

Highly Accurate Lidar-Based Mapping and Localization for Mobile Robots

Alexander Schaefer

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Dissertation zur Erlangung des akademische Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard



Highly Accurate Lidar-Based Mapping and Localization for Mobile Robots

Alexander Schaefer

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Technische Fakultät, Albert-Ludwigs-Universität Freiburg

Dekan	Prof. Dr. Rolf Backofen
Erstgutachter	Prof. Dr. Wolfram Burgard
Zweitgutachter	Prof. Dr. Bernhard Nebel
Tag der Disputation	13. März 2020

Zusammenfassung

Die vorliegende Dissertation stellt neuartige Konzepte, Methoden und Algorithmen vor, um eine Karte der Umgebung eines mobilen Roboters zu erstellen und seine Position anhand dieser Karte zu bestimmen. Sie beschäftigt sich also mit den grundlegenden Fähigkeiten eines jeden mobilen Roboters. Ohne Kartierung und Positionsschätzung kann er sich nicht in der Welt zurechtfinden. Dabei spielen die Daten, die die Sensoren des Roboters liefern, eine wichtige Rolle. Im Folgenden gehen wir immer davon aus, dass der Roboter mit einem Lidarsensor ausgestattet ist. Lidar steht für „light detection and ranging“, was bereits auf die Funktionsweise dieser Art von Sensoren hindeutet: Typischerweise sendet ein Lidarsensor kontinuierlich Lichtpulse aus. Nach jedem Puls wartet er eine gewisse Zeit darauf, dass das Licht von einem Objekt in der Umgebung reflektiert wird. Wird der Puls reflektiert, misst der Sensor die Zeit, die zwischen Aussenden des Pulses und Detektion der Reflexion vergangen ist, und berechnet daraus den Abstand zum reflektierenden Objekt.

Im ersten Teil dieser Arbeit stellen wir ein neuartiges mathematisches Sensormodell für Lidarsensoren vor. Es beschreibt die Interaktion zwischen Sensor und Umgebung formal und ist deshalb Herzstück eines jeden Kartierungs- oder Positionsbestimmungsalgorithmus. Im Gegensatz zu bisherigen Arbeiten auf diesem Gebiet modelliert es die Wahrscheinlichkeit, dass ein vom Sensor emittierter Lichtstrahl reflektiert wird, als exponentiellen Zerfallsprozess. Daraus ergeben sich einige Vorteile. Beispielsweise verarbeitet unsere Methode nicht nur einen Teil, sondern die vollständige Information über den Strahlengang der Lichtpulse, was zu einer höheren Genauigkeit bei der Positionsschätzung führt. Weiterhin ist unser Sensormodell das erste, das zwar den Strahlengang berücksichtigt, dabei aber nicht darauf angewiesen ist, dass der Raum zuvor gerastert, sprich in Zellen eingeteilt wird. Das macht es zum ersten Sensormodell, das sowohl Kartierung als auch Positionsschätzung mit kontinuierlichen Karten ermöglicht – eine Tatsache, auf die wir im dritten Teil dieser Arbeit genauer eingehen.

Im zweiten Teil der vorliegenden Dissertation präsentieren wir eine Me-

thode, um den Informationsgehalt von Rasterkarten aus Lidardaten wesentlich zu erhöhen. Während jede Zelle einer herkömmlichen Rasterkarte meist den wahrscheinlichsten Kartenwert enthält, zeigen wir, wie man dort die komplette Wahrscheinlichkeitsverteilung über alle möglichen Kartenwerte in kompakter Form speichert, ohne die Berechnungskomplexität zu erhöhen. Die mathematische Herleitung erfolgt in geschlossener Form und kommt ohne Näherungen aus. Unsere Experimente zeigen, dass die resultierenden Karten die Güte der Positionsschätzung gegenüber herkömmlichen Karten signifikant verbessern.

Im dritten Teil geht es nicht mehr um Rasterkarten, sondern um eine neuartige kontinuierliche Kartenrepräsentation, die auf der diskreten Kosinustransformation beruht. Der wesentliche Vorteil dieser Methode gegenüber bestehenden kontinuierlichen Lidarkarten liegt darin, dass sie sowohl zur Kartierung als auch zur Positionsbestimmung eingesetzt werden kann. Bestehende Ansätze erlauben es zwar, kontinuierliche Karten zu erstellen. Um bei der Positionsschätzung die Wahrscheinlichkeit einer gegebenen Lidarmessung auf Basis einer solchen Karte zu bestimmen, musste die Karte aber bisher wieder gerastert werden. Ein weiterer Vorteil des neu vorgestellten Kartentyps gegenüber vergleichbaren Arbeiten ist seine hohe Speichereffizienz.

In den verbleibenden Teilen der Arbeit wenden wir uns einem weiteren wichtigen Aspekt der Kartierung und Positionsschätzung zu: der Erkennung von Features. Unter Features verstehen wir abstrakte Merkmale in Sensordaten, zum Beispiel Linien in einem Kamerabild oder ebene Flächen in einer Punktwolke. Nutzt man Features statt Sensorrohdaten zur Kartierung, erhält man eine Featurekarte. Featurekarten unterscheiden sich dadurch von den oben beschriebenen dichten Kartentypen, sprich kontinuierlichen und Rasterkarten, dass sie die Welt als eine Sammlung von Objekten in ansonsten leerem Raum modellieren. Dieser Ansatz spart Speicherplatz, vereinfacht die Integration von Daten aus unterschiedlichen Sensormodalitäten, er kann die Robustheit des Gesamtsystems erhöhen und ermöglicht es, semantische Informationen in die Karten aufzunehmen.

Unsere erste Arbeit zum Thema Featureerkennung beschäftigt sich mit der Detektion von zusammenhängenden linienförmigen Konturen in planaren Lidardaten. Die Erkennung solcher Konturen ist beispielsweise nützlich, um den Grundriss eines Gebäudes in den Daten zu erkennen, die ein horizontal am Roboter angebrachter Lidar liefert. Der entwickelte Ansatz beruht auf dem Prinzip, die kombinierte Wahrscheinlichkeit aller Messungen zu maximieren. Dabei nutzt er die in den Sensordaten enthaltene Information über den Strahlengang der vom Sensor ausgesendeten Lichtpulse vollständig aus. Im Rahmen ausführlicher Experimente mit echten und simulierten Daten zeigen wir, dass die entwickelte Methode eine signifikant höhere Genauigkeit

erreicht als vergleichbare Ansätze.

Aufbauend auf dieser Arbeit beschreibt der nächste Teil der Dissertation einen analogen Ansatz zur Erkennung finiter Ebenen in 3-D-Lidardaten. Da der Datensatz, der normalerweise zum Vergleich solcher Ebenenerkennungsalgorithmen herangezogen wird, in die Jahre gekommen ist und zahlreiche Nachteile aufweist, stellen wir weiterhin einen Nachfolger vor, der synthetisch generierte Lidardaten enthält.

Der letzte Teil der Dissertation dreht sich um Pfosten-Features in 3-D-Lidardaten, also Objekte, die einem stehenden Zylinder gleichen, zum Beispiel Poller, Baumstämme, Masten von Straßenschildern und Lampen. Wir stellen ein komplettes System zur Positionsbestimmung basierend auf diesen Features vor, bestehend aus einem Modul zur Featuredetektion, einem Kartierungsmodul und einem Positionsschätzer. Vergleichsexperimente mit ähnlichen Methoden zeigen, dass der vorgeschlagene Ansatz die Roboterposition genauer schätzt. Während andere Ansätze meist nur auf einem wenige Minuten langen, nichtöffentlichen Datensatz ausgewertet werden, demonstrieren wir zusätzlich die Leistungsfähigkeit und Robustheit unseres Ansatzes über lange Zeiträume hinweg. Dazu evaluieren wir ihn auf einem frei verfügbaren Langzeitdatensatz, der 35 Stunden Daten enthält, die über einen Zeitraum von 15 Monaten aufgenommen wurden.

Abstract

This thesis contributes novel concepts, methods, and algorithms to the topic of mapping and localization for mobile robots. Mapping is the process of building a model of the robot’s environment based on a collection of sensor measurements, while localization refers to the process of using the resulting map and incoming sensor measurements to estimate the current location of the robot. Together, mapping and localization enable the robot to navigate the world – a prerequisite for any meaningful application of a mobile robot.

All of our contributions assume that the mobile robot is equipped with a lidar sensor. Lidar is an acronym of “light detection and ranging”, hinting at the operating principle of a lidar sensor: Typically, it continuously emits light pulses, waits for each pulse to be reflected by a nearby object, measures the time of flight, and uses this measurement to compute the distance to the object.

Our first contribution is a novel mathematical model for lidar sensors. By describing the interaction between the sensor and its environment mathematically, it constitutes the theoretical centerpiece of any mapping and localization algorithm. In contrast to related approaches, the proposed model formulates the reflection probability of a light ray emitted by the lidar as an exponential decay process, hence the name decay-rate model. This formulation yields several advantages compared to existing approaches, the most important being that the model makes use of the full ray-path information contained in the measurements. In this way, it achieves higher localization accuracy than comparable methods, which process only part of this information. To the best of our knowledge, it is also the first beam-based lidar sensor model that is not bound to the notion of voxels. Consequently, the decay-rate model is the first model to truly enable continuous mapping, a fact we make use of in our third contribution.

The second contribution advances the way in which grid maps produced by the reflection model or the decay-rate model represent the world. Conventionally, these models are used to create maximum-likelihood grid maps of the robot’s environment. Maximum-likelihood maps encode for each cell the

mode of the underlying probability distribution over all possible map values. In this thesis, we show that it is possible to represent the full posterior probability distribution of each cell using only two variables – without increasing the computational complexity required to create the map. Our mathematical proof is carried out in closed form and without any simplifications. We also demonstrate that keeping track of the full posterior significantly improves localization performance compared to working with the mode of the distribution only.

The third contribution introduces another innovation to the way the map represents the environment. Instead of tessellating the space and assigning a value to each cell, it proposes a novel continuous representation that is based on the discrete cosine transform. The resulting maps are hence called DCT maps. Built upon the decay-rate model, the major advantage of DCT maps over related continuous lidar-based mapping approaches lies in their consistent nature, which allows to use them not only for mapping, but also for localization: While other continuous maps require re-tessellation to compute the probability of a given lidar measurement, DCT maps naturally support this operation. Furthermore, our experiments show that DCT maps outperform other map types in terms of memory efficiency.

The remainder of this thesis addresses another highly relevant aspect of mapping and localization: feature extraction. In contrast to dense map representations like grid maps or continuous maps, feature-based maps model the environment as a collection of objects in empty space, yielding memory-efficient maps that abstract from the modality of the sensors in use, that improve system robustness, and that can encode semantics.

First, we focus on polylines extracted from 2-D lidar scans. The polyline detection method proposed within the scope of our fourth contribution follows a maximum-likelihood approach that considers the full ray-path information contained in the lidar measurements. Extensive real-world and simulated experiments show that this probabilistic approach outperforms the rich collection of state-of-the-art methods in terms of accuracy.

Building upon this method, our fifth contribution suggests an analogous approach to extract finite planes from 3-D lidar scans. Due to the deficiencies of the most popular benchmarking dataset for plane extraction algorithms based on lidar data, we also present a novel synthetic dataset in the scope of this work.

Our last contribution does not only present a novel approach to detect pole features in 3-D lidar scans, but a complete mapping and localization framework based on poles. The comparative experiments conducted in the scope of this work already demonstrate the proposed method’s superior localization accuracy. In addition, while related methods are often tested on

proprietary datasets with durations of only a few minutes, we showcase the performance and robustness of our approach by evaluating it on a public long-term dataset that contains 35 hours of data recorded over the course of 15 months.

Alexander Schaefer, Georges-Koehler-Allee 080, 79110 Freiburg, Germany



Prof. Dr. Hannah Bast
University of Freiburg
Georges-Köhler-Allee 051
79110 Freiburg
Germany

Alexander Schaefer
University of Freiburg
Georges-Koehler-Allee 080
79110 Freiburg
Germany

Phone: +49 761/203-8176
Email: aschaef@cs.uni-freiburg.de
Web: <http://www2.informatik.uni-freiburg.de/~aschaef/>

May 13, 2019

Application for submitting a cumulative dissertation

Dear Professor Bast,

In accordance with section 8, subsection 3 of the new doctoral regulations (Promotionsordnung) of the Technical Faculty, in effect since October 1, 2016, I herewith ask for permission to submit my dissertation as a cumulative dissertation.

Section 8, subsection 3 of the new doctoral regulations requires all works that are part of the cumulative dissertation

- 1) to target the same overarching scientific problem,
- 2) to be published in internationally renowned conferences,
- 3) to be in essential parts authored by the doctoral candidate.

The works that will be part of the dissertation fulfill all three criteria:

- 1) They all are closely related to the problem of lidar-based mapping and localization for mobile robots.
- 2) They have been accepted and published by the IEEE International Conference on Robotics and Automation (ICRA), by the IEEE International Conference on Intelligent Robots and Systems (IROS), or by the journal IEEE Robotics and Automation Letters (RA-L). For details, please see the paper listing below.
- 3) For all papers, I am either the main author or one of the two main authors, who contributed equally to the paper. For details, please see the paper listing below.

The following listing contains works that will be part of the cumulative dissertation. For each work, the listing states the corresponding paper title, the conference or journal the paper was published in, and the contributions of the individual authors.

Listing of works included in the cumulative dissertation

- 1) Alexander Schaefer, Lukas Luft, Wolfram Burgard:
An Analytical Lidar Sensor Model Based on Ray Path Information.
IEEE Robotics and Automation Letters (RA-L), Volume 2, Issue 3, July 2017.

The first two authors contributed equally to the publication. I developed the idea of considering the distances that all rays travel within a grid cell, as opposed to counting misses only. Lukas Luft came up with the idea to model the interaction of a laser ray with the environment as exponential decay process. I implemented the software and carried out the experiments. We wrote the text of the paper together. Wolfram Burgard provided general consultation and corrections.

- 2) Lukas Luft, Alexander Schaefer, Tobias Schubert, Wolfram Burgard:
Closed-Form Full Map Posteriors for Robot Localization with Lidar Sensors.
Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada, September 2017.

The first two authors contributed equally to the publication. Lukas Luft developed the idea for the paper and for the mathematical framework. Both authors elaborated the framework and wrote the publication collaboratively. I implemented all software and performed the experiments. Tobias Schubert and Wolfram Burgard contributed consultation and part of the writing.

- 3) Alexander Schaefer, Lukas Luft, Wolfram Burgard:
DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform.
IEEE Robotics and Automation Letters (RA-L), Volume 3, Issue 2, April 2018. Presented at the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 2018.

I am the main author of the paper. I developed the idea for the paper, devised the theoretical framework, implemented all software, conducted and evaluated all experiments, and wrote the publication. Lukas Luft and Wolfram Burgard provided general consultation.

- 4) Alexander Schaefer, Daniel Büscher, Lukas Luft, Wolfram Burgard:

A Maximum Likelihood Approach to Extract Polylines from 2-D Laser Range Scans.

Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018.

I am the main author of the work. I developed the idea for the paper, devised the presented algorithm, implemented the software used in the experiments, performed part of the experiments, and wrote the paper. Daniel Büscher conducted the reference experiments and provided consultation. Lukas Luft and Wolfram Burgard provided consultation.

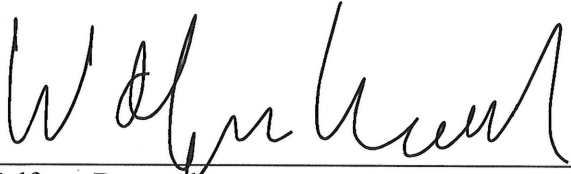
- 5) Alexander Schaefer, Johan Vertens, Daniel Büscher, Wolfram Burgard:

A Maximum Likelihood Approach to Extract Finite Planes from 3-D Laser Scans.

Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, May 2019.

I am the main author of the paper. I developed the idea for the paper, conceived the presented algorithm, implemented the software used in the experiments, conducted part of the experiments, and wrote the paper. Johan Vertens created the simulated dataset, performed reference experiments, and provided consulting. Daniel Büscher accelerated part of the implementation and performed reference experiments. Wolfram Burgard provided consultation.

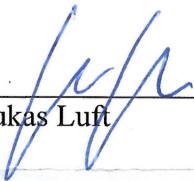
With their signatures below, all involved authors certify the correctness of the above description.



Wolfram Burgard



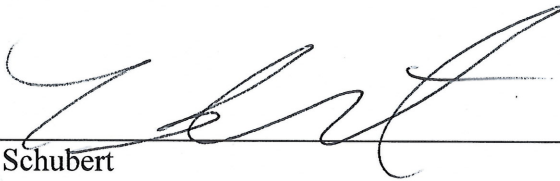
Daniel Büscher



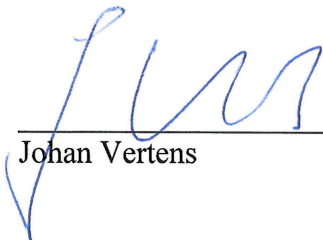
Lukas Luft



Alexander Schaefer



Tobias Schubert



Johan Vertens

Best regards,



Alexander Schaefer, Georges-Köhler-Allee 080, 79110 Freiburg, Germany



PhD Board of the Technical Faculty
University of Freiburg
Georges-Köhler-Allee 101
79110 Freiburg
Germany

Alexander Schaefer
University of Freiburg
Georges-Köhler-Allee 080
79110 Freiburg
Germany

Phone: +49 761/203-8176
Email: aschae@cs.uni-freiburg.de
Web: www.informatik.uni-freiburg.de/~aschae/

February 18, 2020

Addendum to application for submitting a cumulative dissertation

Dear PhD Board of the Technical Faculty,

In May 2019, I applied for submitting a cumulative dissertation. In this application, I listed five scientific papers which, in accordance with section 8, subsection 3 of the new doctoral regulations (Promotionsordnung) of the Technical Faculty as of October 1, 2016,

- 1) target the same overarching scientific problem,
- 2) were published in internationally renowned conferences,
- 3) and were in essential parts authored by me.

For all of these papers, I stated the corresponding paper title, the conference or journal the paper was published in, and the contributions of the individual authors.

At the time of application submission, a sixth paper that also fulfilled all three of the criteria above was still under review. After it was accepted for publication on June 21, 2019, I included it in my cumulative dissertation.

In order to avoid confusion, I would like to extend the list of papers contained in my original application as follows, mentioning the paper title, the conference, and describing the contributions of the individual authors.

■ **Listing of works included in the cumulative dissertation
(continued)**

- 6) Alexander Schaefer, Daniel Büscher, Johan Vertens,
Lukas Luft, Wolfram Burgard:
**Long-Term Urban Vehicle Localization using Pole
Landmarks Extracted from 3-D Lidar Scans.**
*Proceedings of the European Conference on Mobile Robots
(ECMR), Prague, Czech Republic, September 2019.*

I am the main author of the paper. I developed the idea for the paper, conceived the presented algorithms, implemented the software used in the experiments, designed and conducted the experiments, and wrote the paper. Daniel Büscher developed and implemented the refined registration of the NCLT lidar scans. He also helped analyze the datasets, provided consultation regarding the design of the experiments, and helped with conceptual and implementation problems during development. Johan Vertens, Lukas Luft, and Wolfram Burgard provided consultation regarding the overall concept of the work, the implementation, and experiments.

With their signatures below, all involved authors certify the correctness of the above description.



Wolfram Burgard



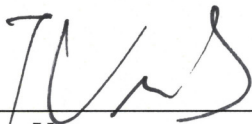
Daniel Büscher



Lukas Luft



Alexander Schaefer



Johan Vertens

Best regards,



Acknowledgements

First and foremost, this dissertation would not have been possible without the support of my supervisor Wolfram Burgard. Wolfram, thank you for your supervision, for the opportunity to work on exciting research projects in an international team, for providing your students with modern equipment, for giving us the opportunity to present our research at top conferences, and for connecting us to researchers all over the world.

Second, I thank all my coauthors, especially Lukas Luft, Daniel Büscher, Johan Vertens, and Tobias Schubert. Thank you for your friendly and competent collaboration and for creating a constructive, creative, and motivating atmosphere.

I also thank all my colleagues who make our group a great place to work at and spend time in, in particular Daniel Büscher, Johan Vertens, Lukas Luft, Marina Kollmitz, Michael Krawez, and Tim Caselitz.

Last but not least, I cordially thank all researchers who supported this work by providing access to their own research: Maani Ghaffari Jadidi for providing his implementation of Gaussian process occupancy mapping [1], so it could be compared to DCT maps in the corresponding paper [2], Michael Veeck for providing his line extraction implementation [3] for evaluation in our article on polyline extraction [4], and Arash Ushani for his assistance with the NCLT dataset [5], featured in our work on pole-based localization [6].

Contents

I	Introduction	1
II	Discussion of the Individual Contributions	9
1	An Analytical Lidar Sensor Model Based on Ray Path Information	13
1.1	Research Context	13
1.2	Contribution	18
1.3	Critical Discussion	18
2	Closed-Form Full Map Posteriors for Robot Localization with Lidar Sensors	23
2.1	Research Context	23
2.2	Contribution	25
2.3	Critical Discussion	25
3	DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform	27
3.1	Research Context	27
3.2	Contribution	29
3.3	Critical Discussion	30
4	A Maximum-Likelihood Approach to Extract Polylines from 2-D Laser Range Scans	33
4.1	Research Context	34
4.2	Contribution	35
4.3	Critical Discussion	36
5	A Maximum-Likelihood Approach to Extract Finite Planes from 3-D Laser Scans	37
5.1	Research Context	37

5.2	Contribution	39
5.3	Critical Discussion	39
6	Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans	43
6.1	Research Context	43
6.2	Contribution	47
6.3	Critical Discussion	48
III	Publications	53
7	An Analytical Lidar Sensor Model Based on Ray Path Information	57
7.1	Abstract	57
7.2	Introduction	58
7.3	Related Work	61
7.3.1	Map Representations	61
7.3.2	Sensor Models	62
7.4	Approach	63
7.4.1	The Basic Idea of the Decay-Rate Model	63
7.4.2	Mapping	65
7.4.3	Localization	65
7.4.4	Integrating Out-of-Range Measurements	66
7.5	Mathematical Details	67
7.5.1	Decay-Rate Maps Maximize the Data Likelihood	67
7.5.2	The Decay-Rate Model Generalizes the Reflection Model	68
7.6	Experiments	70
7.6.1	Monte-Carlo Localization	70
7.6.2	Evaluation of the Pose Likelihood	73
7.6.3	Discussion of Results	75
7.7	Conclusion and Future Work	76
8	Closed-Form Full Map Posteriors for Robot Localization with Lidar Sensors	77
8.1	Abstract	77
8.2	Introduction	78
8.3	Related Work	79
8.4	Approach	80
8.4.1	Factorizing Forward Sensor Models	82
8.4.2	Recursive Map Update	82

8.4.3	Closed-Form Map Posteriors	84
8.4.4	Localization with Map Posteriors	86
8.4.5	Closed-Form Measurement Likelihoods	86
8.5	Experiments	88
8.5.1	Localization in Simulation	89
8.5.2	Real-World Localization	90
8.6	Conclusion and Future Work	93
9	DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform	97
9.1	Abstract	97
9.2	Introduction	98
9.3	Related Work	101
9.4	Approach	103
9.4.1	The Decay-Rate Model	103
9.4.2	Transforming the Spectral Map Representation to the Spatial Domain	105
9.4.3	Computing the Measurement Likelihood	106
9.4.4	Building the Decay-Rate Map	107
9.5	Experiments	109
9.5.1	Map Value Comparison	110
9.5.2	Measurement Probability Comparison	111
9.5.3	Execution Times	113
9.6	Conclusion and Future Work	117
10	A Maximum-Likelihood Method to Extract Polylines from 2-D Laser Range Scans	119
10.1	Abstract	119
10.2	Introduction	120
10.3	Related Work	120
10.4	Approach	123
10.4.1	Probabilistic Sensor Model	124
10.4.2	Polyline Extraction	125
10.4.3	Polyline Optimization	130
10.5	Experiments	130
10.6	Conclusion and Future Work	138
11	A Maximum-Likelihood Approach to Extract Finite Planes from 3-D Laser Scans	141
11.1	Abstract	141
11.2	Introduction	142

11.3	Related Work	142
11.4	Approach	145
11.4.1	Probabilistic Sensor Model	146
11.4.2	Maximum-Likelihood Estimation	147
11.4.3	Agglomerative Hierarchical Clustering	148
11.4.4	Probabilistic Plane Extraction	150
11.5	Experiments	152
11.6	Conclusion and Future Work	157
12	Long-Term Urban Vehicle Localization Using Pole Land-	
	marks Extracted from 3-D Lidar Scans	159
12.1	Abstract	159
12.2	Introduction	160
12.3	Related Work	162
12.4	Approach	163
12.4.1	Pole Extraction	164
12.4.2	Mapping	166
12.4.3	Localization	167
12.5	Experiments	168
12.5.1	Localization on the NCLT Dataset	168
12.5.2	Localization on the KITTI Dataset	173
12.6	Conclusion and Future Work	174
12.7	Acknowledgements	174
IV	Conclusion and Outlook	177
	Bibliography	181

Part I
Introduction

The present dissertation contributes to the area of lidar-based mapping and localization for mobile robots. It is structured as follows. This first part introduces the reader to the topic. The second part discusses the contributions at the base of this dissertation, before the third part replicates the corresponding publications. In the fourth part, we summarize the most important insights gained from this work.

In order to introduce the reader to the topic of this thesis, the next sections provide an intuitive understanding of the task of mapping and localization in the context of mobile robotics, then explain the working principle of lidar sensors, and finally explore alternative sensor types that can be used to support this task.

Mapping and Localization for Mobile Robots

Mobile robots are becoming a more and more important aspect of everyday life. Vacuum robots autonomously clean our homes while we are out, lawn mower robots silently take care of our greens, inspection robots watch for dangerous cracks in sewers, bridges, and wind turbine blades, and the prototypes of autonomous cars collect millions of test miles each year.

In order to perform any meaningful task, each of these mobile robots must be able to navigate its environment. Technically, navigation is the combination of three essential abilities that almost every mobile robot must possess: mapping, localization, and path planning. Mapping is the process of creating a model of the environment based on sensor readings. During localization, the robot makes use of this map and incoming sensor readings to estimate its pose. Path planning refers to the task of generating a trajectory to a given goal location based on the map and the location estimate. This dissertation focuses on the first two competencies, mapping and localization.

Of course, there are mobile robots that get along without mapping and localization software. The first generations of the iRobot Roomba vacuum cleaning robot for example, shown in figure 1, are not aware of their position within a room, but follow a set of simple reactive navigation rules, for example going in spirals, following walls, and “bouncing off” walls [7]. Such an approach, however, is limited to executing simple tasks, and would never solve the complex challenges a mobile robot like an autonomous car is facing. In general, more complicated applications require more powerful mapping and localization software. Tellingly, newer and improved Roomba generations feature an infrared sensor and mapping and localization capabilities. They build a floor plan-like map of the rooms they operate in, which allows them to cover the area to clean more systematically and hence faster,



Figure 1: First-generation iRobot Roomba of 2002. Source: eBay product picture, www.ebay.com, 2019.

to interrupt their work in order to drive back to the recharging station, and to continue cleaning at the same spot where they had stopped before.

Navigation Sensors for Mobile Robots

Over the past years, lidar has become an essential sensing modality for mobile robots navigating complex environments, like autonomous fork lift trucks, rescue and inspection drones, or autonomous cars. In order to provide three-dimensional geometric measurements of its environment, a lidar sensor, short for light detection and ranging, constantly emits laser pulses. If a surface in the environment of the robot reflects such a pulse back to the sensor, the device measures the time of flight of the photons and estimates the distance to the object via the known speed of light. Usually, the emitter constantly changes its orientation to cover a certain area of interest. At the same time, its angular encoders record the direction of the emitter for each light pulse. Based on the computed length and the recorded direction of the ray, the sensor then outputs the corresponding measurement in polar coordinates. If no reflection within the sensor range occurred, the sensor usually sets the ray length to undefined.

This principle of operation entails some important advantages in comparison to other types of depth sensors like stereo cameras or projection-based sensors. First of all, lidars are very precise. The accuracy of the returned data points usually lies within the millimeter range. Furthermore, due to the measurement principle of lidars, this accuracy is invariant under the dis-

tance between the sensor and the reflection. In contrast to stereo cameras, the measurements are readily available, too: no need to solve difficult data association problems and no need to triangulate. Moreover, since lidars are active sensors with a narrow bandwidth, illumination changes have hardly any effect on the quality of the measurements. Consequently, a lidar can operate both in bright sunlight and in complete darkness.

