15:948–952, 1999.

H. Christensen. Slam summer school. 2002. URL `http://www.cas.kth.se/SLAM/toc.html`. last visited on April 12th, 2006.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Computer Graphics*, 30:303–312, 1996. URL `citeseer.ist.psu.edu/article/curless96volumetric.html`.

Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328, Detroit, USA, May 1999.

A. Diosi and L. Klemman. Uncertainty of line segments extracted from static sick pls laser scans. In *Proceedings of Australasian Conference on Robotics and Automation*, 2003.

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

Hugh F. Durrant-Whyte. An autonomous guided vehicle for cargo handling applications. *Int. J. Rob. Res.*, 15(5):407–440, 1996. ISSN 0278-3649.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989a. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.30720.

A. Elfes. *A probabilistic framework for robot perception and navigation*. PhD thesis, Carnegie Mellon University, 1989b.

T.-J. Fan, G. Medioni, and R. Nevatia. Segmented descriptions of 3-d surfaces. *IEEE Jounal of Robotics and Automation*, 3:527–538, 1988.

Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993. ISBN 0-262-06158-9.

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

D. Fox, M. Burgard, and Thrun S. Active markov localization for mobile robots. In *Robotics and Autonomous Systems*, volume 25, pages 195–207. 1998.

D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

D. Fox, S. Thrun, W. Burgard, and F. Dellaert. *Particle filters for mobile robot localization*. 2001. URL `citeseer.ist.psu.edu/fox01particle.html`.

R.C. Gonzalez and S.E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEEE Proceedings of Radar and Signal Processing*, volume 140, pages 107–113, 1993.

M. S. Grewal and A. P. Andrews. *Kalman filtering: theory and practice*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-211335-X.

J. Guivant and E. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001a.

J. Guivant and E. Nebot. Compressed filter for real time implementation of simultaneous localization and map building. In *FSR 2001 International Conference on Field and Service Robots*, pages 309–314, Helsinki, Finland, June 2001b.

J. Guivant, F. Masson, and E. Nebot. Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, 2002.

D. Haehnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environmnets from raw laser range measurements. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 206–211, Las Vegas, USA, October 2003.

Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1106, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7822-4.

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

A. Hoover, J.-B. Gillian, X. Jiang, P. J. Flynn, Horst Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):673–689, July 1996.

H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *ACM Siggraph*, 1992.

Hugues Hoppe. Progressive meshes. *Computer Graphics*, 30:99–108, 1996.

J. Horn and G. Schmidt. Continuous localization of a mobile robot based on 3d-laser-range-data, predicetd sensor images, and dead-reckoning. In *Robotics and Autonomous Systems*, volume 14, pages 99–118. 1995a.

J. Horn and G. Schmidt. Continuous localization for long-range indoor navigation of mobile robots. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, volume 1, pages 387–394, Nagoya, Japan, May 1995b.

P. V. C. Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, CERN, 1959.

P. Jensfelt and S. Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, 17:748–760, 2001.

X. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Mach. Vision Appl.*, 7(2):115–122, 1994. ISSN 0932-8092. doi: http://dx.doi.org/10.1007/BF01215806.

S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

T. Kahlmann and H. Ingensand. Calibration and improvements of the high-resolution range-imaging camera SwissRanger. In *Optical Materials in Defence Systems Technology. Edited by Vere, Anthony W.; Grote, James G.; Kajzar, Francois. Proceedings of the SPIE, Volume 5665, pp. 144-155 (2004).*, pages 144–155, December 2004. doi: 10.1117/12.582513.

Kenichi Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science Inc., New York, NY, USA, 1996. ISBN 0444824278.

P. Kohlhepp, P. Pozzo, and R. Dillmann. Sequential 3d-slam for mobile action planning. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 722–729, Sendai, Japan, September 2004.

E. Kraft. A quaternion-based unscented kalman filter for orientation tracking. In *Proceedings of the 6th International Conference on Infornation Fusion*, 2003.

P. Lamon, A. Tapus, E. Glauser, N. Tomatis, and R. Siegwart. Environmental modeling with fingerprint sequences for topological global localization. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 3781–3786, Las Vegas, USA, October 2003.

R. Lange and P. Seitz. Solid-state, time-of-flight range camera. *IEEE Journal of Quantum Electronics*, 37(3):390–397, 2001.

J. J. Leonard and H.F. Durrand-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of International Workshop on Intelligent Robots and Systems (IROS)*, 1991.

J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotcs and Automation*, 7, 1991.

Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models of indoor environments with mobile robots. In *Proceedings of International Conference on Machine Learning (ICML)*, 2001.

F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 935–938, Seattle, USA, June 1994.

F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.

P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institue of Science of India*, number 12, 1936.

I. Mahon and S. Williams. Three-dimensional robotic mapping. In *Proceedings of Australiasian Conference on Robotics and Automation*, Brisbane, Australia, December 2003.

M. Matos, V. Santos, and P. Dias. 3d reconstruction of real world scenes using a low-cost 3d range scanner. In *In Proceedings of the 4th Conference of Construction Applications of Virtual Reality - CONVR2004*, 2004.

M. Montemerlo, S. Thrun, D. Koller, and Wegbreit B. Fastslam: a factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

M. Montermerlo, S. Thrun, Koller D., and B. Wegbreit. Fastslam 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, 2003.