Despite the fact that these advantages make lidar a popular sensor modality in mobile robotics, the technology is still afflicted with several drawbacks. For example, even though modern high-performance lidars like the ones shown in figure 2 are able to produce millions of points per second, their output rate is still sparse compared to camera data. And although their active sensing technology makes them very robust to changes in illumination, direct sunlight and the emissions of neighboring lidars can produce false measurements. Furthermore, all of today’s commercially available high-performance lidar sensors rely on a spinning array of emitters and receivers, which comes at high cost, high weight, high maintenance effort, and high energy consumption.

Solid-state lidar devices are expected to alleviate the disadvantages of their electromechanical counterparts. The technology relies on a phased array of light emitters that are all part of the same silicon chip and together steer the direction of the outbound laser ray. But although solid-state lidars have been announced repeatedly during the past years, hardly any such product is commercially available to date. Given that today’s popularity of electromechanical lidar sensors in mobile robotics already justifies dedicated research, the advent of solid-state technology is expected to make this research even more relevant, as sensors would find their way into applications where the costs, the weight, and the energy consumption of electromechanical devices is prohibitive.

Of course, there exist plenty of other sensing modalities apart from lidar that, depending on the application, may be well suited for mobile robot navigation. Cameras, for example, provide rich visual information about their surroundings. Here, the term camera does not necessarily refer to the standard RGB sensor operating in the visible electromagnetic spectrum: Thermal cameras provide high-resolution images of the infrared spectrum, while hyperspectral cameras can measure radiation intensity in up to dozens of narrow electromagnetic bands, possibly far beyond the visible range. Recently, event cameras have proven to be powerful navigation sensors [8], too. Instead of capturing whole images at a fixed rate like conventional cameras, an event camera registers and reports brightness changes per pixel, in this way creating a stream of relative changes in the image. Since the temporal resolution of an event camera is orders of magnitude higher than that of a conventional



(a) Velodyne HDL-64E.



(b) Ouster OS2.



(c) Sick MRS6000.



(d) Quanergy M8.



(e) Luminar Iris.

Figure 2: Different state-of-the-art electromechanical high-performance lidar sensors that provide up to several million measurements per second. Sources: manufacturer websites, 2019: www.velodynelidar.com, www.ouster.com, www.sick.com, www.quanergy.com, www.luminartech.com.

camera, this type of sensor is well suited for high-speed applications, where conventional cameras would only provide a stream of blurred images.

The above-mentioned camera types provide a projection of the three-dimensional scene onto the image plane. Additional depth data, however, often simplifies the mapping and localization process and improves accuracy. Apart from stereo cameras, which estimate the depth of the scene via triangulation, there are two kinds of camera-based sensors which output 3-D data points: projection-based depth sensors and time-of-flight cameras. Projection-based depth sensors project a known infrared pattern onto the scene. By observing the distortion of the pattern, they estimate the depth. Time-of-flight cameras, on the other hand, work like lidars in the visible spectrum: For each pixel, they measure the time it takes the corresponding light ray to travel to a reflective surface and back. Based on this information, time-of-flight cameras estimate the distance of the corresponding object in the scene.

Cameras use only a fraction of the electromagnetic spectrum, more specifically wavelengths in the nanometer to micrometer range. Radio technology makes it possible to exploit the millimeter to centimeter range for robot navigation, too. Radars, for example, are popular sensors in the context of autonomous driving. Their working principle is similar to that of lidar sensors, but the waves they emit interact differently with their environment: Radars can see through many objects that reflect light, for example living organisms or walls.

Radio sensors do not necessarily have to be active, though. They can also receive the waves emitted by other sources without emitting signals on their own. By measuring the signal strength of nearby senders, every Wi-Fi card, Bluetooth receiver, and Near-Field Communication device can be turned into an inexpensive radio sensor suitable for robot navigation.

Not only electromagnetic waves lend themselves to sense the geometric properties of the environment, but also sound waves. Sonar systems have been successfully applied to mapping and localization since the early days of mobile robots [9]. Analogously to lidar and radar sensors, they emit sound waves and estimate the distance of nearby objects based on the waves' time of flight.

The listing above is by no means exhaustive. There are many other modalities that are well suited for robot navigation, for example force sensors or bumpers. However, the listing may be comprehensive enough to convey an intuition of the breadth of possible sensors and sensing modalities.

Part II

Discussion of the Individual Contributions

The following six chapters describe the individual contributions of this work to the topic of highly accurate lidar-based mapping and localization for mobile robots. Each chapter places the respective contribution into its research context, briefly describes the proposed approach, summarizes the experimental findings, and closes with a critical discussion of the contribution.

This part is intended to serve as an introduction to each developed method, and to provide additional information that is not part of the corresponding research article. It does not go into the theoretic and experimental details of each contribution. For these details, please see the original publications in part III.

Chapter 1

An Analytical Lidar Sensor Model Based on Ray Path Information

The contribution described in the following is a novel consistent probabilistic beam-based lidar sensor model, which we named the decay-rate model. This chapter starts by introducing the reader to sensor models in general, then details the specifics of the proposed approach, and finally discusses the advantages and drawbacks of the method presented.

1.1 Research Context

Before going into detail about the decay-rate model, we first need to clarify what a consistent probabilistic beam-based lidar sensor model is. A sensor model is a mathematical model required for mapping and localization. It describes the relation between the measurements returned by the sensor and the true state of the world. In the context of mapping and localization, the true state of the world is usually represented by two variables: the robot pose and the map. The robot pose can be a homogeneous 4×4 transformation matrix, for example, and the map might be a three-dimensional matrix whose every cell represents the value of a three-dimensional grid map.

Such a sensor model should not be formulated in a deterministic way: Measurement noise, model inaccuracies, and unmodeled effects like unforeseen dynamics in the scene necessitate a probabilistic approach [10]. Following this rationale, the decay-rate model represents the relation between lidar measurements, robot pose, and map in a probabilistic fashion. It assigns a probability to each combination of these variables.

That explains the meaning of a probabilistic sensor model, but what is a consistent sensor model? There are two sorts of sensor models: forward models and inverse models. Forward models describe the physical process of the sensor measuring the environment. They are given the causes, i.e. the robot pose and the map, and predict the probability of a certain effect, i.e. the probability of a set of lidar measurements. Thus, forward models can be used to localize the robot by finding the pose that maximizes the probability of a given measurement in a given map. Inverse sensor models work the opposite way. Given the effects, namely the sensor measurements, and the robot pose, an inverse model reasons about the causes, which in our case translate to the map. Inverse sensor models are hence a vital tool for mapping. A consistent sensor model is now defined as a sensor model that can be used both as forward and as inverse model.

“Beam-based” means that the model makes use of information about the trajectory of the laser beams emitted by the lidar sensor, as opposed to models that only consider the endpoints of the rays. In general, beam-based models come with greater computational complexity due to ray tracing, but they also achieve higher accuracy than endpoint-based models, because they leverage more of the information provided by the sensor.

To illustrate the differences between the types of sensor models discussed above, and in order to provide an overview over the most popular models, we will now contrast the likelihood field model with the reflection model, discuss their advantages and disadvantages, and point out how the decay-rate model provides an improved solution to the underlying problems.

The likelihood field model [10], illustrated in figure 1.1, is based on a heuristic, which means that it does not model a physical process. Instead, it approximates the probability of a lidar measurement by evaluating a zero-centered Gaussian probability distribution at the distance between the endpoint of a given lidar measurement and the nearest object in the map. Using a suitable tree-based data structure to store the objects, this approach can achieve logarithmic computational complexity in the number of mapped objects.

Since the likelihood field model obtains the measurement likelihood directly from the object map without the need for ray tracing, it is classed with correlation-based sensor models, as opposed to beam-based models. As such, it is computationally efficient, but it cannot avoid the disadvantages associated with all correlation-based models. First, it is not able to achieve the accuracy of beam-based models, because it discards valuable information returned by the sensor: As a consequence of not considering the ray path, the model overlooks the fact that rays cannot travel through objects like walls, and it is unable to assign probabilities to measurements that are

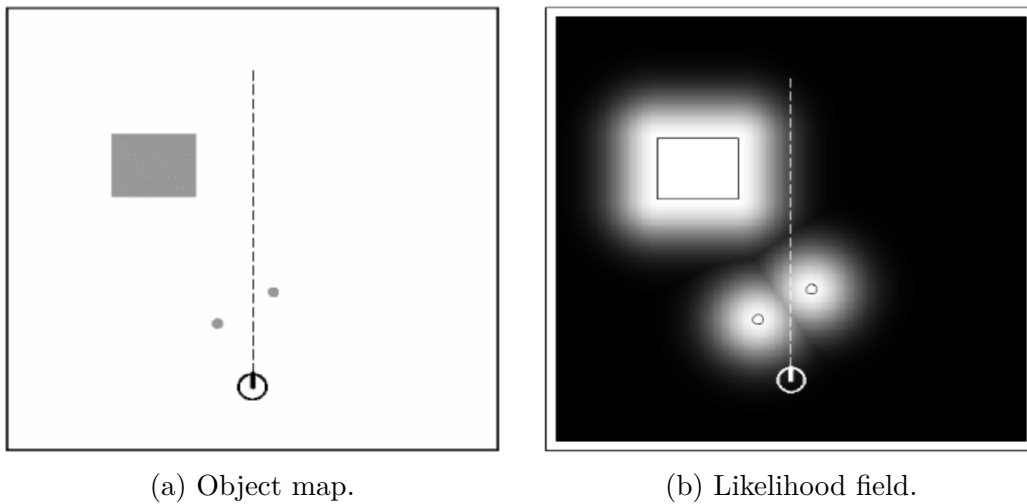


Figure 1.1: Object map and corresponding likelihood field. The circle represents the robot, the dashed line illustrates an exemplary laser ray, and the gray areas in the object map on the left-hand side represent objects. The brightness of the likelihood field on the right encodes the probability of measuring a reflection: black means low probability, white stands for high probability. Source: website for “Probabilistic Robotics” by Thrun et al. [10], <http://probabilistic-robotics.informatik.uni-freiburg.de>, 2019.

not reflected. Second, the likelihood field model is inconsistent, because it assigns probabilities to sensor measurements given a map, but it offers no mechanism to derive an object map from lidar data.

For complex problem settings, the likelihood field model is often overly simplistic. In these cases, the more realistic, beam-based and consistent reflection model [11], also known as hit-counting model, might be more adequate. It represents the robot's environment as a grid composed of homogeneous cells. Every grid cell encodes the probability that an incident ray is reflected. During mapping, this reflection probability is computed as the quotient of all reflections within the cell and the total number of rays that enter the cell. Both the numbers of reflections and the number of rays that enter the cell are determined via ray tracing. Ray tracing also plays an important role in the forward pass, because in order to assign a probability to a measurement given a map, the cells visited by the corresponding ray need to be identified. After this has been done, the measurement probability is calculated as the probability of the ray penetrating all but the last cell, multiplied by the probability of the ray being reflected in the last cell.

As a beam-based model, the reflection model makes use of more sensor information than the likelihood field model. Tracing the individual rays through the grid ensures that obstacles in the ray path have an impact on the probability of reflections behind these objects. However, the reflection model still discards some information. For an illustrative example, see figure 1.2. It shows two cells that are assigned the same reflection probability, because the numbers of reflected and transmitted rays are equal. The distances which the individual rays travel within the cells, however, vary greatly from left to right. On the left-hand side, the rays cover large distances, while on the right-hand side, they barely touch the cell. Intuitively, the longer the distance a ray travels within a cell, the more information it collects about the interior of the cell. In discarding the ray lengths per cell, the reflection model does not account for this information.

This is not the only drawback of the reflection model, though. Since the tessellation of space is an integral part of its definition, it is incompatible with continuous maps. Furthermore, the reflection probability only changes at the cell boundaries, as shown in figure 1.3a. This behavior results in a zero derivative, which does not allow the application of optimization techniques to mapping and localization.

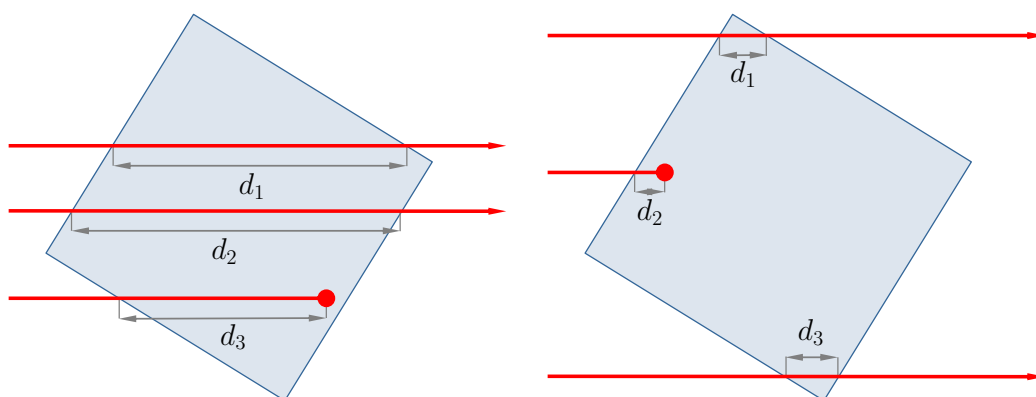
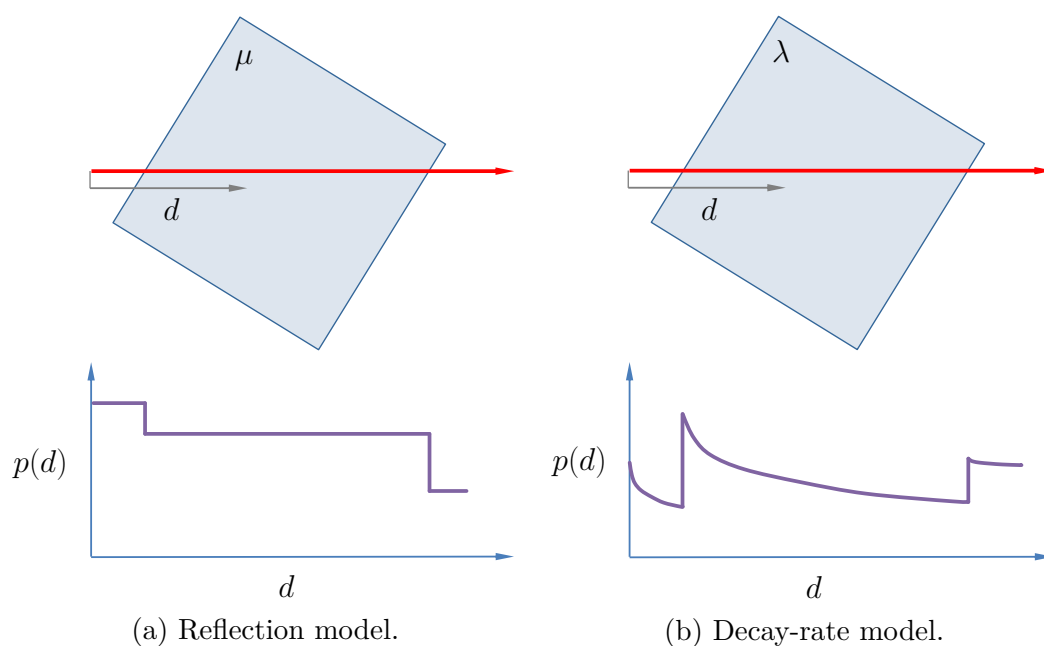


Figure 1.2: Two cells with the same numbers of lidar reflections and transmissions, but with different distances d_i that the rays travel inside the cells. The blue boxes represent the cells, the red arrows represent the lidar rays.



(a) Reflection model.

(b) Decay-rate model.

Figure 1.3: Measurement probability density functions of different sensor models. The blue boxes represent the cell, the red arrows represent the lidar rays. The variable μ means reflection probability of the cell, λ is the decay rate of the cell, d stands for the distance that the ray travels inside the cell, and $p(z)$ is the probability that the ray is reflected. While the measurement probability of the reflection model only changes at the cell boundaries, the measurement probability returned by the decay-rate model changes within the cells, resulting in a non-zero gradient.

1.2 Contribution

The decay-rate model improves on the reflection model and mitigates its afore-mentioned disadvantages. The key idea behind it is modeling the interaction between the laser ray and the environment as an exponential decay process, hence the name. To understand this approach intuitively, one can imagine the space to be filled with tiny particles, whose concentration may vary locally. The higher this concentration, the higher the probability that an incident laser ray is reflected. The particle concentration can be expressed as a decay rate that governs how far, on average, a ray emitted by the sensor travels before it is reflected. Consequently, instead of modeling the distribution of the reflection rate, as is the case for the maps produced by the reflection model, the decay-rate model produces decay-rate maps, which tell for each location the corresponding decay rate.

We do not repeat the detailed mathematical derivation of the model that can be found in our article introducing decay-rate maps in part III, chapter 7. But we want to point out that the decay-rate model results in mapping and localization algorithms that are similar to those of the reflection model, and that do not increase computational complexity. In the context of grid-based mapping, for example, the decay rate of a cell is computed as the quotient of the number of reflections in the cell and the total distance that all rays travel within the cell, as opposed to the number of reflections and the number of rays that enter the cell.

1.3 Critical Discussion

By modeling the interaction between the lidar ray and the world as an exponential decay process, the decay-rate model generalizes and improves the reflection model. During mapping, it measures the lengths that the individual rays travel within the cells and incorporates this information in the map. Consequently, the map is a more accurate representation of the surroundings of the robot. During localization, the measurement probability is no longer constant within a cell, as is the case for the reflection model, but it decreases over the distance the ray travels within the cell. The result is a piecewise continuously differentiable measurement probability as shown in figure 1.3b, which enables optimization-based localization methods. Furthermore, accounting for the ray lengths per cell during both mapping and localization results in higher localization accuracy, as demonstrated in the experiments described our paper in part III. In terms of computational effort, these improvements come for free, because the decay-rate model has the

same computational complexity as the reflection model. This is due to the fact that the ray lengths per cell must be computed during ray tracing, no matter whether they are made use of or not.

As mentioned above, the reflection model always requires tessellation of the space. It cannot be applied to a continuous space, because by definition, all changes in measurement probability occur at the cell boundaries. Without cells, however, there are no cell boundaries. The decay-rate model, on the other hand, does not depend on the discretization of space, although our article derives the mapping and localization algorithms for grid-based maps. In fact, to the best of our knowledge, the decay-rate model is the first lidar sensor model that allows true continuous mapping and localization without intermediate rasterization. In chapter 3, we make use of this fact, and we present a corresponding mapping and localization framework based on continuous maps.

The decay-rate model eliminates many problems of the reflection model, but it is still afflicted with the general drawbacks of beam-based lidar models. For example, despite the fact that it does not introduce any additional computational complexity compared to the reflection model, it is significantly more computation-intensive than the likelihood field model, for example. The reason for this behavior lies in the ray tracing process, which can hardly be simplified.

Like the reflection model, the decay-rate model also suffers from model simplifications that may negatively affect mapping and localization accuracy. One of these simplifications is the assumption that the laser ray is a straight line without volume. However, figure 1.4 illustrates that in reality, every laser ray has a finite cross sectional area. Since we always have to deal with beam divergence, too, the geometric shape of a ray resembles a frustum rather than a line. This situation introduces additional uncertainties to the measurement process. If a ray travels close to the border of an object, for example, its optical axis might not hit the object, but a significant portion of the ray's cross sectional area might, which can lead to measurements that contradict the simplified ray-line model.

As another conceptual simplification, neither the reflection model nor the decay-rate model account for uncertainty in the sensor measurements. And yet, the sensor pose, the laser ray length, and the laser ray direction are affected by noise. Similarly, the map does not encode any confidence information. Both models estimate maximum-likelihood maps during mapping, which they interpret as true representations of the environment during localization.

The next work underlying this thesis focuses on the latter issue. It develops a framework that allows to capture the map uncertainty and in this way

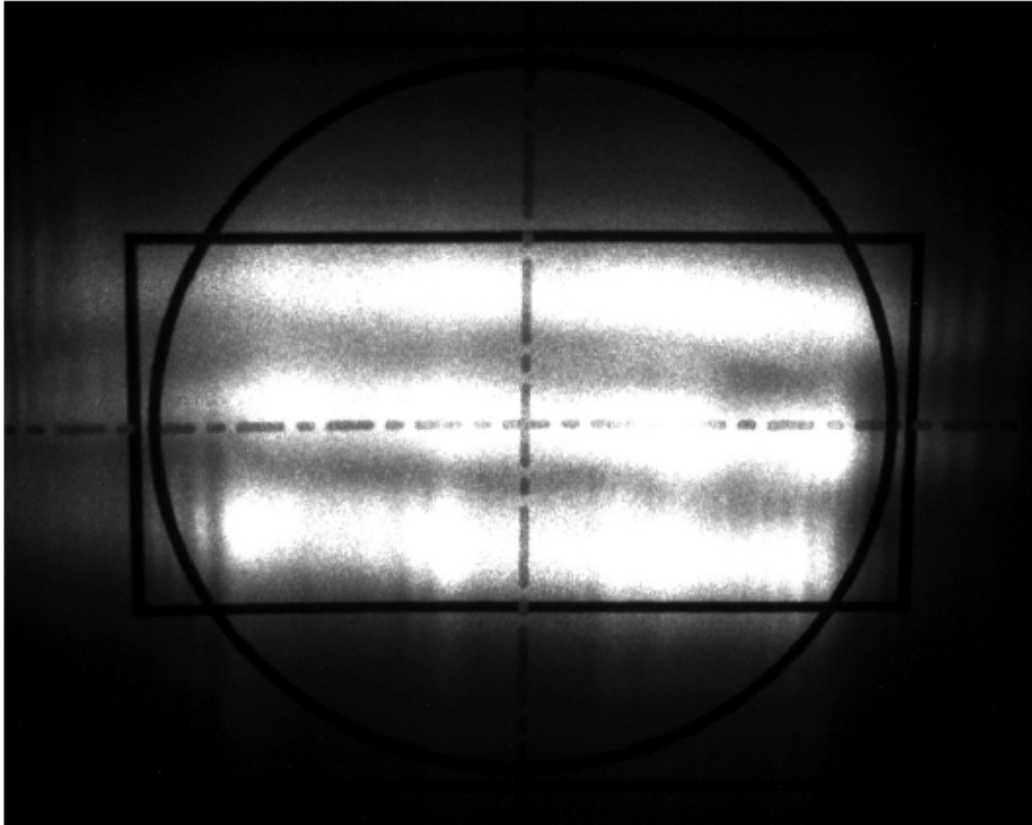


Figure 1.4: Cross section of a laser ray emitted by a Velodyne VLP-32C, photographed at the sensor's ring lens. The spot is 9.5 mm tall by 12.7 mm wide. While shooting off the sensor, the spot size increases according to the beam divergence. For this sensor, the horizontal beam divergence is 3.0 mrad, while the vertical beam divergence amounts to 1.5 mrad. Source: VLP-32C User Manual, 63-9325, Revision B, Velodyne LiDAR Inc., 2018.

improves both mapping and localization accuracy.

Chapter 2

Closed-Form Full Map Posteriors for Robot Localization with Lidar Sensors

As mentioned in the previous chapter, lidar-based grid maps often just represent the most probable map values. This representation ignores the fact that for each cell, all map values are possible, and that every map value has a certain probability attached to it. The present work addresses this issue.

2.1 Research Context

The individual cells of grid-based reflection maps and decay-rate maps contain maximum-likelihood values. This is due to the fact that the derivation of the inverse model is based on maximum-likelihood estimation, which for each cell seeks to find the map value that maximizes the probability of all given measurements. Consequently, each cell of the resulting map tells only the mode of the underlying probability distribution, not the distribution itself, as illustrated in figure 2.1. That means that downstream modules obtain no information about the confidence of the map estimate, a situation that leads to decreased localization accuracy and that renders a probabilistic correlation of multiple maps impossible. These disadvantages motivate the present contribution. The proposed approach does not represent each map cell by its most likely value, but instead describes the posterior probability distribution over all possible map values in closed form via a parametric distribution.

Occupancy grid maps, proposed by Moravec [12] and Elfes [13], serve as a kind of prototype of our full posterior map representation. Their world model relies on the assumption that space is binary: Each point in space can either

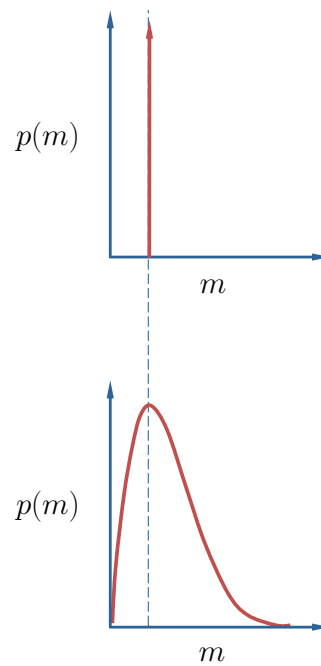


Figure 2.1: Comparison of the probability density functions corresponding to the maximum-likelihood representation (upper graph) and the full-posterior representation (lower graph) of the same distribution. While the full-posterior representation captures the original distribution, the density function of the maximum-likelihood representation is a Dirac delta function at the mode of the full-posterior distribution, as indicated by the red arrow.

be free or occupied. Based on this assumption, they tessellate the space to form a uniform grid and assign an occupancy probability between 0 and 1 to each cell. Consequently, each map cell holds a posterior distribution over a binary random variable. This parametric distribution in one parameter is also known as Bernoulli distribution. Since the map cells are assumed to be independent, the occupancy map represents a probability distribution over all possible occupancy maps. The corresponding probability density function results from computing the joint probability of all individual cells.

2.2 Contribution

The contribution of this part of the dissertation is a representation of the full posterior of reflection maps and decay-rate maps that is equivalent to the representation of occupancy grid maps. That means that instead of storing the most probable map value per cell, this work shows how to assign a parametric distribution to each cell that exactly, without approximations, represents the full posterior probability distribution over all possible map values. It also shows that this representation can not only be applied to the reflection model and the decay-rate model, but in fact to all factorizing sensor models. In this context, we define a factorizing sensor model as a beam-based sensor model that computes the measurement probability as a product over functions of the map values of the cells traversed by the corresponding ray. For the detailed mathematical derivation, please see the original article in chapter 8.

Table 2.1 illustrates the most important findings of the present work by taking the example of the reflection model: Transitioning from the maximum-likelihood representation of a map to the full posterior distribution requires only a slight change in formulas. The effect on the resulting localization accuracy, however, is significant, as shown in the experiments in section 8.5. Fortunately, this transition does not affect the computational complexity of the forward or inverse pass, because the full-posterior formulation takes the same parameters as the maximum-likelihood representation as input and does not introduce any additional computations.

2.3 Critical Discussion

Maintaining the full posteriors of the map cells instead of their maximum-likelihood values entails a number of advantages. For example, full posteriors enable probabilistic map correlations. But most importantly, knowing the

Model	Maximum likelihood	Full posterior
Inverse	$\mu^* = \frac{n_h}{n_h + n_m}$	$p(\mu Z) \propto \text{Beta}(\mu; n_h + \alpha, n_m + \beta)$
Forward	$p(R \mu^*) = \mu^* = \frac{n_h}{n_h + n_m}$	$p(R Z) = \frac{n_h + \alpha}{n_h + \alpha + n_m + \beta}$

μ	reflection rate
μ^*	most likely reflection rate
n_h	number of reflections in the cell
n_m	number of rays traversing the cell without reflection
Z	set of sensor measurements
α, β	parameters of the prior beta or gamma distribution
R	event of a reflection

Table 2.1: Comparison of the maximum-likelihood formulation and the full-posterior formulation of the forward and inverse reflection model. As the table shows, transitioning from one to the other only requires a slight change in formulas.

confidence of the map significantly improves localization performance. This is not only true in theory: The extensive experiments performed in the scope of this work in section 8.5 prove that both in simulation and in the real world, full-posterior mapping results in a higher measurement probability assigned to the ground-truth pose of the robot and to increased localization accuracy compared to the maximum-likelihood approach. These advantages come for free: Full-posterior maps have the same computational and memory complexity as maximum-likelihood maps. Since transitioning from maximum likelihood to full posterior only requires small changes to the mapping and localization algorithms, as demonstrated in table 2.1, there is no reason to use the maximum-likelihood representation any longer.

Chapter 3

DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform

As explained in chapter 1, to the best of our knowledge, the decay-rate model is the first lidar sensor model that genuinely supports continuous mapping. The work discussed in the following builds upon that. It proposes an approach that makes use of the cosine transform to create continuous decay-rate maps, and it derives the corresponding closed-form measurement probability computation algorithm.

3.1 Research Context

The most popular lidar sensor models like occupancy grid mapping or the reflection model rely on the assumption that the space surrounding the robot is tessellated. Tessellation, however, entails important disadvantages. For example, grid maps are not differentiable and can thus not be used for optimization-based localization techniques. In addition, grid maps exhibit aliasing effects. As illustrated in figure 9.1b on page 99, staircase patterns emerge whenever the structures of the world are not aligned with the map, or when the map values exhibit smooth transitions. In order to avoid these disadvantages, more flexible, continuous map representations are required.

Continuous lidar maps are still an active field of research. In order to explore the state of the art, in the following, we will investigate three popular continuous map representation, namely the normal distributions transform [14], Gaussian process occupancy maps [15], and Hilbert maps [16].

The normal distributions transform (NDT) is a technique primarily de-

veloped for scan matching. It creates a continuous map by first tessellating the space around the robot. Next, it assigns to each grid cell a Gaussian distribution that represents the distribution of the laser endpoints inside the cell. This Gaussian is cropped at the boundaries of the respective cell, so that there is no overlap between neighboring distributions. The result is a piecewise continuously differentiable map that enables the computation of spatial gradients. In the forward pass, the model computes the probability of measuring an endpoint at the query position by evaluating the corresponding Gaussian at that point. In order to use the model for scan matching, as intended, one creates an NDT map from one scan and then changes the relative pose of the other scan so as to maximize the sum of the measurement probabilities assigned to the scan endpoints.

Although NDT has been successfully applied to scan matching problems, there are a couple of reasons why it is not suitable for lidar-based mapping and localization. First and foremost, NDT is a heuristic that does not produce true measurement probabilities. This is due to the fact that it sums the probabilities of the individual rays, rather than computing the true joint probability. Second, NDT it is a correlation-based model that does not account for the ray paths. As such, it is regarded as a method to smooth a point cloud rather than as a genuine sensor model.

Gaussian process occupancy maps (GPOMs) are a continuous extension of occupancy grid maps. As the name suggests, they model the posterior of the occupancy value, which is equivalent to the reflection value in the context of the reflection model, as a Gaussian process [17]. During mapping, in the inverse pass, they employ supervised machine learning: They train a Gaussian process on laser scan endpoints and on free points sampled along the rays to produce a continuous occupancy map. This approach achieves a number of advantages compared to occupancy grid maps or reflection maps. For example, the produced maps are continuously differentiable. Moreover, GPOMs overcome the independence assumption between neighboring cells, an assumption that enables conventional occupancy grid mapping and reflection mapping in the first place. Another important improvement is the fact that GPOM are able to propagate sensor pose uncertainty and measurement noise into the map.

However, GPOMs suffer from an essential limitation: They are built upon a model that is intrinsically incompatible with continuous mapping and localization. Despite the fact that it is possible to build continuous occupancy maps, i.e. to perform the inverse pass, it is impossible to compute probabilities based on continuous occupancy maps, i.e. to perform the forward pass, because as explained in section 1.1, the forward model presupposes the tessellation of space. The measurement probability only changes at the borders

of the cells, but if there are no cells, there is no way to compute this probability. Consequently, in order to compute measurement probabilities, GPOMs would have to be tessellated again.

Another problem with GPOMs is the fact that the underlying sensor model is not a true beam-based model. In order to account for free space, free-space points have to be sampled along the rays. This process makes it impossible to incorporate the complete ray path information, because it comprises infinitely many points. Furthermore, GPOMs do not formulate a recursive map update. That entails high computational costs, especially because map building is a large-scale and thus expensive optimization problem.

Hilbert maps are similar to Gaussian process occupancy maps in the sense that they, too, use supervised machine learning to create a continuous occupancy map of the robot’s environment. Instead of relying on Gaussian regression, however, they learn a logistic regression classifier in a Hilbert space. This classifier is then used to discern between occupied and free regions in the space surrounding the robot.

Like GPOMs, Hilbert maps overcome tessellation and the independence assumption between cells, and they handle outliers in a robust manner. But they suffer from the same disadvantages as GPOMs: They do not formulate a forward model, the underlying sensor model is not beam-based, the maps cannot be recursively updated, and Hilbert maps come at high computational cost, since map building is a large-scale optimization problem.

To sum up, the presented state-of-the-art continuous mapping approaches all come with substantial problems. NDT is a heuristic that is not suitable for probabilistic mapping and localization, while GPOMs and Hilbert maps are based on a sensor model that is in its core incompatible with localization based on continuous maps. The contribution of the present work, DCT maps, aims at overcoming these disadvantages by combining the decay-rate model with the continuous extension of the discrete cosine transform.

3.2 Contribution

The original research article on DCT maps, replicated in chapter 9, presents a consistent beam-based lidar sensor model based on continuous maps. It extends the decay-rate model, presented in chapter 7. In contrast to GPOMs and Hilbert maps, it cannot only produce continuous maps, but also assign measurement probabilities to sensor readings based on a continuous map – in short, it formulates a forward model. Furthermore, it does not sample free points along the laser rays, but it leverages the full ray path information.

Just like grid-based decay-rate maps, DCT maps represent the spatial

distribution of the lidar decay rate. But while grid-based maps store the map parameters in the spatial domain, DCT maps store them in the frequency domain. Both types of maps are usually stored in memory as matrices. In the case of grid maps, the elements of the matrix directly represent the decay rates at the corresponding locations. In the case of DCT maps, however, the elements of the matrix determine the amplitudes of cosine waves of different directions and frequencies. To render a DCT map based on this indirect representation, or to look up the value of the DCT map at a given position, one superimposes all cosine waves.

The technique we use in the present contribution to transition from the frequency-based map representation to the spatial domain is called CEIDCT: continuous extension of the inverse discrete cosine transform. The CEIDCT is a Fourier-related transform based on cosine waves, which transforms a discrete signal in the frequency domain to a continuous signal in the spatial domain.

There are multiple reasons for choosing the CEIDCT over the many other Fourier-related transforms. First, it is mathematically proven that with the CEIDCT, the reconstruction of a signal from the frequency domain to the original domain converges to the original signal for an increasing number of parameters [18]. Although it might be desired in most cases, many other Fourier-related transforms do not exhibit this behavior: They diverge. Second, the CEIDCT parameters are real, not complex values, which saves memory.

Based on the CEIDCT, the DCT maps framework formulates a consistent forward and inverse sensor model. The forward model determines the measurement probability of a given ray by iterating over all map parameters, implicitly performing ray tracing. During mapping, the inverse model follows a maximum-likelihood estimation approach and finds the map that explains the collected sensor measurements best. To that end, it maximizes the joint probability of all measurements over the spectral map parameters in a single nonlinear multivariate optimization problem. In order to compute the derivatives of the measurement likelihood with respect to the map parameters, it makes use of analytical derivatives. For a detailed derivation of the DCT maps framework, see chapter 9.

3.3 Critical Discussion

Analogously to the last chapters, we will highlight how the suggested approach advances the state of the art, but we will also discuss the problems associated with it. Since this work opens up a new field of continuous map-

ping approaches, we will furthermore provide an overview over possible future research directions at the end of this section.

DCT maps improve on grid-based decay-rate maps in several respects. For example, they enable robot localization and scan matching using gradient-based optimization techniques. Furthermore, our experimental evaluation in section 9.5 demonstrates that in comparison to grid maps with the same memory footprint, DCT maps reconstruct a given ground-truth map with higher accuracy, and they assign higher probabilities to given measurements. We attribute these effects, which lead to a higher overall localization accuracy, to the fact that DCT maps are able to more precisely reconstruct contours that are not aligned with the main axes of the map: In contrast to grid maps, DCT maps do not exhibit staircase patterns, independent of the alignment of the contours in the map.

The memory efficiency of DCT maps is superior not only to grid maps, but also to other continuous map representations. Given the same amount of memory, they outperform GPOMs and Hilbert maps in terms of map reconstruction quality and in terms of the probability assigned to a ground-truth measurement. In addition, DCT maps are the only continuous maps to incorporate the full ray path information.

The substantial improvement with respect to GPOMs and Hilbert maps, however, lies in the fact that DCT maps formulate a forward model. GPOMs and Hilbert do not support the forward pass, because they are based on occupancy grid mapping, which requires the space to be tessellated. The decay-rate model which DCT maps are based on, however, naturally supports continuous maps. As such, it is the appropriate foundation for any lidar-based continuous mapping approach.

While innovative, DCT maps do have their limitations. One of their major problems is the computational effort required during mapping. Solving a nonlinear optimization problem in the number of map parameters may not be an issue for small maps, but since the number of map parameters grows quadratically or cubically with the edge length of 2-D or 3-D maps, respectively, DCT maps quickly reach their limits with increasing map size.

Furthermore, like GPOMs and Hilbert maps, they do not formulate an incremental map update. That means that in order to incorporate new measurements into the map, the optimization problem must be solved again. Of course, using the previous map as initial guess reduces the computational effort, but it is still significantly larger than for grid maps, for example.

Another disadvantage with respect to grid maps is the fact that looking up the value of the map at a single location requires iterating over all map parameters. This issue is a direct consequence of storing the map parameters in the frequency domain, not in the spatial domain.

GPOMs, Hilbert maps, and DCT maps all have the disadvantage that they do not distinguish between explored and unexplored map areas. Areas far away from explored regions are assigned random values, which may change from optimization run to optimization run.

And of course, DCT maps are maximum-likelihood maps. As such, they do not contain any information about the confidence of the respective map parameters.

DCT maps open the field for a range of possible extensions. Hybrid grid-based and continuous approaches could remedy the problems of high computational effort and limited scalability, for example. Such a hybrid map could consist of a grid-based tree data structure like a quadtree or an octree, whose every cell holds a DCT map. This representation would simplify the map update, and by efficiently representing unexplored regions, it would keep memory requirements low.

Another possible solution to deal with memory limitations is the investigation of compression techniques. Frequency-based data representation lend themselves to compression, for example by discarding high-frequency or low-amplitude parameters.

DCT maps open the way for concurrent mapping and localization, too. To that end, the optimization problem described in the present work would need to be extended to comprise the robot trajectory.

At last, another interesting direction of research would be to compare DCT maps with continuous decay-rate maps based on Gaussian processes and Hilbert regression.

Chapter 4

A Maximum-Likelihood Approach to Extract Polylines from 2-D Laser Range Scans

After having treated grid-based and continuous maps in the previous chapters, we will now focus on feature-based maps. While the former are dense representations that assign a map value to each location in space, the latter take a different approach: They proceed on the assumption that in general, the space around the robot is empty, and store a set of objects that populate the space. A feature map can be as simple as a set of coordinates of undistinguishable landmarks, and as complex as a set composed of features of different classes, represented by feature vectors of different lengths.

There are multiple reasons why feature-based maps are particularly interesting for mapping and localization for mobile robots. First, feature-based maps are very memory-efficient. Take the example of a robot equipped with a 2-D or 3-D lidar navigating an office environment that is composed of planar walls. A naïve approach to mapping would be to agglomerate all measured points, in this way forming a large point cloud. This representation, however, would introduce a high degree of redundancy, since endpoints are highly correlated. Creating a map of line features, on the other hand, would regress the highly redundant information to the most compact representation possible. But not only would this representation save memory: The regressed line segments would also reproduce the real walls more closely, since they average out sensor noise.

The second reason for choosing feature-based maps over dense maps is the fact that they can encode semantics. Semantics are important for many modules downstream the mapping module. The path planner, for example, needs to know whether the object in front of the robot is a solid wall or a

permeable cloud of smoke. A feature map encodes semantics whenever it is created using a semantic feature detector.

Third, feature maps abstract from sensor modality. That means that feature maps can be generated based on the input of any suitable sensor or combination of sensors. In general, that is not true for dense maps: One would have a hard time integrating radar measurement into a decay-rate map, because the map is specifically designed to model the optical properties of the space around the robot, which a radar cannot measure directly.

Fourth, feature-based maps improve map robustness. Imagine two occupancy grid maps of a scene in a park environment created from lidar data, one generated in summer, one in winter. The changes in foliage density alone would lead to significantly different looks of the two maps. Now imagine a map of the same scene based on tree features. Instead of integrating probably irrelevant information about tree foliage, it would encode the coordinates of the tree trunks, thereby rendering the map invariant under seasonal changes.

In this and the following chapters, we will investigate how to extract different kinds of features from lidar data. We will start with the extraction of polyline features from planar lidar scans. As mentioned above, this task is highly relevant for mobile robots navigating man-made environments like offices and warehouses.

4.1 Research Context

Man-made environments are often composed of linear structures and hence well suited to be represented by polyline maps. In order to extract polylines from laser scans, many authors resort to heuristic approaches originally developed in the context of cartography [19], like Visvalingam’s algorithm [20], iterative endpoint fit [21], or split-and-merge [22].

Visvalingam’s algorithm is a greedy top-down line simplification method that starts with the original, fine-grained polyline resulting from connecting all neighboring endpoints of a single planar laser scan. It then iteratively discards the vertex whose removal leads to the least perceptible change in the appearance of the polyline. This change is usually computed as the area of the triangle formed by the point that is to be removed and its two neighbors.

Iterative endpoint fit differs from Visvalingam’s algorithm in that it works bottom up, not top down. It connects the first and the last point of the original polyline and then iteratively inserts the vertex with the greatest normal distance from the simplified polyline.

Split-and-merge is based on iterative endpoint fit, but in addition, it

moves the vertices in order to minimize the squared normal distance between the vertices and the line.

Due to their simple concepts, Visvalingam’s algorithm, iterative endpoint fit, and split-and-merge appeal not only cartographers, but also to roboticists. However, they all suffer from an important disadvantage: They are heuristics. Consequently, they do neither consider information about the paths of the laser rays, nor about the measurement noise of the sensor. This simplification leads to a loss in accuracy of the extracted polylines.

There are only few probabilistic approaches that try to cure this problem, for example the one developed by Veeck and Burgard [3]. This method generates initial line estimates on the basis of an occupancy grid map, minimizes the normal distances between the scan endpoints and the nearest line, before repeatedly applying a set of eight operations like merging and splitting lines, adding and removing vertices, etc. In contrast to the afore-mentioned heuristic concepts, this method is rather complex. Not only does it require to build an occupancy grid map, its performance is also highly dependent on the many tuning parameters.

4.2 Contribution

In this work, we suggest a novel probabilistic approach to extract polylines from a single planar lidar scan. To the best of our knowledge, it is the only method that considers ray path information and sensor noise, more specifically Gaussian distributed radial noise. The algorithm consists of two steps, both of which strive to maximize measurement probability: polyline extraction and polyline optimization.

During polyline extraction, neighboring endpoints are connected to form a polyline. Then, the algorithm iteratively removes the polyline vertex that decreases the measurement probability of the overall scan the least until a stopping criterion is met.

The second step, polyline optimization, takes account of the fact that the vertices of the true polyline, which correspond to corners formed by two walls, will never exactly coincide with a laser endpoint. So in order to find the true polyline vertices, the vertices of the simplified polyline have to be moved. Assuming that moving the vertices to their true positions increases the joint probability of all measurements, we formulate and solve an optimization problem that maximizes the measurement probability of the scan over the vertex locations. For a detailed description of the method, please see the original paper in chapter 10.

In this part of the present dissertation, we do not only propose a novel

polyline extraction algorithm, but we also present extensive real-world experiments that show the superior performance of our method compared to conventional approaches under different metrics. In addition, we provide a free public open-source Matlab implementation of the method.

4.3 Critical Discussion

Despite the fact that extracting polylines from planar laser scans has been studied intensively, the suggested method still outperforms all of the surveyed state-of-the-art algorithms in terms of reconstruction accuracy, as demonstrated in our experiment series. This is due to the fact that in contrast to previous approaches, the novel method, termed PLE for probabilistic line extraction, takes into account ray path information and radial sensor noise.

To the best of our knowledge, PLE is not only the most accurate, but also the fastest polyline extraction algorithm to date, because its asymptotic computational complexity amounts to $\mathcal{O}(k \log(n))$, where n is the number of scan endpoints and k is the number of removed vertices.

Despite its merits, the approach still leaves room for improvements, most of which also apply to the surveyed related methods. For example, like every greedy algorithm, PLE is not guaranteed to find the optimal solution. Moreover, like all polyline extraction algorithms it was compared against, it does not model map uncertainty, i.e. it does not assign a confidence score to the individual lines.

Chapter 5

A Maximum-Likelihood Approach to Extract Finite Planes from 3-D Laser Scans

This part of the present thesis discusses an extension of the polyline extraction method presented in the previous chapter. Instead of extracting polylines from 2-D lidar scans, we are now interested in detecting planes in 3-D scans.

5.1 Research Context

Since plane extraction is an important task in both robotics and computer vision, this problem has been studied intensively in the past. The corresponding approaches can be divided into four classes: methods based on region growing, on clustering, on random sample consensus, and on the Hough transform.

Region growing algorithms need to be given seed points, building upon which they grow a plane by iteratively adding adjacent points. Adjacent points are only added to the plane if they fulfill certain planarity criteria, for example if their normal distance from the plane does not exceed a defined threshold.

Clustering, on the other hand, is an unsupervised machine learning technique that detects planes without requiring seed points. However, clustering techniques often require the number of planes to detect as input.

Random sample consensus, in contrast, selects multiple points at random, fits a plane to them, decides which other points belong to this plane based on a given distance threshold, and returns the planes that contain most of

the points of the lidar scan.

The Hough transform solves the problem of detecting planes by rasterizing the solution space, which is the space of parameters of all feasible planes. Using a heuristic metric that assigns a score to every point in the solution space, it identifies the most plausible plane hypotheses. While this works well for infinite planes, it is hard to employ the Hough transform to detect finite planes. This is due to the fact that the definition of finite planes requires a large number of parameters, which translates to high computational complexity of the detection algorithm. To overcome this problem, the Hough transform is often used to detect infinite planes, whose boundaries are then identified via region growing.

Before we started working on the present scientific contribution, we identified two major problems with all state-of-the-art approaches, no matter which class of approach they belonged to. The first problem concerns the fact that most plane-extraction techniques lack a probabilistic underpinning. Instead of employing a probabilistic sensor model, they rely on heuristics like orthogonal point-to-plane distance. In fact, while surveying the state-of-the-art literature, we were not able to find a single approach that considers the ray path information in the process of detecting planes. This is a problem because discarding this information necessarily diminishes plane detection accuracy.

The second problem relates to the way plane extraction algorithms are currently benchmarked. The most popular dataset for benchmarking plane extraction algorithms is SegComp, created in 1996. It contains lidar measurements of a tabletop scene featuring polyhedral objects. Although the scene seems suitable for evaluating the performance of all kinds of plane extraction algorithms, there are issues that, from our point of view, render the dataset unsuitable. As illustrated in our paper, replicated in chapter 11, the Perceptron laser sensor that was used in 1996 exhibits massive artifacts. Flat surfaces bulge and points near the edges of object are scattered radially. Unless one intends to tailor one's plane extractor to this specific sensor and its artifacts, they impede the development of novel plane extraction methods. Many algorithms evaluated on SegComp overcome this problem by introducing hand-tuned parameters that allow them to overfit to the particularities of the Perceptron laser. The plane extraction approach by Hoover et al. [23], for example, requires ten parameters to be set manually.

5.2 Contribution

With the present contribution, we attempt to solve the problems identified in the previous section. To address the first problem, which consists in the fact that state-of-the-art methods do not make use of any probabilistic sensor models, we propose a new probabilistic approach to the extraction of finite planes that considers ray path information, dubbed PPE for probabilistic plane extraction. PPE is the prototype of an unsimplified greedy algorithm. Similarly to the probabilistic line extraction approach presented in the previous chapter, which starts with the most detailed line possible and iteratively removes vertices until it arrives at a simplified version, PPE starts with an array of so-called atomic planes generated from an organized point cloud. An atomic plane is a plane that represents one lidar endpoint only. Then, by repeatedly creating regular planes from four adjacent atomic planes, extending the regular planes by atomic planes, and merging the regular planes, the algorithm arrives at the final set of detected planes. A maximum-likelihood estimation technique, PPE chooses in each iteration the option that decreases the joint probability of the underlying lidar measurements. The implementation of this algorithm is available free and open-source.

Additionally, in order to overcome the afore-mentioned problems with the SegComp dataset, we propose a novel dataset, called SynPEB for synthetic plane extraction benchmark. It contains synthetic lidar scans of different noise levels sampled in a room filled with diverse polyhedral objects, as illustrated in figure 5.1. Just like the implementation of the PPE algorithm, this dataset is free and publicly available. For more information about both the proposed method and the dataset, please see the original research paper in chapter 11.

5.3 Critical Discussion

The results of the extensive experiments conducted in the scope of this part of the thesis are twofold. On the one hand, PPE outperforms all other compared methods on the new SynPEB dataset in terms of accuracy. The reason for that lies in the probabilistic formulation of the approach. No other method is based on a probabilistic sensor model and uses the underlying ray path information.

On the other hand, PPE shows an average level of performance on SegComp. This does not come as a surprise given the sensor artifacts in the dataset. When looking at the point clouds provided by the lidar sensor, for example, the tabletop looks slightly kinked. In reality, it is perfectly flat, and

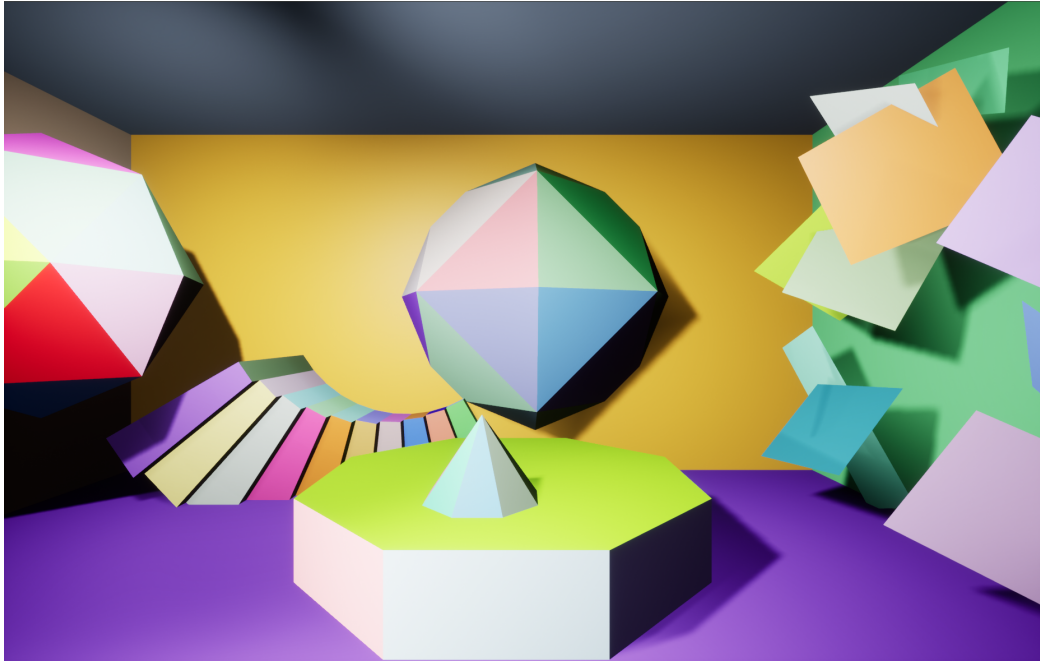


Figure 5.1: Rendered image of the virtual room from which the SynPEB lidar scans are sampled.

hence labeled as a single plane. But without additional information about the systematic errors in the lidar measurements, it is not possible for any plane extraction method to infer whether the sensor is correctly measuring a kinked tabletop or whether it is incorrectly measuring a flat tabletop. In order to mathematically prove that the given labels are not optimal, in the paper, we show that the plane segmentation result provided by PPE actually achieves a higher measurement likelihood than the original ground-truth segmentation. This experiment once again shows that many of the surveyed plane extraction results overfit to SegComp, and it justifies the creation of SynPEB, which, as a synthetic dataset, does not suffer from such inconsistencies.

Despite its merits, the proposed method has its problems. First, it requires the input to be an organized point cloud. An organized point cloud is a point cloud that in addition to providing the point coordinates describes the neighborhood relationships between the individual points. The most common form of neighborhood relationships are matrix-like point arrangements, for example 120×120 or 640×480 . This requirement means that multiple scans or scans captured by a moving laser scanner must be preprocessed before they can be fed to PPE.

Second, PPE is not yet suitable for real-time plane extraction onboard a robotic system. That is due to its high computational complexity. In our experiments, it took us 1.6 h to detect planes in an organized 500×500 point cloud, which is far from any practical requirements. For that reason, we present PPE as a benchmarking method that returns the most accurate plane segmentation result available to date rather than as a method that is ready for use in production code. In order to transform the method accordingly, the algorithm must be simplified, for example by using a superpixel approach or by performing multiple operations like plane creation, plane extension, and plane merging without updating the measurement likelihood after each operation.

Chapter 6

Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans

In the previous chapters, we have addressed different aspects of mapping and localization based on lidar data: We have proposed a novel lidar sensor model, investigated advanced map representations like full posteriors and continuous maps, and we have presented methods for feature extraction. Now, in this chapter, we describe a complete system for localization based on pole features, composed of a novel feature extractor, a mapping module, and an online localization module.

6.1 Research Context

This portion of the thesis addresses the area of autonomous driving, more specifically the problem of localizing a mobile robot in an urban area over long periods of time. Currently, many research institutions and companies are investigating this direction of research. Modern approaches to the problem are often based on features because of the advantages that are connected to them – see chapter 4: Features abstract from the modality of sensor, they improve the robustness of the system, they can encode semantic information, and they lead to memory-efficient maps.

Every feature-based approach must answer the question of what features to use. For mapping and localization in urban environments, pole features are definitely a reasonable choice, because they are ubiquitous, as illustrated



Figure 6.1: Pole-like objects are ubiquitous in a city street. They appear in the form of sign posts, lamp posts, and tree trunks, for example. Source: San Francisco Chronicle website, www.sfchronicle.com, 2019.

in figure 6.1. Street lamp poles, sign posts, traffic light poles, or tree trunks – the geometry of all those objects can be approximated by a standing cylinder or by a tall cuboid with square footprint. In comparison to artificial features, which are computed using an abstract metric based on local lidar endpoint density or on occupancy, they have the advantage of transporting semantic information and of being human-readable.

The present work is not the first approach to pole-based urban vehicle localization based on lidar data. It is, however, the first one that overcomes the problems of comparable state-of-the-art methods. These problems are mainly related to the experiments used to evaluate these approaches: The experiments are often inexpressive, the underlying datasets are small and inaccessible to the public, and the fact that the software used to conduct the experiments is very rarely publicly available makes it impossible to verify the reported performance. For example, the dataset Weng et al. [24] use to evaluate their method is not only proprietary, but it also contains only 3.5 km of trajectory data. Although Sefati et al. [25] test their approach on a session of the publicly available KITTI dataset [26], they use just 46 s of sensor readings. Datasets like those are clearly not suited to demonstrate

the robustness of the proposed methods.

In addition to these issues, the experiments of Wang et al. and Sefati et al. exhibit methodical problems. The authors use the same data for both mapping and localization, although it is obvious that this approach biases the results in their favor.

Problems like these with state-of-the-art methods justify further research. In order to develop a new approach to feature-based mapping and localization, one needs to answer the following design questions. How to design the feature extractor? How to associate landmarks detected in incoming sensor measurements with the correct landmarks in the map and how to deal with multiple detections of the same landmark? What framework to use for localization and how to compute the measurement probability? In order to explore the solution space and to identify the research context, we will present feasible answers to each question in the following, and we will subsequently describe the approach chosen by this work.

There exist many approaches that are suited for extracting pole features from lidar measurements, for example region growing, clustering, RANSAC, the Hough transform, and heuristics. Region growing could be used to segment a point cloud. Given a seed point, the pole segment would iteratively be augmented by all adjacent points that fulfill certain criteria, for example local point density or distance from the nearest segmented point. Clustering is a machine learning technique that returns a segmentation without the need for seed points, while RANSAC would find the parameters of individual poles by randomly taking a small number of points and fitting a pole to them. The Hough transform would produce a map in a rasterized feature space: It would assign a score to each possible pole and would identify probable poles as the modes of this distribution. Heuristics, on the other hand, are handcrafted algorithms that would use a custom set of operations to detect poles.