Hans Moravec. Sensor fusion in certainty grids for mobile robots. *AI Mag.*, 9(2):61–74, 1988. ISSN 0738-4602.

Hans Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121, March 1985.

P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environement modelling. In *International Symposium of Robotics Research*, 1989.

A. Nüchter and H. Surmann. 6d slam with an application in autonomous mine mapping. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 1998–2003, New Orleans, USA, April 2004.

P. M. Newman, D. M. Cole, and K.L. Ho. Outdoor slam using visual appearance and laser ranging. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, May 2006.

University of Southern Florida. Range image database. URL `http://marathon.csee.usf.edu/range/DataBase.html`. last visited on April 12th, 2006.

T. Oggier, M. Lehmann, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, and N. Blanc. An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution (swissranger). In *SPIE Optical Design and Engineering*, volume 5249, pages 534–545, 2004.

A. Reina and J. Gonzalez. Characterization of a radial laser scanner for mobile robot navigation. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 579–585, Grenoble, France, September 1997.

H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

V. Sequeira, J.G.M. Gonçalves, and Ribeiro M.I. High-level surface descriptions from composite range images. In *IEEE International Symposium on Computer Vision*, pages 163 – 168, November 1995.

V. Sequeira, K. Ng, E. Wolfart, J.G.M. Gonçalves, and D. Hogg. Automated reconstruction of 3d models from real environments. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54:1–22, 1999.

R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

C. Stachniss, G. Grisetti, and W. Burgard. Recovering particle diversity in a rao-blackwellized particle filter for slam after actively closing loops. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 655 – 660, Barcelona, Spain, April 2005.

P. Steinhaus and R. Dillmann. Aufbau und modellierung des rosi scanners zur 3d-tiefenbildakquisition. In *18. Fachgespräch "Autonome Mobile Systeme (AMS 2003)"*, Karlsruhe, Germany, December 2003.

Minsoo Suk and Suchendra M. Bhandarkar. *Three-Dimensional Object Recognition from Range Images*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992. ISBN 0387701079.

H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Journal of Robotics and Autonomous Systems*, 45(3-4):181–198, 2003.

Hartmut Surmann, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. Fast acquiring and analysis of three dimensional laser range data. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 59–66. Aka GmbH, 2001. ISBN 3-89838-028-9.

J.D. Tardós. Representing partial and uncertain sensorial information using the theroy of symmetries. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1992.

S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

S. Thrun, W. Burgard, D. Chakrabarti, R. Emery, Y. Liu, and C. Martin. A real-time algorithm for acquiring multi-planar volumetric models with mobile robots. In *Proceedings of the 10th International Symposium of Robotics Research (ISRR'01)*, Lorne, Australia, 2001. Springer.

N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44:3–14, 2003.

Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. pages 221–244, 1992.

R. Van der Merwe, N. de Freitas, A. Doucet, and E. Wan. The unscented particle filter. Technical report, Cambridge University Engineering Department, 2000.

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

J. Weingarten, G. Gruener, and R. Siegwart. A fast and robust 3d feature extraction algorithm for structured environment reconstruction. In *Proceedings of International Conference on Advanced Robotics (ICAR)*, pages 1142–1147, Coimbra, Portugal, June 2003.

J. Weingarten, G. Gruener, and R. Siegwart. Probabilistic plane fitting in 3d and an application to robotic mapping. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 927–932, New Orleans, USA, April 2004a.

J. Weingarten, G. Gruener, and R. Siegwart. A state-of-the-art 3d sensor for robot navigation. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, pages 2155–2160, Sendai, Japan, September 2004b.

G. Welch and G. Bishop. An introduction to the kalman filter, February 2001.

Wikipedia. the free encyclopedia, 2006. URL `http://en.wikipedia.org`. last visited on April 12th, 2006.

O. Wulf and B. Wagner. Fast 3d-scanning methods for laser measurement systems. In *Proceedings of 14th International Conference on Control Systems and Computer Science (CSCS14)*, Bucharest, Romania, July 2003.

C. Ye and J. Borenstein. Characterization of a 2-d laser scanner for mobile robot obstacle negotiation. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 2512–2518, Washington DC, USA, May 2002.

N. Yokoya and M. D. Levine. Range image segmentation based on differential geometry: a hybrid approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:643–649, 1989.

Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 666–673, Corfu, Greece, September 1999.

Curriculum Vitae

I was born on the 16th of July 1975 in Bonn (Germany). After moving to Geneva at the age of five, I entered the Deutsche Schule Genf (DSG) in 1981, which I finished 1994 by obtaining the German Abitur. Then I moved to Germany to start Computer Science studies at the Technische Universität Karlsruhe. After a semester project at the University of Port Elizabeth (South Africa) in 1997/1998, I moved back to Karlsruhe and finished my studies in 2001. My master thesis was entitled "Entwicklung eines Verfahrens zur Lokalisierung und Entnahme chaotisch palettierter Objekte mit Tiefenbildern" and carried out at the Fraunhofer Institut für Produktionstechnik und Automatisierung (IPA) in Stuttgart. In 2002, I moved to Lausanne (Switzerland) to start my doctoral thesis in mobile robotics under the supervision of Prof. Siegwart at the Swiss Federal Institute of Technology (EPFL).