As mentioned above, the most important questions during mapping are data association and merging multiple detections of the same landmark. Data association refers to the problem of identifying which pole in one measurement corresponds to which pole in another measurement. This process allows us to track poles and to find out whether a new pole needs to be inserted into the map or whether we are looking at a pole that has already been mapped. There are different approaches to data association for distinguishable and indistinguishable landmarks. In the case of pole features, distinguishable landmarks are pole landmarks that are not only characterized by their position, but by their diameter, height, or color, to name just a few possible characteristics. Given a pair of distinguishable poles, we can use a statistical metric in the feature space to determine the statistical distance between these poles, i.e. the probability that the poles are identical, and incorporate

this probability into the posterior map. In case the pole landmarks are indistinguishable, we need to resort to geometric strategies like nearest-neighbor matching or pattern matching.

Merging multiple landmarks refers to the problem of being provided with multiple measurements of the same landmark and having to represent the landmark as a single landmark, for example when creating the map. This problem is often solved by arithmetic averaging, but if in addition to the landmark feature vectors, we are also given a confidence score, we can use the score to compute a weighted average.

In order to localize a robot based on landmarks, we first need to opt for a specific framework. Parametric filters like the Extended Kalman Filter or the Unscented Kalman Filter [10] have the advantage of computational efficiency and of yielding the mathematically optimal location estimate. However, they only support linearized models and Gaussian probability distributions, and their estimate is unimodal. Nonparametric approaches like the particle filter are based on a set of distinct hypotheses: the particles. During the motion update, every particle is propagated according to the motion of the vehicle, and during the measurement update, every particle is weighted by its measurement likelihood. As a consequence, contrarily to parametric filters, the particle filter and its relatives provide multimodal distributions, and they naturally accommodate nonlinear motion models and measurement models. Of course, these benefits come at a cost. The curse of dimensionality tells that the computational complexity of nonparametric filtering increases exponentially with the number of dimensions. Furthermore, since even the largest set of particles can only approximate the underlying probability distribution, the particle filter requires mechanisms to cope with the limited number of hypotheses, for example resampling and smoothing of measurement models that are so accurate that they would otherwise lead to particle deprivation.

In addition to the differences between parametric and nonparametric localization mentioned above, the choice of framework influences how data association decisions are processed. When relying on Kalman filter-based approaches, for example, one needs to choose a certain solution to the data association problem, a decision which cannot be reverted later on. If using a particle filter, however, each particle can integrate a different data association, and the goodness of the individual data association will influence the future particle weight. In this way, the particle filter is able to assess the quality of a data association in retrospective and to build upon the particles that have chosen the correct associations.

To implement localization, we do not only have to decide which framework to use, but also how to compute the measurement probability during the measurement update step. There are basically two ways: to compute the

measurement probability in the measurement space or in the feature space. If we opt for the computation in the measurement space, we project the map features into the sensor space and use the sensor model to compare the true measurements predicted by the projections with the online measurements. This method is expedient if we do not have to deal with dynamic objects, which contradict our latent static-world assumption. If dynamic objects are present, they can occlude the projected objects and distort the measurement probability computation.

In order to compute the measurement probability in feature space, we apply feature extraction to the online measurements and in this way project them from the measurement space to the feature space. After performing data association, we use a specified statistical distance metric to compute the distance between corresponding map features and online features, and interpret this distance as measurement probability.

After having pointed up alternatives to build a pole-based mapping and localization framework, we describe the design at the base of the proposed method in the following.

6.2 Contribution

In this part of the thesis, we present a complete mapping and localization framework, composed of a pole feature extractor, a mapping module, and a localization module.

Speaking in terms of the approaches to feature extraction outlined above, the proposed feature extractor is a heuristic, which follows the steps below. First, it creates a 3-D full-posterior reflection map based on a given set of lidar measurements, which it then transforms into an occupancy grid map. Subsequently, it assigns a pole score to each cell of this grid map. The pole score is based on a geometric pole model: A pole is a vertical stack of voxels with quadratic footprint, laterally surrounded by empty space. After projecting the resulting 3-D pole score map to the ground and applying the mean-shift algorithm to the projection, the algorithm returns the continuous 2-D coordinates of the projection's modes, i.e. the positions where poles are most likely to be found. Like the technical details of any step sketched in the following, the details of feature extraction can be found in the paper in chapter 12.

With this pole extractor, we address the problem of mapping. In a naïve approach to create a global map using the feature detector, we would first try to build a reflection map based on all the data collected during the mapping run. Although possible in theory, memory restrictions will most

often render this method impossible. Consequently, we divide the trajectory into multiple short segments and build corresponding local reflection maps. Extracting pole features from these maps yields a set of partly identical, indistinguishable pole landmarks. In order to merge the identical landmarks, we take a simple geometric approach to data association: Two landmarks are the same if their footprints in the global map overlap. To merge them, we compute a weighted average of their positions based on their scores as provided by the feature detector.

After having created a global map, localization starts. We opt for a particle filter-based approach in order to accommodate the nonlinearity of the model and the multimodal distribution that describes the vehicle's probability of presence. Data association between features extracted online and the map is performed via nearest-neighbor search with a maximum threshold. As mentioned above, in this way, we cover many different possible data associations, which improves the robustness of the filter.

The measurement probability is computed in feature space, because this procedure is robust with respect to dynamic objects and computationally efficient.

A new mapping and localization method is worthless without experimental data that demonstrate its usefulness. To evaluate the proposed approach, we perform two experiment series. The first series replicates the experiment conducted by Sefati et al. [25] on the KITTI dataset [26]. After creating a map based on a very short session, we use the same data again to localize the robot. As mentioned above, the resulting accuracy measures are biased, but the experiment serves the purpose of demonstrating superior accuracy compared to the approach by Sefati et al.

The second experiment series, based on the NCLT dataset [5], demonstrates the long-term performance of our method. In 27 sessions distributed over the course of 15 months, the approach consistently achieves robust and accurate localization results. For a detailed discussion of the experiments and the corresponding results, please see the original paper in chapter 12.

In addition to the complete mapping and localization framework delineated above, this portion of the dissertation is accompanied by a free and open-source Python implementation of both the method and the experiments [27].

6.3 Critical Discussion

The present work pushes the boundary of the state of the art in several respects. First, to the best of our knowledge, our method provides the highest

accuracy compared to other approaches to pole-based mapping and localization. Second, it is the only approach that is evaluated on a long-term dataset. While other works are tested using datasets with durations from below one minute to half an hour, the performance of the presented method was assessed based on 35 hours of data spread over 15 months. Although the corresponding dataset was collected in a dynamic and continuously evolving campus environment, localization results show a high level of robustness against changes. Third, while we could not find an open implementation for any of the related works, we provide free and open-source software implementing our method. The software contains all required modules and all experiment scripts.

There are several directions for further development of the presented method. As described, the pole extractor is based on a heuristic. From our experience, heuristics can achieve astonishing levels of performance, but are most often outperformed by probabilistic approaches. Consequently, the feature extractor would probably benefit from an adaptation to a probabilistic foundation.

Furthermore, poles are currently represented by their planar coordinates only. Incorporating additional features like pole height or diameter would certainly facilitate data association and improve mapping as well as localization performance.

As seen in the experiments in section 12.5, the localization module can diverge if the model assumptions are heavily violated. One of the most important model assumptions is the static-world assumption. It states that mapped features do not move. Small violations of this assumption usually do not have a significant effect: Since the robot uses a number of poles most of the time, it does not matter if one pole is moved as long as the other poles remain steady, because the localizer attributes the same importance to each pole. If multiple poles are moved at the same time by the same amount into the same direction, however, things look different: The localizer “corrects” its estimate to account for the movement, hence the pose estimate diverges. At the same time, the localizer maintains high confidence, because the poles detected online are locally consistent with the poles in the map. Only when the vehicle leaves the problematic area, the detected poles and poles in the map become inconsistent, the particle cloud spreads, and the localization module re-localizes. At the end of our long-term experiment series, exactly that happened: Construction workers had moved a chain of construction barrels to the side, each by the same amount, as shown in figure 6.2.

There are a few ways to enable the system to handle cases like this. One could augment the feature vectors of the poles. By encoding the type of pole and by associating the type “construction barrel” with a higher mobil-



(a) Session 2012-12-01.



(b) Session 2013-02-23.

Figure 6.2: Each image shows a section of the point cloud accumulated over the session of the NCLT dataset indicated in the image caption. Both images depict the same scene, composed of – from left to right – a construction area, a footpath, a road, and a grassy area. In December 2012, the construction barrels are positioned on the left side of the footpath. A few months later, the construction barrels have been moved to the right onto the parking spots. The green lines represent the trajectory of the robot.

ity probability than a tree or a street lamp, the localizer could be helped to attribute more weight to poles that remain steady. Change detection algorithms like the one proposed by Luft et al. [28] would also help improve the localization estimate in presence of dynamics.

In addition to the above-mentioned extensions, incorporation of other types of features like road markings, facades, or curbs is expected to boost localization accuracy and reliability significantly. These features do not necessarily have to be static. If their dynamics can be modeled and if the corresponding parameters are incorporated into the feature vector, even pedestrians or cars can serve as valuable sources of information. Although a car, for example, can vary its longitudinal velocity within certain limits, it is unlikely to go sideways, and can hence provide a sort of one-dimensional information.

While all the features described above correspond to real objects and are thus human-readable, which is an advantage during debugging, there is no guarantee that these semantic features are maximally descriptive. Machine learning techniques could be used to investigate novel, maximally descriptive, not necessarily object-based features, for example by the use of an autoencoder architecture.

Part III
Publications

This part replicates the publications that describe the research conducted within the scope of this thesis. As in the previous part, each chapter is given over to one publication. The publications are displayed unchanged, except for adapted formatting, globally consistent reference numbers, and two additional figures in chapter 11.

Copyright Notice

In reference to IEEE copyrighted material which is used with permission in this part of the thesis, the IEEE does not endorse any of the products or services of the University of Freiburg. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Chapter 7

An Analytical Lidar Sensor Model Based on Ray Path Information

The following paper [29] was written by Alexander Schaefer, Lukas Luft, and Wolfram Burgard. Alexander Schaefer and Lukas Luft contributed equal parts to this work. The article was published in the journal *IEEE Robotics and Automation Letters*, Volume 2, Issue 3, in July 2017. In addition, the work was presented at the *IEEE/RSJ International Conference on Robotics and Automation 2017*, which was held in Singapore from May 29 to June 3, 2017. The IEEE holds the copyright on the article: © 2017 IEEE. Reprinted, with permission, from “An Analytical Lidar Sensor Model Based on Ray Path Information”.

7.1 Abstract

Two core competencies of a mobile robot are to build a map of the environment and to estimate its own pose on the basis of this map and incoming sensor readings. To account for the uncertainties in this process, one typically employs probabilistic state estimation approaches combined with a model of the specific sensor. Over the past years, lidar sensors have become a popular choice for mapping and localization. However, many common lidar models perform poorly in unstructured, unpredictable environments, they lack a consistent physical model for both mapping and localization, and they do not exploit all the information the sensor provides, e.g. out-of-range measurements. In this paper, we introduce a consistent physical model that can be applied to mapping as well as to localization. It naturally deals with unstruc-

tured environments and makes use of both out-of-range measurements and information about the ray path. The approach can be seen as a generalization of the well-established reflection model, but in addition to counting ray reflections and traversals in a specific map cell, it considers the distances that all rays travel inside this cell. We prove that the resulting map maximizes the data likelihood and demonstrate that our model outperforms state-of-the-art sensor models in extensive real-world experiments.

7.2 Introduction

In the context of localization and mapping for mobile robots, sensor models serve two purposes: First, the robot uses them to generate a map from recorded measurements; second, they enable the robot to estimate its pose by relating subsequent sensor information to that map.

In practice, lidar sensors are widely used. They send out laser rays and report how far they travel before they are reflected by an object. Ideally, the output distance reveals the closest object in a particular direction. However, especially in unstructured outdoor environments with vegetation, two consecutive laser scans taken from the same point of view might return significantly different values. The reason lies in the unpredictable interaction between the laser ray and unstructured objects, for example a tree canopy. Ignorance about the thickness of single leaves, their poses, etc., makes reflection a probabilistic process.

For lidar sensors, only a few probabilistic approaches formulate a consistent model for both mapping and localization with grid maps, e.g. the reflection model [11] and related ray-tracing based approaches [30, 31, 32, 33]. They tessellate the environment and assign to each voxel the probability that it reflects an incident laser ray.

In this paper, we introduce a novel probabilistic model for lidar sensors, which is a generalization of the aforementioned reflection model. In contrast to the latter, it relies upon a physical model of the interaction between the laser ray and the environment. We model the probability that a ray traverses a specific region as an exponential decay process. Based on the measurements collected during the mapping process, our sensor model assigns a decay rate to each point in space. During the localization phase, we use this decay-rate map to determine the likelihood of incoming measurements.

This paper is structured as follows: Section 7.3 provides an overview over related work on lidar models. Section 7.4 describes how to build a decay-rate map from lidar measurements and how to compute the measurement likelihood for a given scan. In section 7.5, we prove that decay-rate maps



Figure 7.1: Our mobile robot VIONA while recording the forest dataset.

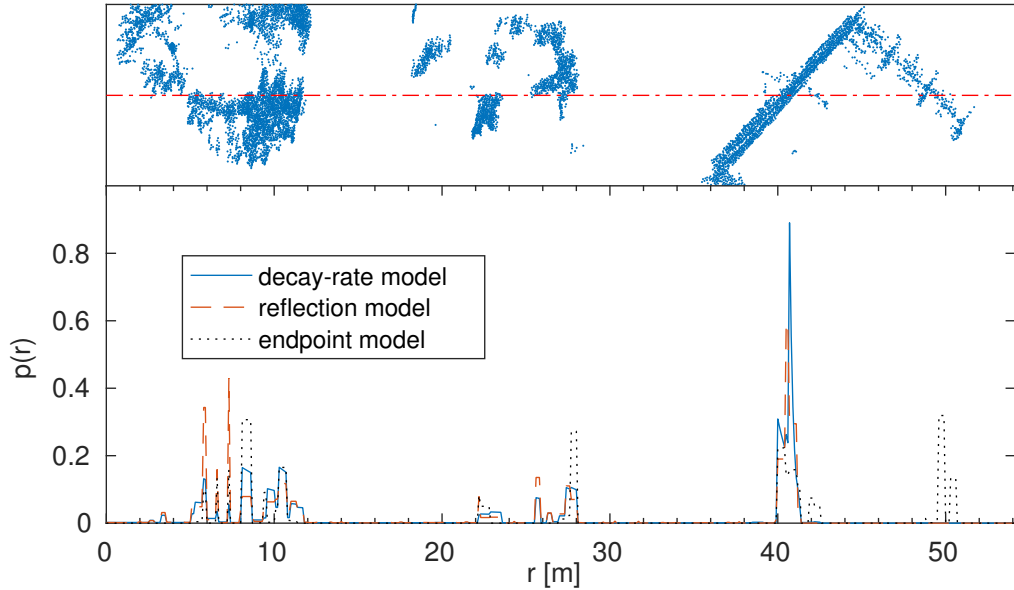


Figure 7.2: The upper part of the image shows a section of the campus environment represented by a point cloud. The dashed line represents a hypothetical laser beam traversing the scene from left to right. It penetrates two treetops and a building. The lower plot shows the corresponding measurement probabilities $p(r)$ obtained by the different sensor models. The endpoint model attributes high probabilities to reflections at the right edge of the building ($r \approx 50$ m) because it ignores the ray trajectory and hence the wall at $r \approx 40$ m. In contrast, the two ray-casting based approaches attribute low probabilities to reflections behind the first wall. The reflection model overestimates the probabilities in the treetops, as it does not account for the distances the rays traveled within the treetop voxels during the mapping process. The overestimations of the endpoint model and the reflection model lead to lower relative probabilities at the left wall of the building.

maximize the likelihood of the underlying data and that our approach generalizes the reflection model. Finally, section 7.6 compares the performance of the proposed approach to state-of-the-art sensor models.

7.3 Related Work

In contrast to our concept, many other approaches address either mapping or localization. Consequently, in the following section, we consider these two categories separately.

7.3.1 Map Representations

Occupancy grid maps, as introduced by Elfes [34], are widely used throughout the robotics community. They divide the environment into cells and assign to each of those a binary random variable that indicates whether the cell contains an object. A binary Bayes filter updates the distribution over these independent variables. As opposed to our model, this approach assumes that the interaction between ray and map is deterministic: The ray is reflected by the first occupied cell on its path.

Point clouds are a direct representation of the reflections measured by the lidar device. However, they neglect out-of-range measurements and valuable information about the ray path.

Likelihood fields [35] heuristically assign to each point in space the likelihood that a ray is reflected. Usually, this likelihood is derived from the distance to the nearest reflection observed during the mapping process. This representation has the advantage that the likelihoods are functions of the space, which can be calculated in advance and stored in a distance map. On the downside, it neglects the ray path information. As a consequence, the likelihood only depends on the endpoint and not on the objects along the ray.

Another popular map representation are reflection maps like used in [11] and [30]. They assign to each cell a reflection probability, which is determined by counting the rays that traverse the cell without reflection – so-called misses – and the rays that are reflected in the cell – so-called hits. Similar to our approach, reflection maps model the interaction between the beam and the map in a probabilistic way. However, in addition to counting hits and misses, our approach considers the distances traveled within each cell. The reflection model discards this information.

Instead of partitioning the map into a set of cubic voxels, Ferri et al. [36] use spherical voxels. Bennewitz et al. [37] explicitly handle erroneous mea-

surements caused by the specific reflection properties of objects. Ahtiainen et al. [38] use the reflection probability of a cell to decide whether it is traversable or not.

In contrast to grid-based approaches, feature-based maps describe the environment by a set of semantic objects. The random finite set formulation as used in [31] and [32] is a way to describe object detections.

There exist lots of other map representations that target specific applications. For example, Limosani et al. [39] use lidar in a long-term mapping run in an office setting to model where dynamic objects like humans are likely to be found.

7.3.2 Sensor Models

Sensor models can be divided into three categories: correlation-based, feature-based, and beam-based models [40]. Correlation-based models relate sensor readings to a given global map. The popular endpoint model [35], for example, evaluates a likelihood field at the ray endpoints. In this way, the endpoint model ignores information about the ray trajectory. If both the global map and the local measurements are represented by point clouds, the iterative closest point method [41] or the normal distributions transform [14] can be used to determine the correlation without the need for an explicit forward sensor model. Feature-based approaches extract features from the sensor readings and compare them to the map.

Our model belongs to the class of beam-based approaches, which explicitly calculate the probability density of the distance measurement along the ray. As further instances of this class, [11] reasons about dynamic objects, and [30] accounts for Gaussian sensor noise and false detections. Thrun et al. [10] derive a basic beam-based model, which De Laet et al. [42] augment by explicitly modeling and marginalizing dynamic objects. Yguel et al. [33] address the problem that beam-based approaches are computationally expensive. They present a GPU-accelerated mapping algorithm for several range sensors with different resolutions. Mullane et al. [43] estimate the grid occupancy probabilities and the corresponding detection likelihoods simultaneously rather than assuming a known measurement model. In this way, their method accounts for false detections.

For a detailed survey on measurement models, see Chapter 12 in [44].

i	voxel index
j	ray index
$k(j)$	index of voxel that reflects ray j
v_i	i^{th} voxel
\mathcal{I}	set of all voxels
$d_i(j)$	distance that ray j travels inside v_i
H_i	total number of reflections in v_i
λ_i	decay rate in v_i
τ_i	mean ray length in v_i
q_i	reflection probability in v_i
s	sensor pose in map frame
m	map
$\lambda(x)$	decay-rate map
r	measured ray length
$p(r)$	probability that ray is reflected at distance r
$N(r)$	probability that ray travels at least distance r
$x(r)$	trajectory of ray for a fixed sensor pose

Table 7.1: Notation.

7.4 Approach

This section describes how to build a decay-rate map from lidar sensor readings and how to calculate the likelihood of a measurement using that map. Table 7.1 provides an overview over the notation used throughout the paper.

7.4.1 The Basic Idea of the Decay-Rate Model

The essence of our approach is to model the probability that a ray traverses a specific region as an exponential decay process. The decay rate of each point in the physical space is stored in a so-called decay-rate map.

To formalize this idea, we define s as the sensor pose, which includes the origin and the direction of the ray, and r as the distance between the sensor and the point of reflection. For ease of notation, we write the measurement

probability as

$$p(r) := p(r \mid s, m). \quad (7.1)$$

In the present paper, we assume that the returned value r is the actual distance traveled by the beam, and model the relation between this distance and the map in a probabilistic fashion. Measurement errors like Gaussian noise and false alarms are not in the scope of the proposed approach. For approaches that account for these uncertainties, please see [44].

Under this assumption, the cumulative probability for a beam to travel at least distance r is

$$N(r) := 1 - \int_0^r p(r') dr'. \quad (7.2)$$

For the measurement probability, it follows

$$p(r) \stackrel{(7.2)}{=} -\frac{dN(r)}{dr}. \quad (7.3)$$

Now, we introduce our essential idea: Locally, $N(r)$ obeys an exponential decay process:

$$\frac{dN(r)}{dr} = -\lambda(r) N(r). \quad (7.4)$$

This model is inspired by the following notion. The physical space is filled with particles, and the probability that a laser ray traverses a region in this space is proportional to the corresponding particle density. Low densities correspond to permeable objects like bushes, while high densities correspond to solid objects like walls. For a ray that penetrates a region of constant particle density, $N(r)$ decreases exponentially over the traveled distance r .

In our model, the decay rate $\lambda(x)$ is a property of the physical space. We obtain $\lambda(r) = \lambda(x(r, s))$ by evaluating the decay rate along the trajectory of the ray.

Solving differential equation (7.4) for constant decay rate λ yields

$$N(r) \stackrel{(7.4)}{=} e^{-\lambda r} \quad (7.5)$$

$$p(r) \stackrel{(7.3)+(7.5)}{=} \lambda e^{-\lambda r}, \quad (7.6)$$

assuming $N(0) = 1$. This solution is the basis of the mapping and localization algorithms derived in the following section.

7.4.2 Mapping

For a given model, a map has to fully determine the interaction between a sensor and the environment. According to equation (7.4), the decay rate λ meets this requirement. Thus, we choose $\lambda(x)$ as map. In order to relate the abstract parameter λ to quantities which the sensor can observe, we introduce τ – the mean length which a ray travels in a hypothetical, infinitely large medium with constant λ , before it is reflected:

$$\tau := \mathbb{E}[r] = \int_0^\infty r \cdot p(r) dr \stackrel{(7.6)}{=} \lambda^{-1}. \quad (7.7)$$

On the basis of a finite number of measurements, the integral can be approximated as

$$\tau = \lambda^{-1} \stackrel{(7.7)}{\approx} H^{-1} \sum_{j \in \mathcal{J}} d(j), \quad (7.8)$$

where H is the number of recorded reflections, \mathcal{J} is the set of measured rays, and $d(j)$ is the distance that ray j travels before it is reflected. To determine $d(j)$, one uses ray tracing between sensor position and reflection point.

To build a map of the environment, we tessellate the physical space using voxels $\{v_i\}_{i \in \mathcal{I}}$ of constant decay rates λ_i , so that the decay-rate becomes a function of physical space:

$$\lambda(x \in v_i) = \lambda_i. \quad (7.9)$$

Inspired by (7.8), we define

$$\lambda_i := \frac{H_i}{\sum_{j \in \mathcal{J}} d_i(j)}, \quad (7.10)$$

where H_i is the number of recorded reflections within v_i , and $d_i(j)$ is the distance that ray j traveled within v_i . With (7.10), we can now determine our map – the set $\{\lambda_i\}_{i \in \mathcal{I}}$ – from sensor measurements. In practice, we have to account for finite memory. Therefore, we compute λ_i for all voxels inside a region of interest and assign a single prior to all points outside.

In section 7.5.1, we prove that the computation of the map parameters λ_i according to (7.10) indeed maximizes the data likelihood.

7.4.3 Localization

During the localization phase, the robot uses the map to assign probabilities to measurements. For a ray starting and ending in the same voxel v_i , (7.6)

readily provides us with this probability. Almost every ray, however, will traverse multiple voxels. In order to determine the corresponding measurement probability, we plug the piecewise constant decay rate as defined in (7.10) into the differential equation (7.4) and solve for $N(r)$:

$$N(r) = \prod_{i \in \mathcal{I}} e^{-\lambda_i d_i}. \quad (7.11)$$

To verify that (7.11) satisfies the differential equation (7.4), we need to differentiate $N(r)$ with respect to r . Doing so, we need to keep in mind that for a particular r , all but the last d_i are constants obtained by ray tracing. Only the distance d_k within the last voxel v_k explicitly depends on r :

$$d_k = r - \sum_{i \in \mathcal{I} \setminus \{k\}} d_i. \quad (7.12)$$

With these prerequisites, the measurement likelihood becomes

$$p(r) \stackrel{(7.3)}{=} -\frac{dN(r)}{dr} \stackrel{(7.11)+(7.12)}{=} \lambda_k \prod_{i \in \mathcal{I}} e^{-\lambda_i d_i}. \quad (7.13)$$

As described above, the values computed during mapping (7.10) and localization (7.13) are mainly linear combinations of values obtained by ray tracing. Thus, the complexity of our method is determined by the complexity of the used ray tracing algorithm. In particular, our approach has the same complexity as the reflection model, while it makes use of more measurement information.

7.4.4 Integrating Out-of-Range Measurements

Until now, we have implicitly assumed that the sensor always returns a real value r . In practice, however, lidar sensors have a limited range $[r_{\min}; r_{\max}]$. They return

$$z := \begin{cases} \text{sub} & \text{for reflections below } r_{\min} \\ r & \text{for reflections in } [r_{\min}; r_{\max}] \\ \text{sup} & \text{for reflections above } r_{\max} \end{cases} \quad (7.14)$$

Consequently, the measurement probability of a scan that contains J rays becomes a mixture of probability densities and absolute probabilities:

$$\begin{aligned} & p(z_1, \dots, z_J \mid s_1, \dots, s_J, m) \\ & \sim \prod_{j \in \mathcal{J}_{\text{sub}}} P(\text{sub} \mid s_j, m) \cdot \prod_{j \in \mathcal{J}_{\mathbb{R}}} p(r_j \mid s_j, m) \cdot \prod_{j \in \mathcal{J}_{\text{sup}}} P(\text{sup} \mid s_j, m), \end{aligned} \quad (7.15)$$

where \mathcal{J}_{sub} , $\mathcal{J}_{\mathbb{R}}$, and \mathcal{J}_{sup} are the sets of ray indices that correspond to $z_j = \text{sub}$, $z_j = r_j$, and $z_j = \text{sup}$, respectively.

To compute the probabilities of out-of-range measurements, we integrate over all real values which they represent:

$$\begin{aligned} P(\text{sub} \mid s_j, m) &= \int_0^{r_{\min}} p(r \mid s_j, m) dr \\ P(\text{sup} \mid s_j, m) &= \int_{r_{\max}}^{\infty} p(r \mid s_j, m) dr, \end{aligned} \quad (7.16)$$

with $p(r \mid s_j, m)$ as in (7.13).

In the context of localization, the fact that (7.15) represents a mixture of probability densities and absolute probabilities does not bother us, as we are typically interested in the relative probabilities between pose hypotheses. To obtain absolute probabilities, the measurement likelihood provided by (7.15) has to be normalized.

7.5 Mathematical Details

This section proves that the proposed mapping algorithm maximizes the data likelihood and derives the reflection model from our more general approach.

7.5.1 Decay-Rate Maps Maximize the Data Likelihood

We prove that the map parameters λ_i according to (7.10) maximize the likelihood of the underlying data by solving the following optimization problem:

$$\begin{aligned} m^* &= \operatorname{argmax}_{m=\{\lambda_i\}_{i \in \mathcal{I}}} p(r_1, \dots, r_J \mid s_1, \dots, s_J, m) \\ &= \operatorname{argmax}_m \prod_{j \in \mathcal{J}} p(r_j \mid s_j, m) \\ &= \operatorname{argmax}_m \sum_{j \in \mathcal{J}} \log p(r_j \mid s_j, m) \\ &\stackrel{(7.13)}{=} \operatorname{argmax}_m \sum_{j \in \mathcal{J}} \log \left(\lambda_{k(j)} \prod_{i \in \mathcal{I}} e^{-\lambda_i d_i(j)} \right) \\ &= \operatorname{argmax}_m \underbrace{\sum_{j \in \mathcal{J}} \left(\log(\lambda_{k(j)}) - \sum_{i \in \mathcal{I}} \lambda_i d_i(j) \right)}_{=: f(m)}. \end{aligned} \quad (7.17)$$

With

$$\frac{\partial \log(\lambda_{k(j)})}{\partial \lambda_i} = \begin{cases} \frac{1}{\lambda_i} & \text{if } k(j) = i \\ 0 & \text{otherwise} \end{cases} \quad (7.18)$$

we obtain the partial derivatives of f :

$$\frac{\partial f(m)}{\partial \lambda_i} = \frac{H_i}{\lambda_i} - \sum_{j \in \mathcal{J}} d_i(j). \quad (7.19)$$

Equation (7.10) satisfies both the necessary condition for m^*

$$\frac{\partial f(m)}{\partial \lambda_i} = 0 \quad \forall i \in \mathcal{I} \quad (7.20)$$

and the sufficient condition

$$\frac{\partial^2 f(\lambda)}{\partial \lambda_i^2} = -\frac{H_i}{\lambda_i^2} < 0. \quad (7.21)$$

Hence, the decay-rate map computed according to (7.10) is the most probable map given the sensor data.

7.5.2 The Decay-Rate Model Generalizes the Reflection Model

In order to show that the decay-rate model is a generalization of the reflection model, we derive the latter from our approach with additional restrictive assumptions. Reflection maps assign to each cell of the physical space a reflection probability

$$q_i = \frac{H_i}{H_i + M_i}, \quad (7.22)$$

where H_i is the number of reflections in v_i recorded during mapping, and where M_i is the number of rays that penetrated v_i . The model states that the probability of a ray ending in v_k is

$$P(x(r) \in v_k \mid s, m) = q_k \prod_{i \in \mathcal{B}(k)} (1 - q_i), \quad (7.23)$$

where $\mathcal{B}(k)$ denotes the indices of the voxels through which the ray travels.

The first assumption inherent to the reflection model is that every ray travels the same distance in every voxel it traverses:

$$d_i = \begin{cases} d & \text{if } i \in \mathcal{B} \\ 0 & \text{if } i \notin \mathcal{B} \end{cases} \quad (7.24)$$

Applying this assumption to the decay-rate model, we obtain

$$\lambda_i \stackrel{(7.10)}{=} \frac{H_i}{\sum_{j \in \mathcal{J}} d_i(j)} \stackrel{(7.22)+(7.24)}{=} \frac{q_i}{d}. \quad (7.25)$$

With this simplified version of λ_i , we derive the reflection probability (7.23) from our model:

$$\begin{aligned} P(x(r) \in v_k \mid s, m) &= \int_{x(r) \in v_k} p(r \mid s, m) dr & (7.26) \\ &\stackrel{(7.13)}{=} \int_{x(r) \in v_k} \lambda_k \prod_{i \in \mathcal{I}} e^{-\lambda_i d_i} dr \\ &\stackrel{(7.25)}{=} \int_{x(r) \in v_k} \frac{q_k}{d} \prod_{i \in \mathcal{B}} e^{-q_i} dr \\ &\stackrel{(7.24)}{=} q_k \prod_{i \in \mathcal{B}} e^{-q_i} \\ &\approx q_k \prod_{i \in \mathcal{B}} (1 - q_i). \end{aligned}$$

The second simplification implicitly made by the reflection model expresses itself in the transition between the last two lines: The model aborts the Taylor series of the exponential after the first derivative.

We just argued that the standard reflection model can be seen as a special case of the decay-rate model. Another way of looking at the relation between the two models is that the decay-rate approach is formally equivalent to the standard approach in a grid where each cell is partitioned into subvoxels with constant q_i within the original grid cell v_i . This can be seen as follows. The length of a ray that travels through the grid can be expressed by the number of traveled subvoxels n and the subvoxel size l as $r = nl$. We get the cumulative distribution

$$N(r) = (1 - q_i)^{nl} = e^{\log(1 - q_i)nl}, \quad (7.27)$$

which obeys an exponential decay and has the same form as the decay-rate model (7.5). Thus, one can formally switch from the decay-rate model to a fine-grained version of the reflection model by choosing the values $\{q_i\}$ such that $\log(1 - q_i) = -\lambda_i$.

7.6 Experiments

In order to evaluate the proposed approach, we conduct extensive real-world experiments. The data processed in these experiments were collected with the mobile off-road robot VIONA by Robot Makers, equipped with a Velodyne HDL-64E lidar sensor and an Applanix POS LV localization system. We use the Applanix system, which fuses information coming from multiple GPS receivers, an IMU, and odometry sensors, as highly accurate pose ground truth. The datasets were recorded in three different environments: on the campus of the University of Freiburg, on a small trail in the middle of a forest, and in a park. All scenarios contain pedestrians. The length of the recorded trajectories varied between 50 m and 400 m. Figure 7.1 shows the robot while recording the forest dataset. Figure 7.3 shows the point clouds of the three datasets.

In the experiments, we compare the decay-rate model to two well-established, state-of-the-art sensor models: the reflection model [11] and the endpoint model [35]. For an illustration of the differences between these models, see figure 7.2.

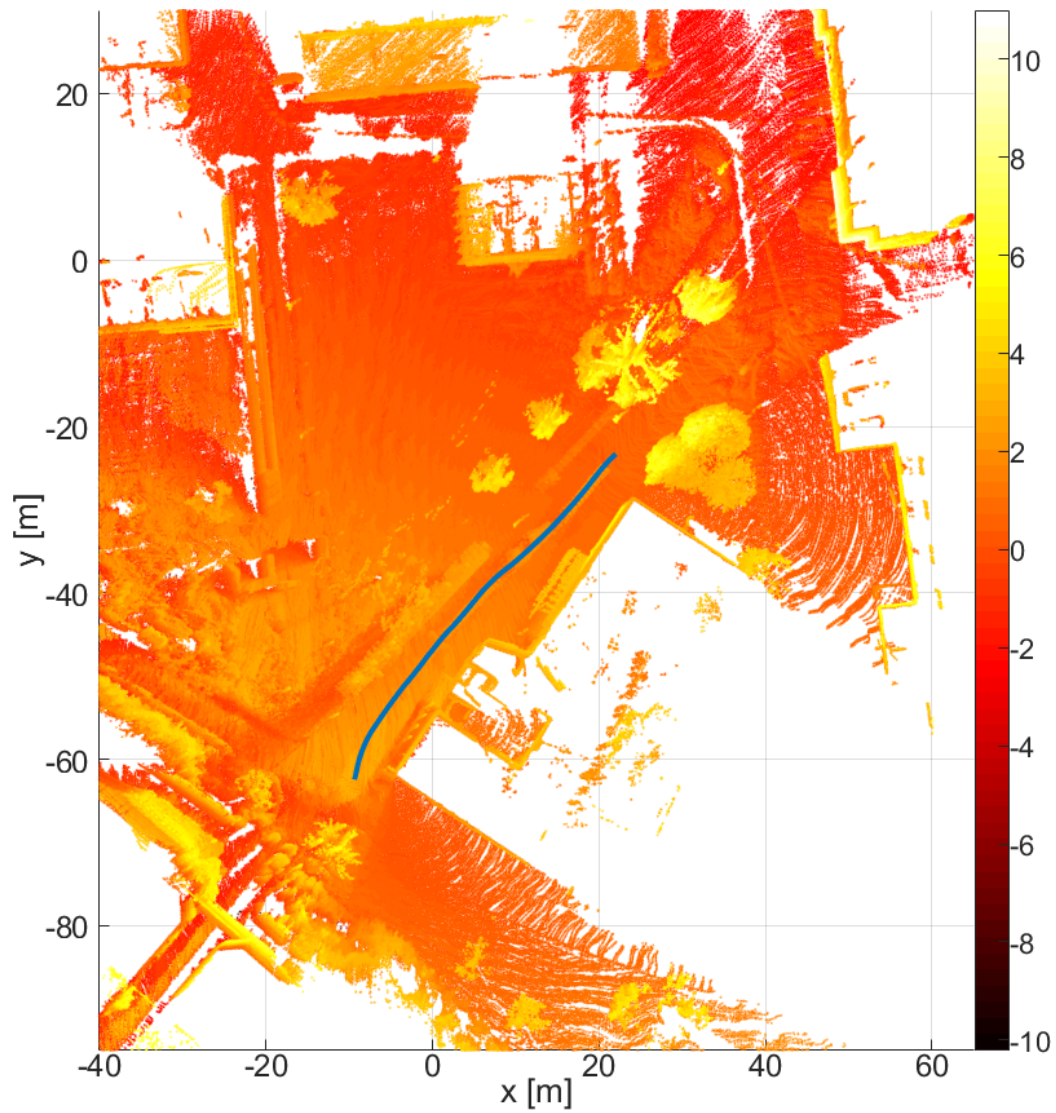
The set of measurements for mapping and the set for localization are disjoint. We use the pose ground truth to perform mapping with known poses for the different environments. While our approach is applicable to any tessellation, in our experiments, we build maps consisting of cubic axis-aligned voxels with an edge length of 0.5 m. This way, the campus maps contain $444 \times 406 \times 43$ voxels, the park maps contain $515 \times 561 \times 41$ voxels, and the forest maps contain $393 \times 403 \times 86$ voxels.

7.6.1 Monte-Carlo Localization

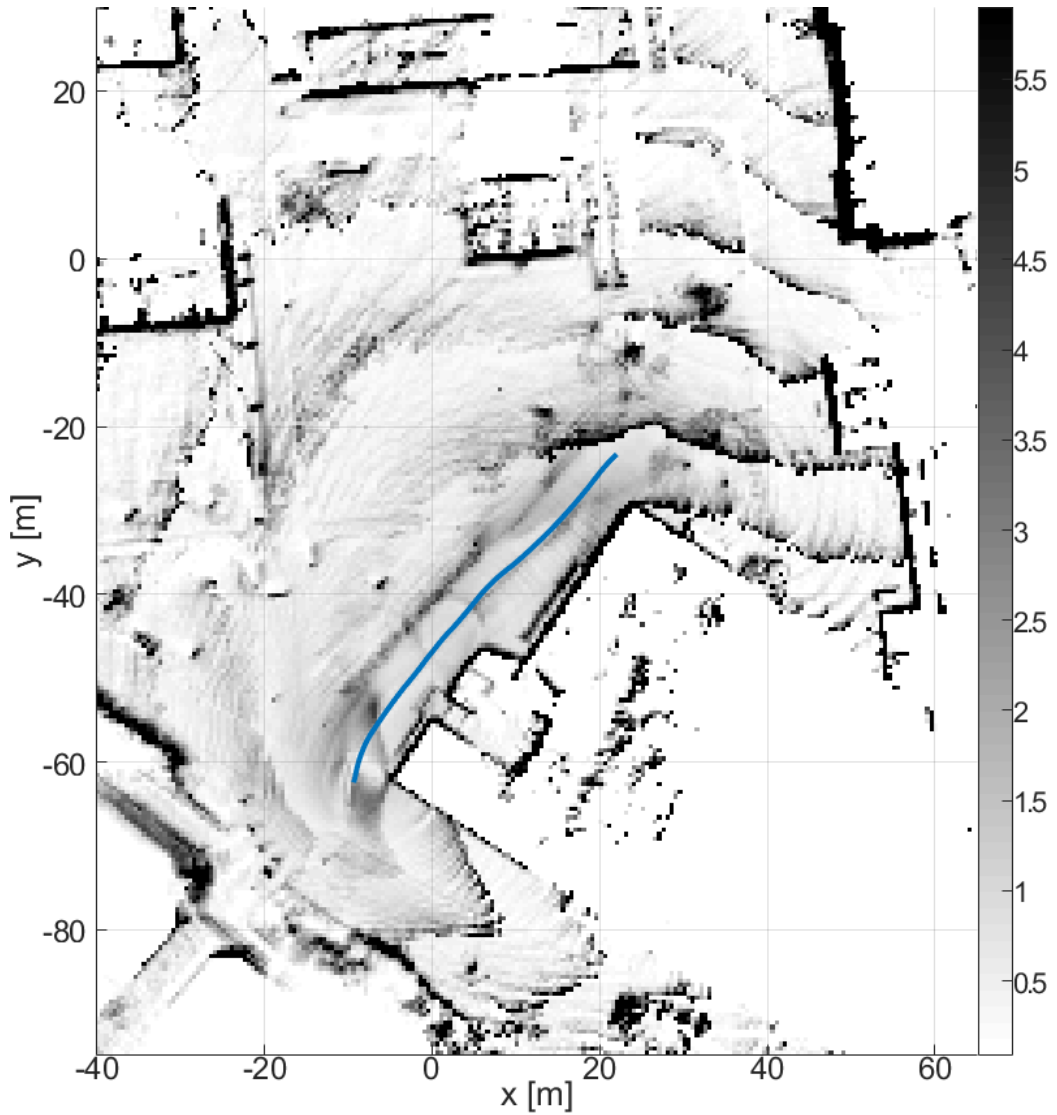
One of the main applications for sensor models is mobile robot localization. In order to compare the different models with respect to localization accuracy in six dimensions, we run separate, identically parameterized particle filters for the three environments. The filters only differ in the measurement models used to weight the particles in the correction step: The first filter employs the decay-rate model proposed in this paper, the second employs the reflectivity model, and the third employs the endpoint model.

We use 300 particles sampled from a Gaussian distribution with a variance of 1 m in the horizontal plane, 0.2 m vertically, and 0.1 rad in every rotational dimension. The offset between the mean particle pose in the initialization step and the ground-truth start pose is sampled from this distribution, too.

To compare the robustness of the models, we also simulated sensor failures in the campus dataset by setting 10% of the measurements to the minimum



(a) A section of the point cloud built from the campus dataset. The point heights are color-coded; the colorbar on the right tells which color denotes which height above the start position of the robot in [m].



(b) A section of the decay-rate map built from the campus dataset. This projection of the 3D decay-rate map onto the x - y plane is computed by summing up the decay rates in z -direction. The colorbar on the right shows which color denotes which decay rate in $[1/m]$. Note that tree trunks are assigned a high decay rate, whereas the canopies have lower decay rates, e.g. at $(20, -10)$.

Figure 7.3: Bird's eye view of a section of the campus dataset. The blue curve shows the robot trajectory ground truth as recorded by the Applanix localization system. The robot travels along a footpath that is framed by a small lawn with trees and bushes on the left and by a building on the right.

	Decay-rate model	Reflection model	Endpoint model
Campus	0.230	0.284	0.280
Campus*	0.252	0.284	0.366
Forest	0.331	0.352	0.417
Park	0.088	0.089	0.124

Table 7.2: Particle filter estimation errors as Euclidean distances between ground truth and estimated position in [m], averaged over time. The scenario campus* includes simulated sensor failures.

sensor range.

Table 7.2 shows the resulting averaged Euclidean distances between estimated and true poses for all recorded datasets and for the campus dataset with simulated sensor failures.

7.6.2 Evaluation of the Pose Likelihood

To evaluate the measurement models independently of filter design, we employ two metrics that assess how well the pose likelihood derived from the output of the models matches ground truth. First, we use the Kullback-Leibler divergence $\mathcal{D}(g||h)$ to relate the pose likelihood h to the ground truth g , which we approximate as a Dirac distribution. With $z = \{z_1, \dots, z_J\}$ and $s = \{s_1, \dots, s_J\}$, we state:

$$\begin{aligned}
\mathcal{D}(g||h) &= \int g(s') \log \left(\frac{g(s')}{h(s')} \right) ds' & (7.28) \\
&= \int \delta(s' - s) \log \left(\frac{\delta(s' - s)}{p(s' | m, z)} \right) ds' \\
&= -\log [p(s | z, m)] + \eta \\
&= -\log [p(z | s, m)] + \eta' \\
&= -\sum_{j=1}^J \log (p(z_j | s_j, m)) + \eta' \\
&=: \mathcal{D}'(g||h) + \eta'.
\end{aligned}$$

In the evaluation, we omit the constant factor η' , as it is independent of the sensor model. $\mathcal{D}'(g||h)$ rewards high likelihoods at the real robot position, but it does not punish high likelihoods far from the real position. To account for these false positives, we also employ the inverse Kullback-Leibler

	Decay-rate model	Reflection model	Endpoint model
Campus	$6.07 \cdot 10^4$	$6.99 \cdot 10^4$	$1.01 \cdot 10^5$
Forest	$2.70 \cdot 10^4$	$3.33 \cdot 10^4$	$5.02 \cdot 10^4$
Park	$1.11 \cdot 10^8$	$1.14 \cdot 10^9$	$1.16 \cdot 10^9$

(a) Divergence $\mathcal{D}'(g\|h)$ between Dirac-distributed ground truth and pose likelihood as defined in (7.28). Low values indicate high pose likelihoods at the true position. The values are computed over all measurements in the dataset.

	Decay-rate model	Reflection model	Endpoint model
Campus	1.87	4.44	2.09
Forest	0.96	1.41	1.14
Park	3.56	4.64	4.17

(b) Inverse Kullback-Leibler divergence $\mathcal{D}(h\|g)$ between Gauss-distributed ground truth and pose likelihood as in (7.29), averaged over all scans. Low values indicate low pose likelihood far away from the true pose. We used $M=50$ samples from a uniform distribution within a circular area with radius 2.5 m centered at the true robot pose.

Table 7.3: Kullback-Leibler divergence between ground truth distribution and pose likelihood for different sensor models. For both metrics, smaller numbers correspond to higher similarity to ground truth.

divergence

$$\begin{aligned} \mathcal{D}(h\|g) &= \int p(s' | m, z) \log \left(\frac{p(s' | m, z)}{\mathcal{N}(s'; s, \Sigma)} \right) ds' \\ &\approx \sum_{i=1}^M p(s_i | m, z) \log \left(\frac{p(s_i | m, z)}{\mathcal{N}(s_i; s, \Sigma)} \right). \end{aligned} \quad (7.29)$$

To approximate the integral, we sum over M poses s_i sampled from a uniform distribution in a circular area centered at the true pose s . We then obtain $p(s_i | m, z)$ by normalizing $p(z | s_i, m)$ over all s_i and assume the real position to be distributed according to $\mathcal{N}(s'; s, \Sigma)$. Plagemann et al. [40] use a similar metric. Table 7.3 shows the corresponding results.

Note that it is impossible to directly compare the output of the three models, as one model returns absolute probabilities, the other probability densities, and yet another heuristic values. To account for that, we always convert the output for real-valued measurements to probability densities and the output for out-of-range measurements to absolute probabilities, as described in the following.

The reflection model yields absolute probabilities for both real-valued and out-of-range measurements. For the former, we assume an underlying density that is implicitly integrated over the voxel that reflects the ray:

$$P(x(r) \in v_k) = \int_{r|x(r') \in v_i} p(r' | s, m) dr'. \quad (7.30)$$

As all rays ending in one voxel have the same probability, we conclude

$$P(x(r) \in v_k) = p(r | s, m) \int_{r'|x(r') \in v_i} dr'. \quad (7.31)$$

Now we can identify the underlying probability density

$$p(r | s, m) = P(x(r) \in v_k) \left(\int_{r'|x(r') \in v_i} dr' \right)^{-1}. \quad (7.32)$$

The endpoint model assumes an absolute probability P as prior for out-of-range measurements. For measurements within the sensor range, it outputs heuristic values. To obtain the corresponding probability density, for each ray, we normalize the integral over all values within the sensor range to $1 - P$.

As the decay-rate model already expresses the probabilities as required, all models are now comparable to one another.

7.6.3 Discussion of Results

The results of the localization experiments are listed in table 7.2. The proposed decay-rate model outperforms the two standard approaches on all datasets. This is due to the fact that the decay-rate model leverages more of the information the sensor provides.

In the campus environment, the endpoint model performs better than the reflection model. In the other, less structured environments, and in the scenario with sensor failures, the reflection model outperforms the endpoint model. We attribute this to the fact that especially in unstructured environments, the ray path information is more informative than the distance to the nearest point.

A comparison of the results of the campus dataset with campus*, which contains simulated sensor failures, indicates that the two beam-based approaches are more robust against outliers than the endpoint model.

Although we chose a poor initial estimate, the particle filter converges to the true position for all datasets and all models. The park dataset is recorded over the longest period of time. Therefore, the bad initialization has less impact in this scenario than in the other three.

The evaluation of the pose likelihoods are listed in table 7.3. These results are more informative than the particle filtering results, as the latter are influenced by parameters and design choices. The proposed decay-rate model outperforms the two baseline approaches in all scenarios: In contrast to the endpoint model, it leverages ray-path information, and in contrast to the reflection model, it considers the distances the rays traveled within the cells.

7.7 Conclusion and Future Work

In this paper, we introduce a physics-inspired, probabilistic lidar sensor model. As a generalization of the reflection model, it can consistently be applied to both mapping and localization. We prove that the resulting maps maximize the data likelihood. In extensive experiments, our model outperforms state-of-the-art measurement models in terms of accuracy.

Our approach models the uncertainty in the interaction between a ray and the environment. In the future, we will extend it to account for additional measurement uncertainties like Gaussian noise and false detections. Currently, we are working on a GPU-accelerated, real-time capable implementation on our off-road robot and plan to build a SLAM framework based on the proposed approach. We will evaluate the lidar calibration performance using ground-truth data obtained by SLAM, and we will also investigate different front-ends and methods for data association. In this context, we plan to benchmark the localization accuracy and the computational requirements of all three sensor models. We are also working on a differentiable extension of our model.

Chapter 8

Closed-Form Full Map Posteriors for Robot Localization with Lidar Sensors

This contribution [45] was authored by Lukas Luft, Alexander Schaefer, Tobias Schubert, and Wolfram Burgard. Lukas Luft and Alexander Schaefer contributed equal parts to the work. It was accepted for and presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems 2017, held in Vancouver, British Columbia, Canada, from September 24 to 28, 2017. The IEEE holds the copyright on the article: © 2017 IEEE. Reprinted, with permission, from “Closed-Form Full Map Posteriors”.

8.1 Abstract

A popular class of lidar-based grid mapping algorithms computes for each map cell the probability that it reflects an incident laser beam. These algorithms typically determine the map as the set of reflection probabilities that maximizes the likelihood of the underlying laser data and do not compute the full posterior distribution over all possible maps. Thereby, they discard crucial information about the confidence of the estimate. The approach presented in this paper preserves this information by determining the full map posterior. In general, this problem is hard because distributions over real-valued quantities can possess infinitely many dimensions. However, for two state-of-the-art beam-based lidar models, our approach yields closed-form map posteriors that possess only two parameters per cell. Even better, these posteriors come for free, in the sense that they use the same parameters as the traditional approaches, without the need for additional computations.

An important use case for grid maps is robot localization, which we formulate as Bayesian filtering based on the closed-form map posterior rather than based on a single map. The resulting measurement likelihoods can also be expressed in closed form. In simulations and extensive real-world experiments, we show that leveraging the full map posterior improves the localization accuracy compared to approaches that use the most likely map.

8.2 Introduction

Robot mapping and localization are probabilistic processes. Therefore, it is desirable to determine the posterior probability distribution over all possible maps given all observations rather than to determine a particular map. For some types of grid maps, it is well-known how to compute this distribution.

Grid maps are a popular representation of the environment of a robot. In their basic formulation, each voxel of the map holds a binary value which expresses whether the voxel is occupied or not. For these so-called occupancy grids, the posterior distribution over each occupancy state is characterized by one real-valued parameter. Moravec [12] and Elfes [13, 34] show how to compute this parameter.

In real-world scenarios, however, map voxels are not always completely free or completely occupied. They often contain structures smaller than the grid resolution. As occupancy grids are not capable of representing these structures, it makes sense to use real-valued maps.

A popular example of real-valued maps are so-called reflection maps [11]. Another method which also characterizes cells by real values builds so-called decay-rate maps [29]. In contrast to posteriors over discrete map values, posteriors over real-valued maps, like reflection maps and decay-rate maps, can contain infinitely many parameters. Therefore, one typically only computes the mode of the map posterior and uses it as map – with few exceptions [46].

In this paper, we present a method to derive the full posterior over real-valued grid maps based on data provided by lidar sensors. Our approach, which is applicable to a broad class of forward sensor models, relies on a rigorous Bayesian formulation of the mapping process. For the reflection model and the decay-rate model in particular, the proposed approach leads to closed-form map posteriors. These posteriors come for free: The most likely map already contains the required parameters.

In addition, we leverage the full map posterior for robot localization, the process of estimating the belief over the robot pose. Just like approaches that use a single given map, we can express the recursive Bayesian update in closed-form. Although our approach possesses the same computational

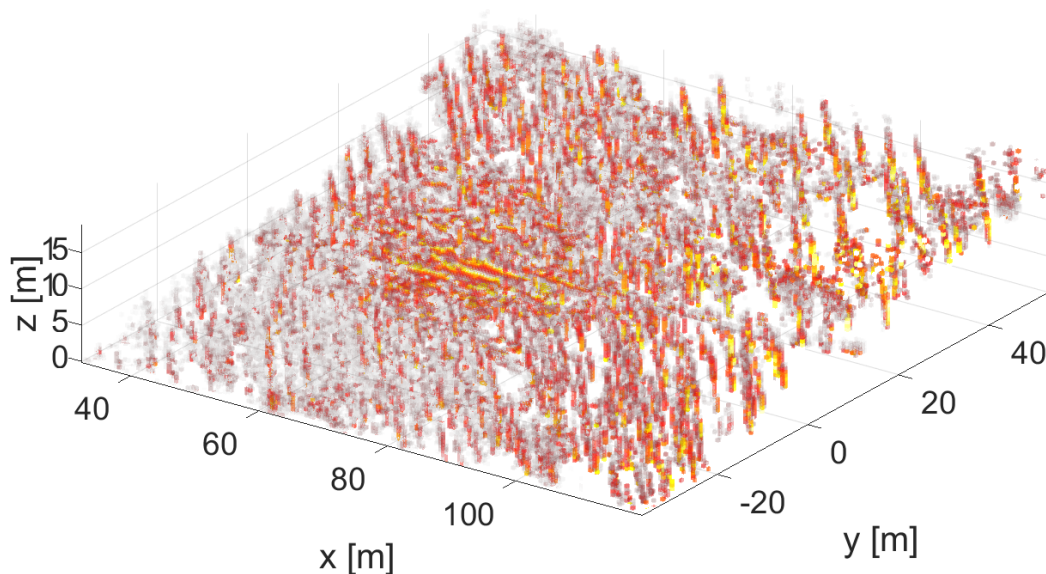


Figure 8.1: Perspective view of a section of the reflection map built from the forest dataset. The map encodes the reflection probability of each voxel by the voxel color: Bright yellow corresponds to low reflection probability, dark red corresponds to high reflection probability. Although the map is highly cluttered, one can clearly recognize a large number of tree trunks.

complexity as the former ones, we demonstrate that it yields higher accuracies in extensive localization experiments.

8.3 Related Work

The proposed approach computes the posterior over real-valued grid maps. Therefore, we structure our overview of the related work as follows: We start with grid-based mapping approaches that compute the full posterior, move on to approaches which compute the most likely real-valued grid map, and close with posteriors over feature-based maps.

In robotics, occupancy grid maps are widely used. To derive the posterior over their binary values, most approaches assume that the individual voxels are independent. Then, the binary Bayes filter allows to recursively update the map posterior based upon the inverse sensor model, as shown by Moravec [12] and Elfes [13], [34]. To obtain a full posterior over a discrete map in the context of SLAM, Doucet et al. [47] and Tipaldi et al. [48] employ a Rao-Blackwellized particle filter [49]. Each of the particles represents not only a pose hypothesis, but also holds a distribution over a discrete

map. Thrun [30] uses a forward sensor model to compute posteriors over grid maps. He drops the assumption of voxel independence by accounting for measurement noise. Marks et al. [46] present an approach to compute posterior distributions over real-valued grid maps; in their case, each map voxel represents the height variance of the surface.

Other than Marks et al. [46], most approaches for real-valued grid maps compute the most likely map only: Hähnel et al. [11], for example, extend the reflection model by introducing a binary variable that expresses whether a reflection is caused by a dynamic object or a static object. The recently introduced decay-rate model [29] also produces real-valued grid maps. These maps represent decay rates of the laser ray instead of reflection probabilities. All approaches in this paragraph have in common that they consistently leverage the forward sensor model for both mapping and localization.

Instead of using voxels, maps can also be represented by a finite set of landmarks. Extended Kalman Filtering techniques assume the robot pose and the positions of these landmarks to be normally distributed and compute the full posterior over the robot pose and the map in closed form. For one example among a wide range of publications in this context, see [50]. Extended Kalman Filtering is also a popular choice for collaborative localization, where robots use their teammates as moving landmarks. The belief of the joint pose state can then be interpreted as posterior over a dynamic landmark map, see for example [51].

8.4 Approach

This section describes how to calculate the full posterior over a real-valued grid map and how to use it for localization. Our approach is applicable to a broad class of beam-based sensor models, which we define in section 8.4.1. We call them factorizing models. In this section, we also recall the formulas for two examples of this class: the reflection model and the decay-rate model. In section 8.4.2, we derive a recursive update equation to compute the posteriors over grid maps based on factorizing models. We leverage this equation in section 8.4.3 to derive closed-form posteriors over reflection maps and decay-rate maps. Once the posteriors are established, we move on to perform robot localization: Section 8.4.4 establishes a general recursive Bayesian update equation for localization based on map posteriors rather than based on a single given map. For the reflection model and the decay-rate model, this update equation possesses closed form, as presented in section 8.4.5. Table 8.1 provides an overview over the notation used throughout the paper.

i	voxel index
v_i	i^{th} voxel
H_i	total number of hits in v_i during mapping
M_i	total number of misses in v_i during mapping
λ_i	decay rate in v_i
μ_i	reflection probability in v_i
x	sensor pose with respect to map frame
m	map
m_i	map value in voxel v_i
r	radius of a laser ray
r_i	distance that a ray travels inside v_i
R_i	total distance that all rays travel in v_i during mapping
\mathcal{I}	set of all voxels
$N = \mathcal{I} $	number of all voxels
$\mathcal{I}(r, x)$	set of voxels entered by a beam with r and x
X_m	sensor poses during mapping
Z	sensor measurements recorded during localization
Z_m	sensor measurements recorded during mapping
z	most recent measurement

Table 8.1: Notation

8.4.1 Factorizing Forward Sensor Models

The present paper deals with mapping and localizing based on lidar data. The formalism presented later on is valid for a broad class of sensor models which we call factorizing models. They are characterized by the following property:

$$p(r \mid x, m) = \prod_{i \in \mathcal{I}(r, x)} f(r_i, m_i, \delta_i), \quad (8.1)$$

where r is the length of the measured laser ray, x denotes the sensor pose, and m is a fixed map, $\mathcal{I}(r, x)$ is the set of indices of all voxels which the beam enters, r_i is the radius that the beam travels within voxel v_i , m_i denotes the map value of voxel i , and $\delta_i(r, x)$ tells whether or not voxel i reflects the ray:

$$\delta_i = \delta_i(r, x) = \begin{cases} 1 & \text{if } v_i \text{ reflects the ray} \\ 0 & \text{else} \end{cases} \quad (8.2)$$

The reflection model [11] defines $f(r_i, m_i, \delta_i)$ as a binomial distribution over the event δ_i :

$$f(r_i, \mu_i, \delta_i) = \mu_i^{\delta_i} (1 - \mu_i)^{1 - \delta_i} = f(\mu_i, \delta_i). \quad (8.3)$$

It disregards the information about how far the ray travels inside each voxel: (8.3) does not depend on r_i .

In contrast, the decay-rate model [29] incorporates this information, as it computes f as follows:

$$f(r_i, \lambda_i, \delta_i) = \lambda_i^{\delta_i} e^{-\lambda_i r_i}. \quad (8.4)$$

Here, λ_i is the decay rate within voxel i . If we fix the value of r_i , equation (8.4) yields a binary Poisson distribution over δ_i . Conversely, for $\delta_i = 1$, it yields an exponential distribution over r_i .

8.4.2 Recursive Map Update

This section addresses the core of our approach. We show how to calculate a posterior distribution over all maps. This distribution is called the belief $bel(m) := p(m \mid Z_m, X_m)$, where Z_m denotes the set of all measurements recorded during the mapping process, and where X_m denotes the set of corresponding sensor poses. In order to calculate the full map posterior, we first

need to introduce the following definitions.

$$bel(m) = p(m \mid Z_m, X_m) \quad (8.5)$$

$$\overline{bel}(m) = p(m \mid \overline{Z}_m, X_m) \quad (8.6)$$

$$bel(m_i) = p(m_i \mid Z_m, X_m) \quad (8.7)$$

$$\overline{bel}(m_i) = p(m_i \mid \overline{Z}_m, X_m) \quad (8.8)$$

Here, $\overline{Z}_m = Z_m \setminus \{z\}$ represents the set of mapping measurements without the most recent measurement z . Note that when referring to general sensor models, we call the sensor output z ; only in the context of factorizing models, we write r for radius. Proposition 1 now shows how to recursively compute the full map posterior from the latest belief and the latest measurement.

Proposition 1. Assuming a factorizing sensor model

$$p(r \mid x, m) = \prod_{i \in \mathcal{I}(r, x)} f(r_i, m_i, \delta_i), \quad (8.9)$$

and mutual independence of the individual voxels

$$bel(m) = \prod_{i=1}^N bel(m_i), \quad (8.10)$$

the belief over each map value is recursively updated according to

$$bel(m_i) = \eta_i \overline{bel}(m_i) f(r_i, m_i, \delta_i), \quad (8.11)$$

where η_i is a normalizing constant independent of m_i .

Proof. To prove the above proposition, we define the following notation:

$$m_{\mathcal{I} \setminus i} := m \setminus \{m_i\} \quad (8.12)$$

and

$$\int_{m_{\mathcal{I} \setminus i}} (\cdot) dm_{\mathcal{I} \setminus i} := \int_{m_1} \cdots \int_{m_{i-1}} \int_{m_{i+1}} \cdots \int_{m_N} (\cdot) dm_N \cdots dm_{i+1} dm_{i-1} \cdots dm_1.$$

Using this notation, we derive proposition 1 as follows:

$$\begin{aligned}
bel(m_i) &= p(m_i \mid Z_m, X_m) \\
&\stackrel{(8.8)}{=} \eta p(z \mid m_i, \bar{Z}_m, X_m) \bar{bel}(m_i) \\
&= \eta \bar{bel}(m_i) \int_{m_{\mathcal{I} \setminus i}} p(z, m_{\mathcal{I} \setminus i} \mid m_i, \bar{Z}_m, X_m) dm_{\mathcal{I} \setminus i} \\
&\stackrel{(8.12)}{=} \eta \bar{bel}(m_i) \int_{m_{\mathcal{I} \setminus i}} p(z \mid m, \bar{Z}_m, X_m) \\
&\quad p(m_{\mathcal{I} \setminus i} \mid m_i, \bar{Z}_m, X_m) dm_{\mathcal{I} \setminus i} \\
&= \eta \bar{bel}(m_i) \int_{m_{\mathcal{I} \setminus i}} \prod_{j \in \mathcal{I}(r, x)} f(r_j, m_j, \delta_j) \\
&\quad \prod_{k \in \mathcal{I} \setminus \{i\}} \bar{bel}(m_k) dm_{\mathcal{I} \setminus i} \\
&= \eta_i \bar{bel}(m_i) f(r_i, m_i, \delta_i).
\end{aligned} \tag{8.13}$$

η and η_i are normalizing constants independent of m_i . To obtain (8.13), we make use of both (8.9) and (8.10). To transition from (8.13) to the last line, we pull $f(r_i, m_i, \delta_i)$ out of the integral and merge the remaining integral, which is independent of m_i , with the normalizer. \square

The update equation (8.11) in proposition 1 might look familiar: It is a generalization of the well-known map update $bel(m) = \eta \bar{bel}(m) p(z \mid x, m)$, which can be derived from Bayes rule in a straight-forward manner, see equation (7) in [52]. Another update equation which is related to (8.11) is the voxel-wise update for binary occupancy maps, see (18) in [52]. In contrast to the proposed update equation, the latter employs the inverse sensor model.

8.4.3 Closed-Form Map Posteriors

In this section, we leverage proposition 1 to derive the closed-form map posteriors for the reflection model and for the decay-rate model. For the reflection model, update equation (8.11) yields the following posterior over μ_i :

$$\begin{aligned}
bel(\mu_i) &\propto \prod_{Z_m} f(\mu_i, \delta_i) p(\mu_i) \\
&\stackrel{(8.3)}{\propto} \mu_i^{H_i} (1 - \mu_i)^{M_i} p(\mu_i) \\
&\propto \text{Beta}(H_i + 1, M_i + 1) p(\mu_i).
\end{aligned} \tag{8.14}$$

Here, $p(\mu_i)$ denotes the prior distribution over the map, H_i tells how many rays are reflected in v_i , and M_i is the number of rays that penetrate v_i without reflection. $\text{Beta}(\cdot)$ denotes a beta distribution.

If the prior is a beta distribution $p(\mu_i) = \text{Beta}(\alpha, \beta)$, which is the conjugate prior for the binomial distribution $f(\mu_i, \delta_i)$, equation (8.14) yields

$$\text{bel}(\mu_i) = \text{Beta}(H_i + \alpha, M_i + \beta). \quad (8.15)$$

The most likely reflection map can easily be derived from (8.15): Assuming a uniform prior distribution $p(\mu_i) = \text{Beta}(1, 1) = 1$ and computing the mode of the resulting beta posterior distribution yields the same result as formulated by Hähnel et al. [11] for maximum-likelihood reflection maps:

$$\mu_i^* = \frac{H_i}{H_i + M_i}. \quad (8.16)$$

For the decay-rate model, the update equation (8.11) becomes

$$\begin{aligned} \text{bel}(\lambda_i) &\propto \prod_{Z_m} f(r_i, \lambda_i, \delta_i) p(\lambda_i) \\ &\stackrel{(8.4)}{\propto} \lambda_i^{H_i} e^{-\lambda_i R_i} p(\lambda_i) \\ &\propto \text{Gamma}(H_i + 1, R_i) p(\lambda_i). \end{aligned}$$

$\text{Gamma}(\cdot)$ denotes a gamma distribution, $p(\lambda_i)$ is the prior map distribution, and R_i is the sum of the distances all rays travel within v_i .

If $p(\lambda_i)$ is a gamma distribution $\text{Gamma}(\alpha, \beta)$, which is the conjugate prior for the Poisson distribution and for the exponential distribution, we obtain the gamma-distributed belief ¹

$$\text{bel}(\lambda_i) = \text{Gamma}(H_i + \alpha, R_i + \beta). \quad (8.17)$$

Setting $\alpha = 1$ and $\beta = 0$ leads to the so-called uninformative prior. Plugging this prior into (8.17) and computing the mode of the resulting posterior leads to the decay rates of the maximum-likelihood approach as given in [29]:

$$\lambda_i^* = \frac{H_i}{R_i}. \quad (8.18)$$

For both the reflection model and the decay-rate model, the parameters α and β of the prior distribution need to be estimated during mapping. In section 8.5, we explain how we obtain the prior parameters used throughout the experiments.

¹In the context of height maps, Marks et al. [46] also obtain gamma-shaped posteriors over grid values.

8.4.4 Localization with Map Posteriors

In this section, we formulate robot localization on the basis of the full posterior over the map rather than on the basis of a fixed map. In contrast to the well-known approach [10] which computes the belief over the robot pose on the basis of the given map m as

$$bel_m(x) = \eta \overline{bel}(x) p(z | x, m), \quad (8.19)$$

we leverage the full posterior $bel(m) = p(m | X_m, Z_m)$ instead of m . Consequently, we derive the pose belief as follows:

$$\begin{aligned} bel(x) &= p(x | Z, Z_m, X_m) \\ &= \eta p(z | x, \overline{Z}, Z_m, X_m) \underbrace{p(x | \overline{Z}, Z_m, X_m)}_{=: \overline{bel}(x)} \\ &= \eta \overline{bel}(x) \int p(z | x, m) p(m | \overline{Z}, Z_m, X_m) dm \\ &= \eta \overline{bel}(x) \underbrace{\int p(z | x, m) bel(m) dm}_{=: L(z, x)}. \end{aligned} \quad (8.20)$$

Here, X_m and Z_m are the poses and measurements recorded during the mapping process, respectively, Z denotes the measurements recorded during localization, and $\overline{Z} = Z \setminus \{z\}$ is the set of all measurements but the most recent one.

Equations (8.19) and (8.20) differ only in the last term, which, in (8.19), is called the measurement likelihood. Analogously, we call $L(z, x)$ in (8.20) the measurement likelihood based on the map posterior. The next section presents closed-form solutions of $L(z, x)$ for the reflection model and the decay-rate model.

8.4.5 Closed-Form Measurement Likelihoods

For two particular factorizing sensor models, the reflection model and the decay-rate model, the solution of the integral contained in the measurement likelihood $L(z, x)$ leads to closed-form expressions. To compute those, we

need to factorize $L(z, x)$ first:

$$\begin{aligned}
L(z, x) &= \int_m p(z \mid x, m) \text{bel}(m) \, dm \\
&= \int_m \prod_{i \in \mathcal{I}(r, x)} f(r_i, m_i, \delta_i) \prod_{j=1}^N \text{bel}(m_j) \, dm \\
&= \int_{m_{\mathcal{I}(r, x)}} \prod_{i \in \mathcal{I}(r, x)} f(r_i, m_i, \delta_i) \prod_{j \in \mathcal{I}(r, x)} \text{bel}(m_j) \, dm_{\mathcal{I}(r, x)} \\
&= \prod_{i \in \mathcal{I}(r, x)} l(r_i, \delta_i)
\end{aligned}$$

with

$$l(r_i, \delta_i) := \int f(r_i, m_i, \delta_i) \text{bel}(m_i) \, dm_i. \quad (8.21)$$

Now, we evaluate this integral for both sensor models. For the reflection model with beta-shaped prior $p(\mu_i) = \text{Beta}(\alpha, \beta)$, we take the distribution $f(\mu_i, \delta_i)$ from (8.3) and the posterior from (8.15), plug them into (8.21), and solve the integral:

$$l_{\text{ref}}(r_i, \delta_i) = l_{\text{ref}}(\delta_i) = \frac{(H_i + \alpha)^{\delta_i} (M_i + \beta)^{1-\delta_i}}{H_i + \alpha + M_i + \beta}. \quad (8.22)$$

If the posterior $\text{bel}(\mu_i)$ is set to a Dirac delta distribution $\text{Delta}(\mu_i - \mu_i^*)$, equation (8.21) reproduces the measurement likelihood based on the most likely map, as derived in [11]:

$$l_{\text{ref}}(\delta_i) = f(\delta_i) = \frac{H_i^{\delta_i} M_i^{1-\delta_i}}{H_i + M_i}. \quad (8.23)$$

Equation (8.23) is only valid for $H_i + M_i > 0$. If no ray has visited the voxel during mapping, the maximum-likelihood approach has to assign some initial value. In contrast, (8.22) is also valid for voxels that have not been visited by any ray during mapping.

Note that in the special case of $\alpha = \beta$, equation (8.23) can be transformed into (8.22) using Laplace smoothing – see for example equation (13) in [53].

In order to obtain closed-form solutions for the decay-rate model, we assume a gamma-shaped prior $p(\lambda_i) = \text{Gamma}(\alpha, \beta)$ and plug (8.4) and (8.17) into (8.21). This leads to

$$l_{\text{dec}}(r_i, \delta_i) = \left(\frac{R_i + \beta}{R_i + \beta + r_i} \right)^{H_i + \alpha} \left(\frac{H_i + \alpha}{R_i + \beta + r_i} \right)^{\delta_i}.$$

Analogously to the reflection model, the measurement likelihood based on the most likely map as derived in [29] can be reproduced from (8.21) by setting $bel(\lambda_i) = \text{Delta}(\lambda_i - \lambda_i^*)$:

$$l_{\text{dec}}(r_i, \delta_i) = f(r_i, \delta_i) = e^{-\frac{H_i}{R_i} r_i} \left(\frac{H_i}{R_i} \right)^{\delta_i}.$$

Until now, we have assumed that the lidar sensor reports a reflection for each emitted beam. In practice, however, the sensor range is always limited by a lower bound r_{\min} and an upper bound r_{\max} . The following equations show for both measurement models how to attribute probabilities to these out-of-range measurements:

$$P(r < r_{\min} \mid x, Z_m, X_m) = 1 - \prod_{i \in \mathcal{I}(r_{\min}, x)} l(r_i, \delta_i = 0),$$

$$P(r > r_{\max} \mid x, Z_m, X_m) = \prod_{i \in \mathcal{I}(r_{\max}, x)} l(r_i, \delta_i = 0).$$

8.5 Experiments

In the previous section, we have derived all the necessary equations to compute the posterior over a grid map and to leverage this posterior for localization. Now, to demonstrate that localization benefits from employing the full map posterior instead of the most likely map, we perform a simulation and extensive real-world experiments.

As shown in section 8.4.3, the computation of the map posterior for the reflection model and for the decay-rate model requires the specification of the two parameters α and β of the prior distribution. In our experiments, we determine them as follows. First, we compute the empirical mean and variance of the map value over all voxels. Then, we choose α and β such that the mean and variance of the parameterized distributions equal these empirical values. To that end, we solve the known equations for the mean and the variance of the beta and gamma distribution for α and β . For the reflection model, we obtain

$$\alpha = -\frac{\text{E}[\mu] \left(\text{E}[\mu]^2 - \text{E}[\mu] + \text{var}[\mu] \right)}{\text{var}[\mu]},$$

$$\beta = \frac{\text{E}[\mu] - \text{var}[\mu] + \text{E}[\mu] \text{var}[\mu] - 2 \text{E}[\mu]^2 + \text{E}[\mu]^3}{\text{var}[\mu]}.$$

For the decay-rate model, the parameters are

$$\alpha = \frac{\mathbf{E}[\lambda]^2}{\text{var}[\lambda]},$$

$$\beta = \frac{\mathbf{E}[\lambda]}{\text{var}[\lambda]}.$$

To strictly avoid the repeated use of the same information, one has to estimate these parameters for each voxel separately, computing the empirical mean and variance over all voxels but the considered one. However, the sheer number of observations in our datasets render this effect negligible. Hence, it is sufficient to compute α and β only once.

Throughout the experiments section, MLM denotes the maximum-likelihood approach, while FMP refers to the proposed approach, which leverages the full map posterior. REF and DEC denote the reflection model and the decay-rate model, respectively.

8.5.1 Localization in Simulation

We simulate the following scenario to evaluate the localization performance with both maximum-likelihood maps and full posteriors: A mobile robot is located in a corridor that is modeled by a row of $N = 100$ consecutive voxels. Each experiment run consists of two phases: During the mapping phase, the robot visits every voxel n times and for each voxel collects the measurements H_i , M_i , and R_i , which it uses to build both a reflection map and a decay-rate map. In the localization phase, which consists of 100 iterations, the robot traverses the corridor from start to end knowing its motor commands. In each iteration, the robot moves to the next voxel, fires its sensor once, and updates the belief over its pose. Therefor, with the maximum-likelihood approach it uses (8.19), with the proposed approach it uses (8.20). After the robot has arrived at the last voxel of the corridor, we evaluate the pose belief at the true pose averaged over each iteration:

$$\rho := \mathbf{E}(\text{bel}(x_{\text{true}})).$$

We perform 10,000 such localization runs for each $n \in \{1, 2, 3, 4, 5, 10, 20, 50, 100, 200\}$. In each run, we synthesize a new map m by drawing samples from a uniform distribution for the reflection model or from a gamma distribution $\text{Gamma}(1, 1)$ for the decay-rate model. This map is hidden from the localization algorithms. We use m to simulate the measurements the robot records during mapping and localization by sampling from the distribution described in equation (8.3) for the reflection model or from (8.4) for

the decay-rate model. The belief over the robot pose is initialized with a uniform distribution over x .

For both measurement models, we compare three algorithms: MLM, FMP with uniform prior, and FMP with conjugate prior. The results in figure 8.2 indicate that the proposed method – FMP with conjugate prior – yields higher localization accuracy than the compared methods for both measurement models. Moreover, we observe that the benefit of the proposed method is greater for the reflection model than for the decay-rate model. We perform a one-tailed, paired-sample t-test, which validates these observations: For the reflection model, the proposed method outperforms the other two approaches for $n \leq 100$ with a probability greater than 0.9999. For the decay-rate model, we obtain the same significance level for $n \leq 4$.

8.5.2 Real-World Localization

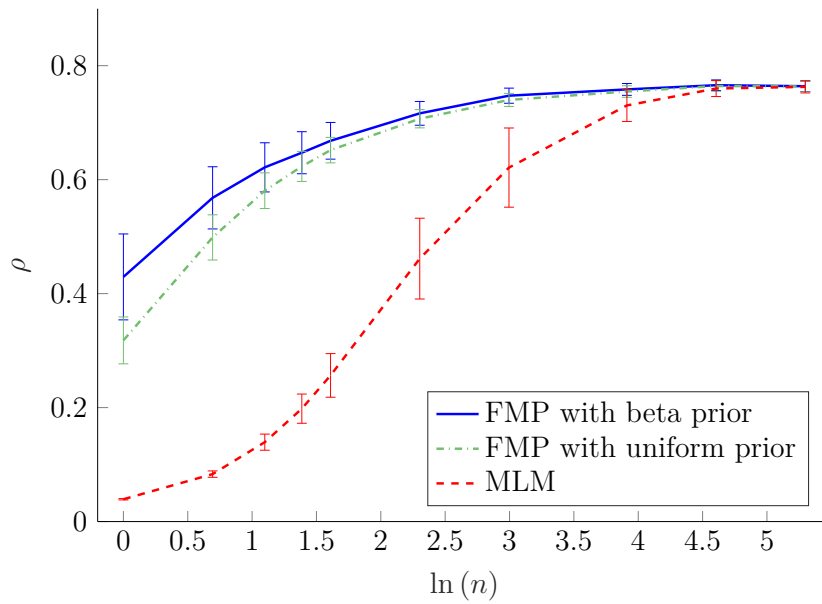
In order to validate the findings of the simulation in the real world, we test our approach on three datasets in which our mobile off-road robot navigates different kinds of environments: In the campus dataset, the robot drives around on the campus of the University of Freiburg. The forest dataset, a section of which is shown in figure 8.1, was recorded on a small trail in the middle of a forest, while the park dataset contains a long trajectory along a broad road through an open forest. We tessellate the environment into cubic axis-aligned voxels with edge length 0.5 m. Consequently, the campus maps contain $444 \times 406 \times 43$ voxels, the park maps contain $515 \times 561 \times 41$ voxels, and the forest maps contain $393 \times 403 \times 86$ voxels.

We use the following hardware: The off-road robot VIONA by Robot Makers carries a Velodyne HDL-64E lidar sensor and an Applanix POS LV localization system, which provides a centimeter-accurate estimate of the robot pose by fusing the data from multiple GPS sensors, an IMU, and odometry. Due to its accuracy, we use the Applanix data as pose ground truth during mapping and in the evaluation of all localization estimates.

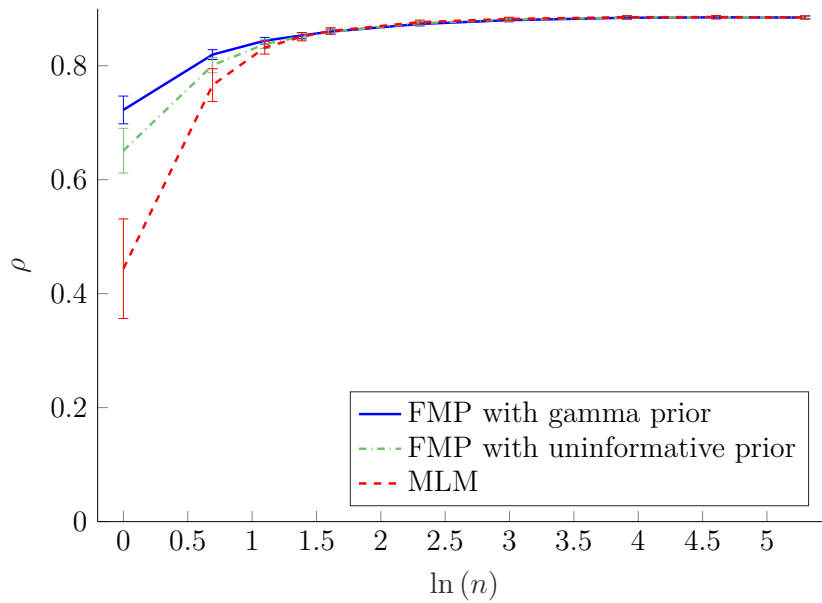
To assess the localization performance using full posteriors versus maximum-likelihood maps, we employ different measures:

Measurement Likelihood

We employ the logarithm of the measurement likelihood at the true pose as a measure of how well the approaches predict the measurements given the true pose. For MLM, the measurement likelihood is defined as $p(z | x, m)$. For FMP, it is defined by $L(z, x)$ in (8.20). We sum up the likelihood values of all measurements in a dataset and divide the result for MLM by the result for



(a) Reflection model.



(b) Decay-rate model.

Figure 8.2: Localization accuracy in a simulated environment. n is the number of observations per voxel during the mapping process. The data points correspond to $n \in \{1, 2, 3, 4, 5, 10, 20, 50, 100, 200\}$. The accuracy measure ρ shows the average probability which the algorithm assigns to the ground truth position. The error bars represent the variances over 10 000 runs.

	REF	DEC
Campus	1.21	1.16
Forest	1.22	1.31
Park	1.25	1.27

Table 8.2: Log-likelihood ratios of the measurements given the ground-truth pose. For each dataset, we show the ratio of the cumulated measurement log-likelihoods for MLM to the ones for FMP. Values greater than one mean that FMP attributes higher likelihoods to measurements given the true robot pose.

FMP. The ratios are presented in table 8.2. For both measurement models and all three datasets, these values are greater than one, which means that FMP achieves better prediction of the measurements than MLM.

Kullback-Leibler Divergence

As a measure of dissimilarity, we employ the Kullback-Leibler (KL) divergence from the estimated position distribution p to the ground-truth pose distribution p_{gt} :

$$D_{KL}(p_{gt}(x) \parallel p(x)) = \int p_{gt}(x) \log \left(\frac{p_{gt}(x)}{p(x)} \right) dx. \quad (8.24)$$

We assume that p_{gt} is a bivariate normal distribution with standard deviation $\sigma_x = \sigma_y = 0.05 m$, chosen according to the specifications of the Applanix localization system. The formula for the pose likelihood p depends on the approach:

$$p = \begin{cases} p(x | z, m) \propto p(z | x, m) & \text{for MLM} \\ p(x | z, X_m, Z_m) \propto L(z, x) & \text{for FMP} \end{cases} \quad (8.25)$$

To evaluate equation (8.24), we perform a Monte-Carlo integration over x :

$$D_{KL}(p_{gt} \parallel p) \approx \sum \log \left(\frac{p_{gt}}{p} \right),$$

where p_{gt} and p are normalized over the samples. We compute the results shown in table 8.3 by summing up the approximated D_{KL} values for all measurements in one dataset and dividing the result for MLM by the result of

	REF	DEC
Campus	1.24	1.25
Forest	1.27	1.82
Park	1.37	1.59

Table 8.3: Ratios of the Kullback-Leibler divergence from the position distribution estimated by MLM and FMP to the ground truth distribution. For each dataset, we show the ratio of the cumulated KL divergences for the MLM approach to the corresponding value for the FMP approach. Ratios greater than one indicate that the distribution estimated by FMP is closer to the ground-truth than the one estimated by MLM.

FMP. All ratios are greater than one, which implies that the distribution estimated by FMP is closer to the ground truth than the distribution estimated by MLM. Moreover, all ratios are higher than the corresponding ratios of the measurement likelihoods in table 8.2.

Monte-Carlo Localization

The results of the two aforementioned measures are entirely reproducible, in the sense that they are independent of localization algorithm design. In the corresponding experiments, the proposed approach always yields better results than the maximum likely approach. To demonstrate the consequences of this behavior in a real application, we evaluate the localization accuracy of two particle filters. Each filter employs 3000 particles to localize the off-road robot in six dimensions in the park scenario with the decay-rate model. The initial variance is 0.1 m in the translational dimensions and 0.1 rad in the rotational dimensions. The filters only differ in the method used to weight the particles: One uses the likelihood derived from the most likely map, the other uses the likelihood derived from the full map posterior. Figure 8.3 shows the corresponding localization errors averaged over ten runs.

8.6 Conclusion and Future Work

In this paper, we present an approach to compute posterior distributions over real-valued grid maps from lidar observations. We demonstrate that for the well-established reflection sensor model and the recently introduced decay-rate sensor model, the posterior distributions can be represented in closed form. Our approach requires the same measurement information and has

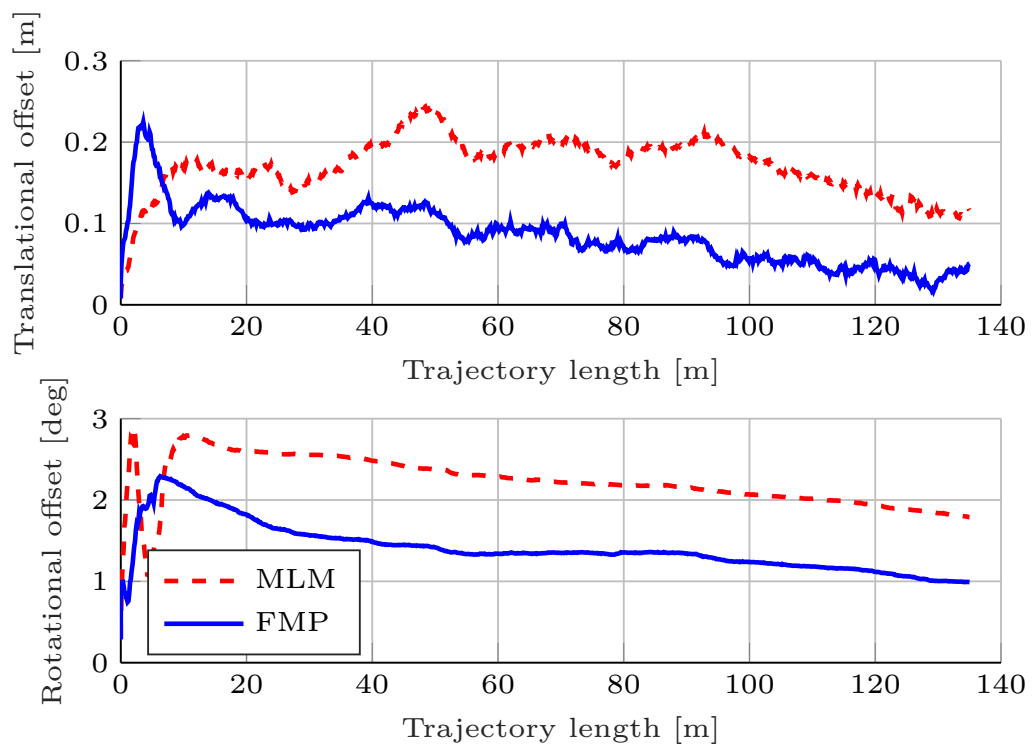


Figure 8.3: Localization accuracy with the decay-rate model on the park dataset averaged over ten runs.

the same computational demands as approaches which only determine the mode of the distribution. Simulations and extensive real-world experiments show that taking into account the full map posterior improves the accuracy of robot localization.

In the future, we plan to relax the assumption that all map cells are independent by accounting for measurement noise. Moreover, we will embed the presented approach into a SLAM framework, which we will then use to localize our off-road robot during operation.

Chapter 9

DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform

The original research article presenting DCT maps [2] was authored by Alexander Schaefer, Lukas Luft, and Wolfram Burgard and published in the journal *IEEE Robotics and Automation Letters*, Volume 3, Issue 2, April 2018. We presented this work during the IEEE/RSJ International Conference on Robotics and Automation 2018, which took place in Brisbane, Queensland, Australia, from May 21 to May 25, 2018. The IEEE holds the copyright on the article: © 2018 IEEE. Reprinted, with permission, from “DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform”.

9.1 Abstract

Most robot mapping techniques for lidar sensors tessellate the environment into pixels or voxels and assume uniformity of the environment within them. Although intuitive, this representation entails disadvantages: The resulting grid maps exhibit aliasing effects and are not differentiable. In the present paper, we address these drawbacks by introducing a novel mapping technique that does neither rely on tessellation nor on the assumption of piecewise uniformity of the space, without increasing memory requirements. Instead of representing the map in the position domain, we store the map parameters in the discrete frequency domain and leverage the continuous extension of the inverse discrete cosine transform to convert them to a continuously differentiable scalar field in the position domain, which we call DCT map. A DCT map assigns to each point in space a lidar decay rate, which models the local

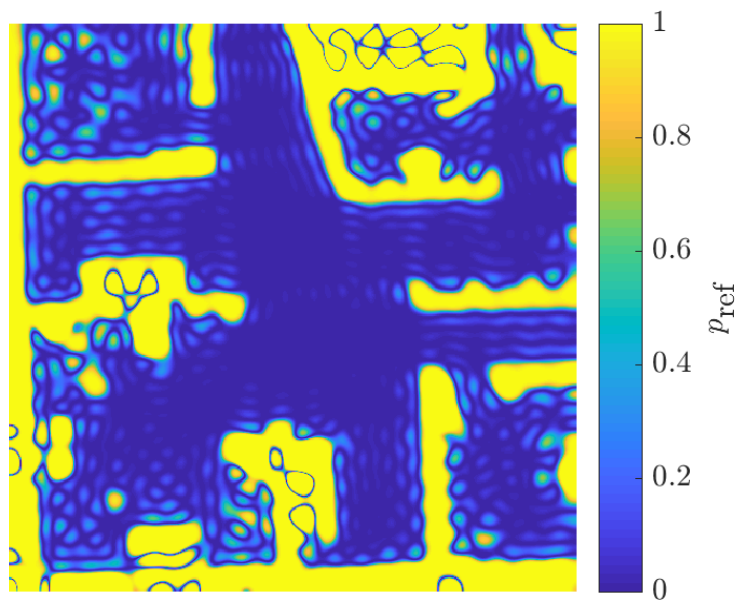
permeability of the space for laser rays. In this way, the map can describe objects of different laser permeabilities, from completely opaque to completely transparent. DCT maps represent lidar measurements significantly more accurate than grid maps, Gaussian process occupancy maps, and Hilbert maps, all with the same memory requirements, as demonstrated in our real-world experiments.

9.2 Introduction

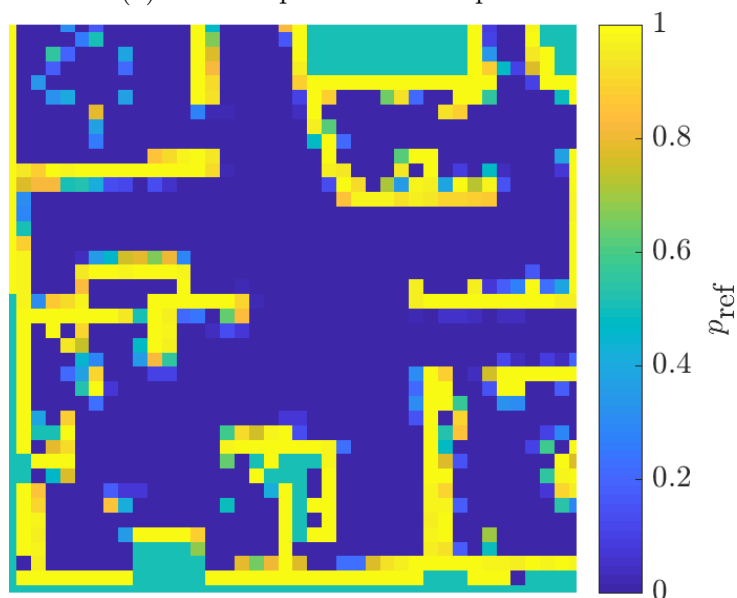
Mapping and localization are at the heart of almost every mobile robotic system. In this context, lidar is a popular sensor modality, as lidar sensors produce relatively accurate, low-noise signals. Using these signals for mapping and localization requires an inverse and a forward sensor model. The inverse sensor model converts recorded measurements to a map. The forward model uses this map to assess the probability of incoming sensor readings given the robot pose. The maps produced by the inverse pass are often grid maps: They tessellate the physical space into square pixels or cubic voxels. Each pixel or voxel contains a value that is assumed to be constant within it. This value characterizes the statistical optical properties of the corresponding portion of space. Fig. 9.1b shows an example of such a grid map built from 2-D lidar scans recorded in an office environment.

Although tessellation is intuitive, grid maps bring with them several drawbacks. First, they can only coarsely approximate the true spatial distribution of the optical properties of interest. Aliasing effects occur whenever the optical characteristics of the environment change, as these transitions are never perfectly aligned with the raster of the grid. The grid map in fig. 9.1b exhibits the resulting characteristic staircase patterns. Although increasing the map resolution can theoretically alleviate this problem (see fig. 9.1c), quadratic or cubic memory complexity quickly renders this approach prohibitive. Depending on the use case, non-cubic voxels may mitigate the errors induced by tessellation [55]. Second, grid maps are not continuously differentiable, although this is a desirable property of any map. Continuous differentiability would allow to localize the robot by maximizing the measurement likelihood over the robot poses, and even to perform SLAM by maximizing the measurement likelihood over the robot poses and the map parameters.

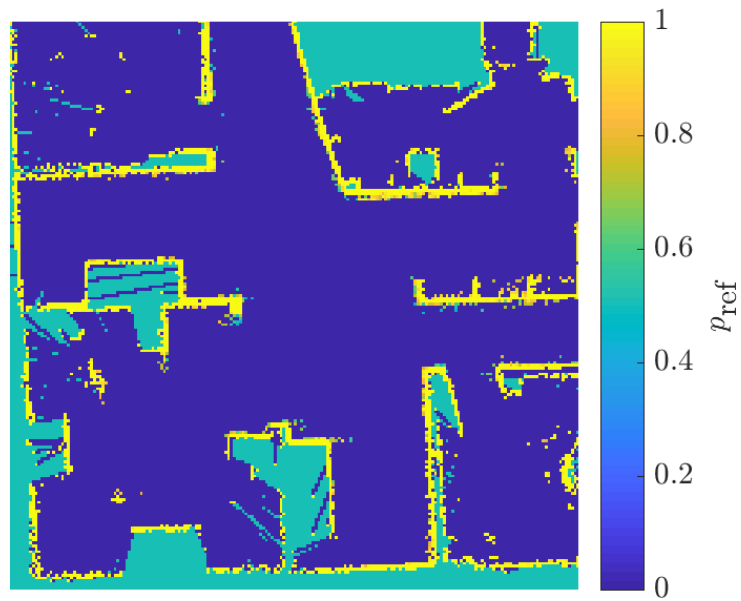
In the present paper, we choose a different approach to avoid the aforementioned detrimental effects without increasing the memory demands. Inspired by well-established digital image compression algorithms like JPEG, we store the map parameters in the discrete frequency domain and use the so-called continuous extension of the inverse discrete cosine transform [18] to



(a) DCT map with 40×40 parameters.



(b) Grid map composed of 40×40 pixels with edge length 25 cm.



(c) Grid map composed of 200×200 pixels with edge length 5 cm.

Figure 9.1: Decay-rate maps of the same $10 \text{ m} \times 10 \text{ m}$ patch of the Intel Research Lab dataset [54] generated from the identical set of planar lidar measurements. The colors encode the reflection probability $p_{\text{ref}} := 1 - \exp(-\lambda)$, where λ denotes the local laser decay rate.

obtain a continuously differentiable scalar field in the position domain. In addition to the regular inverse discrete cosine transform, its continuous extension not only computes the function values at discrete grid points in the spatial domain, but also closely approximates them in between. We combine this map representation with the recently developed decay-rate model for lidar sensors [29]. The resulting DCT maps model the local permeability of the space for laser rays. Fig. 9.1a depicts such a DCT map. It was built from the identical information as the grid map in fig. 9.1b and has the same memory footprint, but it does not exhibit staircase patterns and better preserves the map contours. Indeed, our experiments show that DCT maps represent lidar data with higher accuracy than other approaches. Moreover, the continuous derivatives of DCT maps can be calculated in closed form, a fact that enables optimization-based SLAM.

In the following, we first survey different map representations. Then, we describe the mathematics behind DCT maps in detail. Finally, the findings of experiments conducted with publicly available real-world 2-D lidar datasets are presented.

9.3 Related Work

Occupancy grid maps [56] were among the first probabilistic map representations used in robotics and are still widely used today. They tessellate the space into independent cells and assign each cell the posterior probability of being occupied. Occupancy grid maps cannot model semi-transparent objects; they assume that each cell is either completely free or completely occupied. In contrast, the decay-rate model, which we employ to formulate DCT maps, explicitly models the permeability of each cell for a laser ray. If used in conjunction with grid maps, it even allows to calculate posterior distributions over the decay-rate values without additional computational effort [45].

While 2-D occupancy maps are able to model large areas, even moderately-sized 3-D occupancy grids quickly outgrow the memory limitations of modern computers. For this reason, several research projects target compressed map representations. Elevation maps [57] assume that the environment can be represented by a 2-D grid map whose cells contain not only occupancy values, but also one height coordinate per cell. To relax the assumption that the world is a single surface, [58] extends elevation maps to multi-level surface maps. Multi-volume occupancy grids [59] manage volumetric data as 2-D arrays, too, but in contrast to multi-level surface maps, each cell contains a list of occupied height regions and one of free height

regions. Octrees [60] approach the memory limitation problem by hierarchically partitioning the space using an octal tree data structure. They have found broad application in robotics to model the spatial distribution of the occupancy value [61, 62, 63]. The authors of [64] present an octree-based data structure that is efficient to update and to copy, so it can be used in particle filter-based SLAM, where hundreds of maps must be maintained in parallel. To model the dynamics of the environment, [65] assumes that the occupancy values in an octree are subject to periodic changes. For each cell, the authors record the occupancy value over time and transform the resulting function to the frequency domain to predict the occupancy value at a later point in time. In this way, they achieve high compression ratios compared to storing one occupancy map per time step. Multi-resolution occupied voxel lists [66] differ from traditional occupancy mapping in that they store only the positions of the voxels that have been observed more frequently as occupied than as free. They are neither able to differentiate between unoccupied and unknown volumes, nor to account for semi-transparent voxels.

The normal distributions transform [14] was initially conceived in the context of scan matching. Based on this work, [67] introduces the so-called normal distributions transform occupancy map. Basically, this map is a grid map, but instead of a single scalar, every cell contains a normally distributed occupancy probability density function, which is cropped at the voxel bounds. In this way, it drops the assumption that the space is uniform within each voxel. As opposed to DCT maps, however, normal distributions occupancy maps achieve higher accuracy at the cost of increased memory consumption. Like all other occupancy-based approaches, they are not able to model semi-permeable objects, either. Normal distributions occupancy maps are extended and advanced in [68, 69, 70].

Other approaches completely abandon the notion of voxels. For example, [71] uses Haar wavelets to represent 3-D occupancy data. The authors of [72] drop the restriction that the elementary volumes of a map shall fill the space without gaps. Instead, they model the environment by non-overlapping spheres of equal sizes. In this way, they are able to more closely represent curved surfaces.

Point clouds are a simple and convenient way to represent lidar sensor data. However, in contrast to occupancy maps or decay-rate maps, they are lossy in the sense that they store only the endpoints of the rays. They discard the ray path information of both reflected rays and rays that are not reflected. When point clouds are used for mapping, they accumulate memory for every incoming measurement, which limits their suitability for long-term navigation. Despite their drawbacks, many SLAM systems [73, 74, 75] represent lidar data in the form of point clouds.

In object reconstruction in computer graphics, objects are modeled as line segments in 2-D [76] or as polygon meshes in 3-D [77]. The resulting models can achieve an astonishing level of detail [78]. However, similar to mapping approaches based on implicit shape potentials like KinectFusion [79], they are not perfectly suited for lidar-based robot localization due to their sheer memory footprint and their inability to deal with semi-transparent materials.

Recently, machine learning techniques have completely relaxed the independence assumption between grid cells and produce continuous occupancy maps. Gaussian process occupancy maps (GPOM), for example, learn the environment of a robot and predict future states [80, 15]. Building on the latter, the authors of [1] present an incremental GPOM formulation that enables online mapping. Gaussian processes have also been applied to other map representations like implicit shape potentials [81]. Hilbert maps [82] are continuous occupancy maps built by projecting the lidar measurements in a Hilbert space, learning a logistic regression classifier, and then classifying each point in space as free or occupied.

9.4 Approach

In this section, we shortly revisit how the decay-rate model computes measurement probabilities conditioned on any kind of map, then we define the map using the continuous extension of the inverse discrete cosine transform. With these prerequisites, we derive the forward model, which computes the probability of a lidar measurement given the spectral parameters of the DCT map. In the last step, we formulate the inverse model as an optimization problem: We estimate the map parameters by maximizing the joint likelihood of all measurements collected during mapping.

For brevity and without loss of generality, the following derivation is performed for 2-D space. The derivation of the forward and inverse sensor model in 3-D exactly parallels the 2-D case.

9.4.1 The Decay-Rate Model

The decay-rate model [29] models the probability that a lidar ray traverses a uniform medium as exponential decay process. The corresponding map assigns a decay rate to each point in space. This decay rate is a non-negative real number that describes the interaction between the laser ray and the environment completely.

To formulate the forward model mathematically, we introduce the following definitions. A lidar measurement $z := \{s, v, r\}$ describes a ray that

originates at the sensor position s , travels in direction v , and ends after having traveled distance r . Assuming that the sensor provides its true position s , the true ray direction v , and that we are given a specific map \mathcal{M} , we model the non-deterministic interaction between the ray and the environment by the measurement probability density with respect to the radius

$$p(r) := p(r \mid s, v, \mathcal{M}). \quad (9.1)$$

Consequently, the absolute probability that the ray covers at least distance r is

$$\mathcal{N}(r) := 1 - \int_0^r p(r') dr'. \quad (9.2)$$

Alternatively, we can express equation (9.2) in form of the differential equation

$$p(r) = -\frac{d\mathcal{N}(r)}{dr}. \quad (9.3)$$

The essential idea of the decay-rate model consists in the assumption that $\mathcal{N}(r)$ obeys an exponential decay process

$$\frac{d\mathcal{N}(r)}{dr} = -\lambda(r) \mathcal{N}(r), \quad (9.4)$$

where $\lambda(r)$ denotes the decay rate at a specific radius r along the ray. By combining this model assumption with differential equation (9.3), we obtain the measurement probability density

$$p(r) = \lambda(r) \mathcal{N}(r). \quad (9.5)$$

In (9.4) and (9.5), $\lambda(r)$ is obtained by evaluating the map $\lambda(x, y)$ along the trajectory of the ray.

The above formulation of the decay-rate model is independent of any specific map representation. To use it as forward model in combination with DCT maps, we need to define the map function $\lambda(r)$ and solve the differential equation. In order to do so, we describe the spatial representation of DCT maps in the next section in detail. After that, we have all prerequisites at hand to solve the differential equation. The solution enables us to express the measurement probability of a lidar measurement given the map in closed form.

9.4.2 Transforming the Spectral Map Representation to the Spatial Domain

To avoid the disadvantages related to tessellation, DCT maps represent the map parameters in the discrete frequency domain instead of the position domain. Calculating the measurement likelihood from such a representation requires the definition of the transformation from the frequency domain to the spatial domain. We employ the continuous extension of the inverse discrete cosine transform (CEIDCT) [18]. Like other continuous extensions of Fourier-related transforms, it converts a discrete signal in the frequency domain to a continuous signal in the spatial domain. However, it differs from its relatives in that the continuous signal converges to the continuous function from which it was sampled for an increasing number of parameters (see [18], pp. 11–12). Moreover, its parameters are purely real-valued. For these reasons, it is particularly suited for our use case.

If we assume the spectral map parameters to be given by a matrix \mathcal{A} with L rows and M columns, and if we denote the elements of \mathcal{A} by a_{lm} with $l \in \{0, 1, \dots, L-1\}$ and $m \in \{0, 1, \dots, M-1\}$, the CEIDCT transforms them to a continuously differentiable decay-rate map defined for each point (x, y) in the spatial domain

$$\begin{aligned}
 \lambda(x, y) &= \left(\sum_{l=0}^{L-1} \sum_{m=0}^{M-1} a_{lm} \cos(l\tilde{x}) \cos(m\tilde{y}) \right)^2 & (9.6) \\
 &= \left(\sum_{i=0}^{I-1} a_i \cos(l_i \tilde{x}) \cos(m_i \tilde{y}) \right)^2 \\
 &= \sum_{i=0}^{I-1} \sum_{j=0}^{I-1} a_i \cos(l_i \tilde{x}) \cos(m_i \tilde{y}) a_j \cos(l_j \tilde{x}) \cos(m_j \tilde{y}) \\
 &= \frac{1}{8} \sum_{i=0}^{I-1} \sum_{j=0}^{I-1} a_i a_j \sum_{\alpha \in Q} \sum_{\beta \in Q} \sum_{\gamma \in Q} \left((l_i + \alpha l_j) \tilde{x} + \beta(m_i + \gamma m_j) \tilde{y} \right) & (9.7)
 \end{aligned}$$

with $I := LM$ and $Q := \{-1, +1\}$. The tildes denote the π -normalization of the map coordinates: $\tilde{x} := \frac{\pi x}{X}$, $\tilde{y} := \frac{\pi y}{Y}$, where X and Y indicate the extent of the map. The variables l_i and m_i are the row and column indices into the matrix \mathcal{A} that correspond to its i -th element a_i .

The original formulation of the CEIDCT does not square the double sum in (9.6). We employ this variant, however, because it ensures that the decay-rate is non-negative for every point in the map. Negative decay rates would cause problems, as we cannot interpret the negative measurement probabilities in which they might result.

To solve equation (9.4), we still need to transition from $\lambda(x, y)$ to $\lambda(r) := \lambda(r, s, v)$. To that end, we apply the ray equation $[x, y]^T = s + v r$ to (9.7) and obtain

$$\lambda(r) = \frac{1}{8} \sum_{i=0}^{I-1} \sum_{j=0}^{I-1} a_i a_j \sum_{\alpha \in Q} \sum_{\beta \in Q} \sum_{\gamma \in Q} \cos \left((l_i + \alpha l_j) [\tilde{s}_x + \tilde{v}_x r] + \beta (m_i + \gamma m_j) [\tilde{s}_y + \tilde{v}_y r] \right). \quad (9.8)$$

9.4.3 Computing the Measurement Likelihood

Now we express the measurement probability of a lidar ray as a function of the measurement z and the spectral representation of the map \mathcal{A} by solving the differential equation (9.4). The solution is

$$\mathcal{N}(r) = \exp \left\{ -\mathcal{S}(s, v, r) \right\} \quad (9.9)$$

with

$$\mathcal{S}(s, v, r) = \int_0^r \lambda(r') dr' = \frac{1}{8} \sum_{i=0}^{I-1} \sum_{j=0}^{I-1} a_i a_j \sum_{\alpha \in Q} \sum_{\beta \in Q} \sum_{\gamma \in Q} A_{ij}$$

where

$$A_{ij} := A(i, j, \alpha, \beta, \gamma) = \begin{cases} \frac{[\sin((l_i + \alpha l_j) [\tilde{s}_x + \tilde{v}_x r'] + \beta (m_i + \gamma m_j) [\tilde{s}_y + \tilde{v}_y r'])]_0^r}{(l_i + \alpha l_j) \tilde{v}_x + \beta (m_i + \gamma m_j) \tilde{v}_y}, & \text{if } (l_i + \alpha l_j) \tilde{v}_x + \beta (m_i + \gamma m_j) \tilde{v}_y \neq 0 \\ r \cos((l_i + \alpha l_j) \tilde{s}_x + \beta (m_i + \gamma m_j) \tilde{s}_y), & \text{if } (l_i + \alpha l_j) \tilde{v}_x + \beta (m_i + \gamma m_j) \tilde{v}_y = 0 \end{cases} \quad (9.10)$$

Note that out of the infinite number of solutions to (9.4), we chose the one that satisfies the boundary condition $\mathcal{N}(0) \stackrel{!}{=} 1$.

By plugging equations (9.8) and (9.9) in (9.5), we finally obtain the closed-form solution of the measurement likelihood $p(r)$ for rays with real-valued radius r .

Not all lidar measurements are real-valued, though. In practice, the range of every lidar scanner is limited to a finite interval $[r_{\min}, r_{\max}]$. We call the rays reflected outside this interval no-return rays. In the following, we assume that the sensor identifies rays falling below r_{\min} by the tag sub and rays that exceed r_{\max} by the tag super. Consequently, the

space of all possible measurements r is the mixed discrete-continuous set $\mathcal{D} := \{\text{sub}, \text{super}, r' : r' \in [r_{\min}, r_{\max}]\}$.

Fortunately, the decay-rate model easily accommodates both sorts of no-return rays:

$$P(\text{sub}) = \int_0^{r_{\min}} p(r) dr = 1 - \mathcal{N}(r_{\min}), \quad (9.11)$$

$$P(\text{super}) = \int_{r_{\max}}^{\infty} p(r) dr = \mathcal{N}(r_{\max}). \quad (9.12)$$

Supporting no-return rays is an important feature of the model. In a typical outdoor setting, no-return rays represent a considerable fraction of all measurements. If the model is unable to incorporate the information they convey, which is the case for the endpoint model, for example, it will inevitably lose accuracy.

During mapping and localization, one does not need to evaluate the measurement probability of a single ray, but of a whole laser scan $Z := \{z_k\}$ consisting of K rays, both with real-valued radius and without detected reflection. To obtain this probability, we first formulate the probability density function for each ray over the mixed space \mathcal{D} by combining equations (9.5), (9.11), and (9.12) to

$$p(z | \mathcal{M}) := \begin{cases} P(\text{sub}), & \text{if } r = \text{sub} \\ p(r), & \text{if } r \in [r_{\min}, r_{\max}] \\ P(\text{super}), & \text{if } r = \text{super} \end{cases}$$

The above result, which we call a mixed probability density, is positive, real, and integrates to unity. Using the independence assumption, we then compute the joint probability density of all rays as the product

$$p(Z | \mathcal{M}) = \prod_{k=0}^{K-1} p(z_k | \mathcal{M}).$$

9.4.4 Building the Decay-Rate Map

During the inverse pass, we want to determine the map parameters \mathcal{A} that best explain the lidar measurements collected in the mapping run:

$$\mathcal{A} = \operatorname{argmax}_{\mathcal{A}} p(Z | \mathcal{A}) = \operatorname{argmax}_{\mathcal{A}} \log \left\{ p(Z | \mathcal{A}) \right\}.$$

This non-linear multivariate optimization problem can be solved by iterative computational optimization techniques like stochastic gradient descent or

trust-region methods. These methods work considerably more reliable and faster when provided with first-order and second-order analytical logarithmic derivatives with respect to the spectral map parameters. To calculate the derivatives, we introduce the following shortcut notation:

$$\frac{\partial \lambda(x, y)}{\partial a_i} =: \sum_{j=0}^{I-1} a_j B_{ij} =: B_i,$$

with

$$B_{ij} := 2 \cos(l_i \tilde{x}) \cos(m_i \tilde{y}) \cos(l_j \tilde{x}) \cos(m_j \tilde{y})$$

and

$$\frac{\partial \mathcal{N}}{\partial a_i} = -\mathcal{N} \frac{\partial \mathcal{S}(s, v, r)}{\partial a_i} =: -\mathcal{N} C_i$$

where

$$C_i = \frac{1}{8} \sum_{j=0}^{I-1} a_j \sum_{\alpha \in Q} \sum_{\beta \in Q} \sum_{\gamma \in Q} A_{ij} + A_{ji} =: \sum_{j=0}^{I-1} a_j C_{ij}$$

with A_{ij} as defined in (9.10). Using this notation, we can express the first-order logarithmic derivative of the absolute probability $P(z | \mathcal{A})$ of a single measurement in a compact way:

$$\frac{\partial \log \{p(z | \mathcal{A})\}}{\partial a_i} = \begin{cases} \frac{\mathcal{N} C_i}{1-\mathcal{N}}, & \text{if } r = \text{sub} \\ \frac{B_i}{\lambda} - C_i, & \text{if } r \in [r_{\min}, r_{\max}] \\ -C_i, & \text{if } r = \text{super} \end{cases}$$

The derivative of the joint absolute measurement probability is then simply the sum of the derivatives of the individual measurement likelihoods

$$\frac{\partial \log \{p(Z | \mathcal{A})\}}{\partial a_i} = \sum_{k=0}^{K-1} \frac{\partial \log \{p(z | \mathcal{A})\}}{\partial a_i}.$$

The second-order derivatives of the measurement likelihood with respect to the map parameters are given by

$$\frac{\partial^2 \log \{p(z | \mathcal{A})\}}{\partial a_j \partial a_i} = \begin{cases} \frac{N(C_{ij} - C_i C_j)}{1-N} + \frac{N^2 C_i C_j}{(1-N)^2}, & \text{if } r = \text{sub} \\ \frac{B_{ij}}{\lambda} - \frac{B_i B_j}{\lambda^2} - C_{ij}, & \text{if } r \in [r_{\min}, r_{\max}] \\ -C_{ij}, & \text{if } r = \text{super} \end{cases}$$

As before, the second-order derivative of the joint measurement log-likelihood is the sum of the second-order derivatives of all measurements.

Acapulco Convention Center
University of Texas, ACES3
Belgioioso Castle
Massachusetts Institute of Technology, CSAIL
Edmonton Convention Centre
FHW museum
University of Washington, Seattle
University of Freiburg, 079
University of Freiburg, 101
Infinite corridors
Intel Research Lab
Örebro University

Table 9.1: The 12 datasets taken from the Robotics Data Set Repository [54] and used in all three experiment series.

9.5 Experiments

To assess how well DCT maps represent lidar data in comparison to existing mapping approaches, we conduct three series of experiments. In the first series, we compare the spatial map values of DCT maps and grid maps with identical memory requirements to a ground truth map and use the resulting error as a measure of map accuracy. In the second series, we evaluate the likelihoods that DCT maps, grid maps, Gaussian process occupancy maps, and Hilbert maps assign to measurements that were used to build them. The higher this likelihood, the better the respective map explains the underlying data. We conclude this section with a comparison of the empirical execution times of the different approaches.

The data at the basis of our experiments stems from rich planar lidar datasets recorded in spacious indoor environments. Each set contains the corresponding robot poses computed by SLAM, which we use as ground truth poses to build all maps. The data is publicly available from the Robotics Data Set Repository [54]. Table 9.1 shows which datasets were used in our experiments.

9.5.1 Map Value Comparison

In this experiment series, we compare the map values of DCT maps and decay-rate grid maps of different resolutions to the values of a decay-rate ground truth grid map. All grid maps are computed according to the algorithm described in [29]. At the beginning, for each dataset, we create a fine-grained ground truth grid map that covers a $10\text{ m} \times 10\text{ m}$ patch densely filled with 10^4 lidar measurements. It consists of 200×200 pixels with an edge length of 0.05 m . Then, we use the same sets of measurements to build pairs of one DCT map and one grid map, respectively, for each dataset and each resolution. The maps in these pairs possess the same number of parameters and require the identical amount of memory. We use five different map resolutions: 10×10 , 13×13 , 20×20 , 29×29 , and 40×40 . For grid maps, they correspond to pixel edge lengths of 1.00 m , 0.75 m , 0.50 m , 0.35 m , and 0.25 m . To give an intuition of what these maps look like, fig. 9.1 exemplarily shows the 40×40 DCT map of the Intel Research Lab dataset, the corresponding grid map, and the ground truth grid map.

Having created the maps, we sample the ground truth map at the mid-points of all cells that were observed at least once and look up the corresponding values in the DCT map and in the grid map. The resulting map values are hard to compare: As the decay rate λ is defined over the half-open interval $[0, \infty)$, the map values might be infinite. In order to make them comparable, we employ the strictly increasing monotonic transformation function $p_{\text{ref}} = 1 - \exp(-\lambda)$, which maps every decay value to a finite value $p_{\text{ref}} \in [0, 1]$. The value p_{ref} can be interpreted as the absolute probability that a ray is reflected before having traveled a distance of 1 m in a hypothetical homogeneous medium of decay rate λ . Please note that the distance 1 m is an arbitrarily chosen parameter. However, while surveying different distance values, we found out that varying this parameter has little effect on the quality of the results. We compute the root mean squared error (RMSE) in p_{ref} between the DCT maps and the ground truth map and between the grid maps and the ground truth map. Table 9.2 condenses the corresponding results by determining the mean and the standard deviation of the RMSE values. Additionally, it indicates the p -values of the one-sided paired-sample t -test. Small p -values indicate that the null hypothesis is unlikely and that the alternative hypothesis – the mean RMSE of DCT maps is smaller than the one of grid maps – becomes likely.

While at a resolution of 10×10 , both map modalities hardly differ in terms of accuracy, for all finer resolutions, DCT maps outperform grid maps at a confidence level of at least 99%. Note that the maximum gain in accuracy is located at a resolution of 29×29 ; at 40×40 , DCT maps are still significantly

l [m]	DCT		GM		p [%]	Δ_μ [%]
	μ	σ	μ	σ		
1.00	0.3314	0.0679	0.3330	0.0708	39.36	0.48
0.75	0.3146	0.0675	0.3319	0.0752	1.01	0.52
0.50	0.2932	0.0645	0.3093	0.0690	0.03	5.21
0.35	0.2571	0.0611	0.2822	0.0672	0.05	8.89
0.25	0.2370	0.0563	0.2543	0.0583	0.07	6.80

Table 9.2: Mean and standard deviation of the absolute RMSE values of DCT maps and grid maps with respect to the ground truth map, computed over all datasets. GM denotes grid maps, l is the pixel edge length, μ and σ denote the mean and the standard deviation of the RMSE, respectively, and p is the p -value of the one-sided paired-sample t -test. The variable $\Delta_\mu := 1 - \frac{\mu_{\text{DCT}}}{\mu_{\text{GM}}}$ indicates the improvement in RMSE of DCT maps relative to grid maps.

more accurate than grid maps, but the gain is not as large as for 29×29 . We attribute this to the fact that with increasing resolution, grid maps converge to the ground truth map, which itself is a grid map.

9.5.2 Measurement Probability Comparison

The maps computed in the first experiment series are maximum-likelihood maps. Maximum-likelihood maps shall maximize the measurement probability of the data that was used to create them. The higher this likelihood, the better the map represents the underlying data. Consequently, in the second experiment series, we interpret the likelihood a map assigns to its underlying data as a measure of its quality. We compare four different approaches: DCT maps, decay-rate grid maps, Gaussian process occupancy maps (GPOM), and Hilbert maps, which also model the spatial occupancy probability as a continuous scalar field. More specifically, we use GPOMs with Matérn kernel functions as described in [1] and Hilbert maps with Nyström features, which, according to [82], give the most accurate map results. All hyperparameters are set as described in [1] and [82], respectively. The data at the basis of the experiments is the same as in the previous experiment series, but the number of lidar measurements is reduced to 500.

The comparison is designed to guarantee that all maps have the same memory requirements in terms of numbers of real-valued parameters. For grid maps and DCT maps, we can ensure that by comparing maps with the same number of pixels and spectral parameters, respectively. For GPOM, we

l [m]	$lp_{\text{DCT}} - lp_{\text{GM}}$		$lp_{\text{DCT}} - lp_{\text{GPOM}}$		$lp_{\text{DCT}} - lp_{\text{HM}}$	
	μ	σ	μ [10^4]	σ [10^4]	μ [10^4]	σ [10^4]
1.00	88.5	74.1	4.21	3.00	4.30	3.42
0.75	150.6	146.5	3.89	2.81	4.19	3.46
0.50	135.7	63.9	3.53	2.60	4.06	3.42
0.35	196.1	101.5	2.97	2.15	4.12	3.32
0.25	96.8	132.0	2.68	1.90	4.12	3.31

Table 9.3: Mean and standard deviation of the log-likelihood differences between DCT maps and the other mapping approaches, computed over all datasets. The variable lp denotes the cumulated log-likelihood of all measurements in one dataset, GM denotes grid maps, HM means Hilbert maps, l is the pixel edge length, and μ and σ are the mean and the standard deviation of the log-likelihood differences, respectively.

randomly downsample the design matrix and the target vector so that the length of the Gaussian process parameter vector matches the number of grid pixels and spectral parameters, respectively. For Hilbert maps, we set the number of components of the Nyström features to the number of parameters corresponding to the specific resolution.

Now, we compute the joint measurement likelihood of all lidar measurements for each map modality. For grid maps, we calculate the measurement likelihood according to [29]. For DCT maps, we follow the equations given in section 9.4.3. For GPOMs and Hilbert maps, we rasterize their continuous occupancy fields with a pixel edge length of 0.05 m, perform ray tracing, and cumulate the occupancy probabilities along the rays.

Table 9.3 displays the resulting findings: the mean and standard deviation of the log-likelihood differences between DCT maps and the other approaches over all datasets. After having performed Anderson-Darling tests to ensure that the measurement probability quotients are indeed log-normally distributed, we perform one-sided paired-sample t -tests. For all resolutions, they indicate that DCT maps yield significantly higher measurement log-likelihoods at a confidence level of at least 98.56 %.

The results show that the log-differences between DCT maps and grid maps are two magnitudes smaller than those between DCT maps and GPOM or Hilbert maps, respectively. The level of the difference is influenced by the raster size chosen when computing the measurement likelihood for GPOMs and Hilbert maps. But the main reason for these large differences is the fact that both GPOMs and Hilbert maps have comparatively high memory

l [m]	t_{DCT} [s]	t_{GM} [s]	t_{GPOM} [s]	t_{HM} [s]
1.00	3.52	0.0917	1.12	22.8
0.75	4.69	0.0915	1.98	40.6
0.50	3.70	0.0923	3.25	108.4
0.35	18.25	0.0926	6.45	328.3
0.25	39.84	0.0942	14.04	949.3

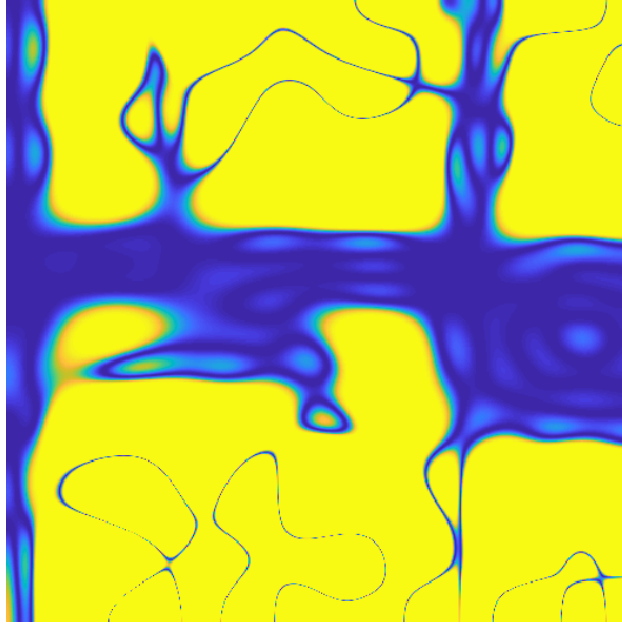
Table 9.4: Empirical execution time measurements collected during map creation. The variable t denotes the median of the mapping times over all datasets.

requirements. GPOMs store the map information in the parameter vector. The number of training points processed is proportional to the length of the parameter vector. As we restricted this length, only a limited number of training points could be processed; as a consequence, the classification results of GPOMs remain rather vague. As far as Hilbert maps are concerned, in [82], the authors recommend to use 1000 components for mapping with Nyström features. In our experiments, we use numbers as small as 100 parameters. Additionally, both GPOMs and Hilbert maps suffer from the fact that they need to sample a limited number of free and occupied training points along the laser rays, whereas DCT maps and decay-rate grid maps incorporate the full path information of an arbitrary number of rays. Fig. 9.2 illustrates the resulting differences in accuracy between the maps produced by the four approaches for 13×13 parameters.

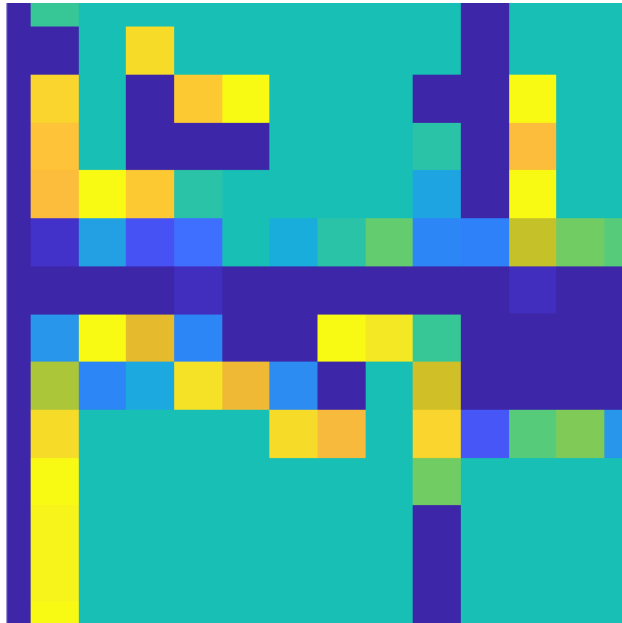
9.5.3 Execution Times

To give an intuition of the empirical runtime requirements of each of the methods used in the previous section, we average over ten mapping runs performed per method, dataset, and resolution. Table 9.4 lists the medians of these averages over all datasets. The measurements are collected on an Intel i7-2600K CPU running at 3.40 GHz. Grid maps, DCT maps, and GPOM are implemented in MATLAB R2017b. The GPOM implementation is based on the publicly available GPML toolbox [83]. To time Hilbert maps, we customized the Python implementation provided by [82]. The DCT optimization process is stopped once the relative change in the measurement log-likelihood is smaller than $1 \cdot 10^{-3}$.

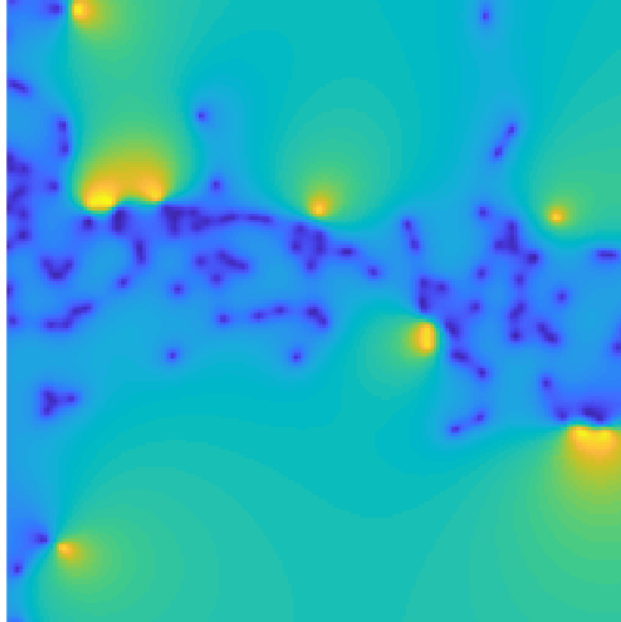
Table 9.4 indicates that grid maps are by far the fastest mapping technique. DCT maps and GPOMs are approximately two orders of magnitude



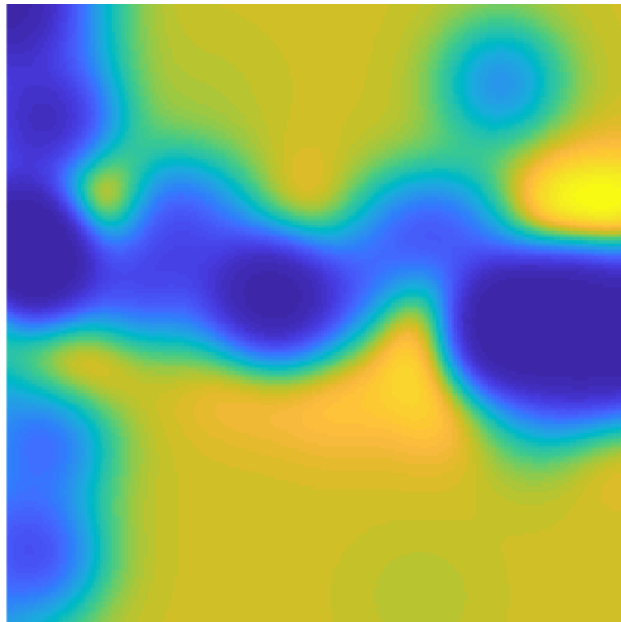
(a) DCT map.



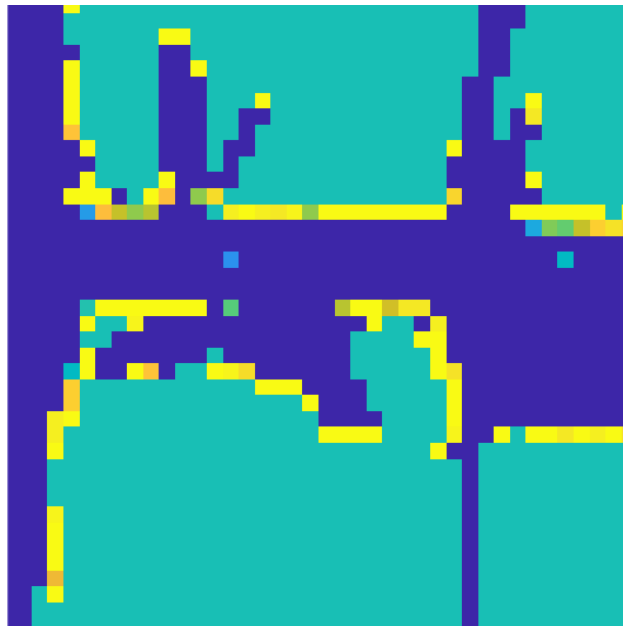
(b) Grid map.



(c) Gaussian process occupancy map.



(d) Hilbert map.



(e) Ground truth grid map.

Figure 9.2: Maps of different modalities created in the experiment series described in section 9.5.2 for the University of Washington dataset. The four maps to the left all have the same memory requirements of a mere 169 real-valued parameters. The grid map (e) is given as ground truth with a resolution of 40×40 . The decay-rate maps (a), (b), (e) show p_{ref} values as described in section 9.5.1, the other maps show occupancy probabilities. Blue means 0, green means 0.5, yellow means 1.

slower. This is due to the fact that during the optimization phase, DCT maps and GPOMs need to consider all parameters, which leads to quadratic computational complexity in the number of parameters. The most expensive operation in grid mapping, however, is ray casting, resulting in approximately linear computational complexity in the map resolution. Hilbert maps are three to four magnitudes slower than grid maps, the reason for this probably being the non-differentiable nature of the objective function, which needs to be approximated using finite differences.

9.6 Conclusion and Future Work

We presented a novel map representation based on the recently introduced decay-rate model for lidar sensors [29]. In contrast to most conventional maps, our so-called DCT maps store the map parameters in the discrete frequency domain. We applied the continuous extension of the inverse discrete cosine transform to the spectral parameters to obtain a continuously differentiable scalar field in the position domain.

Compared to other mapping approaches like decay-rate grid maps, Gaussian process occupancy maps (GPOM), and Hilbert maps, the proposed approach results in significantly improved map accuracy, as demonstrated in extensive real-world experiments. Moreover, DCT maps provide a ray tracing-based forward sensor model that allows to infer measurement probabilities directly from the spectral map representation in closed form, whereas the computation of ray tracing-based measurement probabilities based on continuous occupancy maps like GPOM and Hilbert maps necessitates the rasterization of the map and hence the introduction of a rasterization parameter. As opposed to GPOM and Hilbert maps, DCT maps use the full ray path information when building the map instead of sampling points along the ray.

Due to the promising experimental results, we plan improvements and extensions of DCT maps. First, we will address the issue that the computational complexity of DCT maps is higher than the one of grid maps, and that incremental updates require the repeated optimization of the map parameters. More specifically, we will develop a hybrid approach that locally optimizes the map and that makes use of massive parallelization. Second, we will extend the method by explicitly representing unexplored areas in the map. Currently, DCT maps are not able to distinguish between observed and unobserved map regions. Third, we will investigate how well DCT maps are suited for lossy compression. Finally, we plan to present a complete SLAM system based on DCT maps in the near future.

Acknowledgments

We thank Maani Ghaffari Jadidi for kindly supporting us with his GPOM implementation, Lionel Ott for releasing his Python implementation of Hilbert maps, and Patrick Beeson, Mike Bosse et al., Dieter Fox, Dirk Hähnel, Nick Roy, and Cyrill Stachniss for providing the datasets.

Chapter 10

A Maximum-Likelihood Method to Extract Polylines from 2-D Laser Range Scans

This scientific publication [4], written by Alexander Schaefer, Daniel Büscher, Lukas Luft, and Wolfram Burgard, was accepted for and presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems 2018, held in Madrid, Spain, from October 1 to 5, 2018. The IEEE holds the copyright on the article: © 2018 IEEE. Reprinted, with permission, from “A Maximum-Likelihood Method to Extract Polylines from 2-D Laser Range Scans”.

10.1 Abstract

Man-made environments such as households, offices, or factory floors are typically composed of linear structures. Accordingly, polylines are a natural way to accurately represent their geometry. In this paper, we propose a novel probabilistic method to extract polylines from raw 2-D laser range scans. The key idea of our approach is to determine a set of polylines that maximizes the likelihood of a given scan. In extensive experiments carried out on publicly available real-world datasets and on simulated laser scans, we demonstrate that our method substantially outperforms existing state-of-the-art approaches in terms of accuracy, while showing comparable computational requirements. Our implementation is available under <https://github.com/acschaefer/ple>.

10.2 Introduction

In order to navigate planar, structured environments like offices, households, or factory work floors, mobile robots often rely on horizontally mounted 2-D laser scanners. These sensors allow them to create floor plan-like maps, which they in turn use to localize themselves. Mapping and localization based on raw laser data, however, demand large amounts of computation power and memory, both of which tend to be restricted on a mobile platform.

A popular solution to this problem is feature extraction. Encoding all the data of a scan in a few polyline features, for example, can drastically reduce computation time and memory footprint. This is due to the ability of polylines to exploit the high redundancy of scans recorded in approximately line-shaped environments. Consider figure 10.1, which depicts a typical laser scan captured in an office. The scan spends hundreds of rays to describe straight walls, while a set of polylines with a total of ten vertices is sufficient to accurately represent these linear structures.

Polyline features like the ones in figure 10.1 have been shown to be useful for different tasks in mapping and localization, for example for feature-based SLAM [50, 84, 85, 86, 87] or for tracking line segments in consecutive 2-D scans recorded by a moving sensor to estimate 3-D planes in the environment [88].

In this paper, we present a novel method to extract polylines from 2-D laser scans. What sets our approach apart from most others is its probabilistic motivation. The derived algorithm does not rely on a geometric heuristic, but strives to find the set of polylines that maximizes the measurement likelihood of the scan. Furthermore, while most other approaches operate on the scan endpoints only and thereby discard valuable information encoded in the ray paths, our algorithm leverages this data to yield as accurate polyline estimates as possible. As demonstrated in our experimental evaluation, this results in superior accuracy both on real-world and on simulated data.

10.3 Related Work

In this section, we provide an overview over existing work on feature extraction techniques for 2-D lidar scans and related sensor modalities.

Early approaches to extract line features from 2-D laser scans produce so-called line maps [89]. As opposed to the chains of line segments that are polylines, line maps model the environment by infinite lines, and thus suffer from two major drawbacks: They can only represent convex maps, and they do not store any topology information, i.e. the connections between

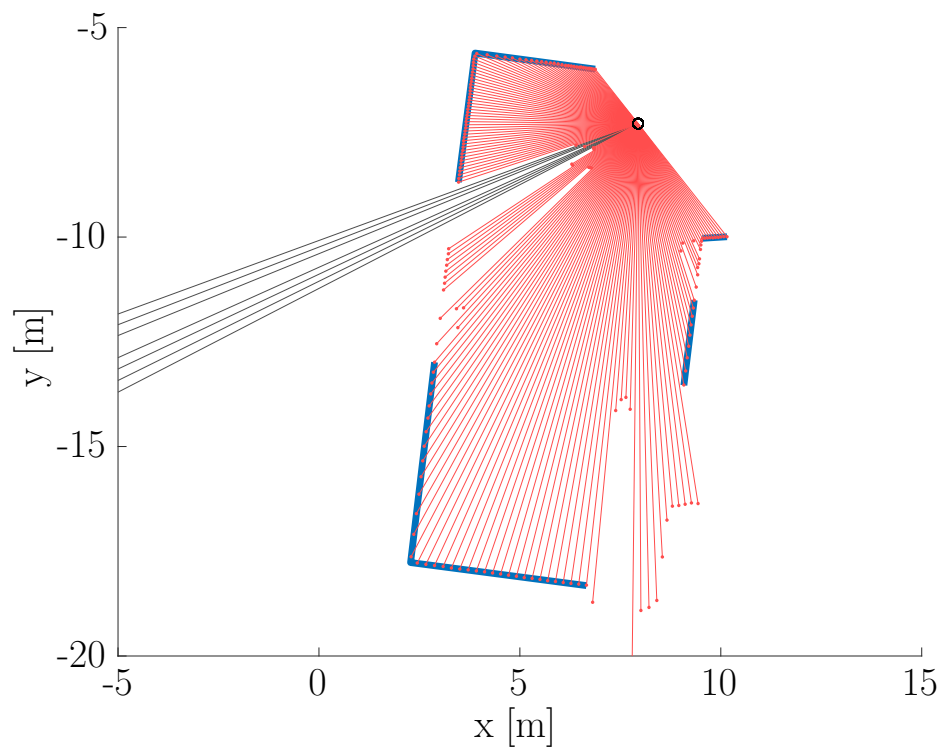


Figure 10.1: Exemplary result of our polyline extraction method applied to a scan captured in an office. The original scan consists of 361 rays, of which every second is displayed as a red line. Gray lines indicate maximum-range readings. The extracted polyline map, drawn as blue lines, consists of only ten vertices, reducing memory requirements to less than 3%. On average, the distance between the measured endpoints of the rays and their hypothetical intersections with the map is as low as 5.4 mm, with a maximum absolute value of 23.0 mm.

the lines. This is why infinite line representations have largely been replaced by polylines. For a survey on different methods to build infinite line maps, see Sack and Burgard [89].

Nguyen et al. [19] present an overview of various techniques to extract line segments from 2-D lidar data and evaluate the performance of six popular algorithms experimentally. They conclude that split-and-merge [22] and iterative endpoint fit [21] perform most favorably in terms of accuracy and speed.

Each of the following approaches tackles the line extraction problem from planar scans from a different perspective. Borges and Aldon [90] use a fuzzy clustering algorithm, which does not require prior knowledge of the number of lines. Latecki and Lakaemper [91] combine perceptual grouping techniques with the expectation maximization algorithm to determine polylines. Pfister and Burdick [92] extract both line and point features from scan endpoints using a multi-scale Hough transform. Similarly, Berrio et al. [93] determine line segments via the Hough transform in combination with a so-called successive edge following algorithm. Harati and Siegwart [94] build a wavelet framework to extract initial estimates of line segments. They do not, however, provide the corresponding line fitting algorithm.

In cartography, the Visvalingam line simplification algorithm [20] is a popular method to reduce the numbers of vertices of a polyline by iteratively removing the vertex that incurs the least perceptible change. Although to our knowledge, the algorithm has not reportedly put to use for line extraction from laser scans, it is well suited for this task.

All methods discussed thus far have in common that they are built upon some kind of heuristic and lack a probabilistic foundation. In contrast to those, Pfister et al. [95] present a take on line extraction from multiple scans that follows an elaborate maximum-likelihood formalism. First, they generate a set of infinite line estimates using the Hough transform. Then, taking into account the pose uncertainty of the robot and the measurement uncertainty of the sensor, they numerically maximize the measurement likelihood over the line parameters. In order to obtain line segments, they finally project the scan points onto the infinite lines and crop them accordingly. As opposed to this method, our method leverages probabilistics not only to optimize a given initial line estimate, but also to generate the estimate itself.

In another probabilistic approach, Veeck and Burgard [3] formulate an algorithm to extract polylines from multiple 2-D laser scans captured at known poses. In the first step, they create an occupancy grid map, which they then use to estimate the line contours of the environment. Second, they repeatedly apply a set of eight operations to these initial lines, including merging, splitting, adding vertices, removing vertices, moving vertex locations on a

raster grid, and removing the resulting zig-zag patterns. In this way, they strive to optimize the Euclidean distances between the laser scan endpoints and the nearest polyline. Our approach is different from Veeck and Burgard's in various aspects. For example, it is less complicated both conceptually and in terms of implementation. Moreover, their approach does not incorporate any ray path information in the result.

Polylines are useful features not only in the context of lidar. For an approach to extract line segments from sonar data using the Hough transform, see Tardós et al. [96]. Navarro et al. present methods to localize a robot using lines extracted from a rotating ultrasound sensor [97] and from infrared distance sensors [98].

For 2-D laser scans, there is only little research investigating features other than lines. Tipaldi and Arras' multi-scale FLIRT descriptors [99] are among these few. Bosse and Zlot convert a whole laser scan into a single feature [100] and extract features from quadratic areas formed by a set of scans [101].

10.4 Approach

In this work, we present a method to extract a set of polyline features from a 2-D lidar scan. In contrast to prevalent line extraction techniques like split-and-merge or iterative endpoint fit, our approach does not rely on a geometric heuristic, but maximizes the measurement probability of the scan to accurately determine polylines.

The method consists of two steps: polyline extraction and polyline optimization. Polyline extraction starts by connecting all neighboring scan endpoints to form a set of initial polylines. It then iteratively removes the vertex that incurs the least error in terms of measurement probability until it reaches a given threshold. The result is a set of polylines whose vertex locations coincide with the locations of a subset of the scan endpoints. To do away with this limitation, we formulate an optimization problem that moves the vertices to the positions that maximize the measurement probability of the scan. We call this latter process polyline optimization.

In the following, we first define the probabilistic sensor model. Then, we explain the polyline extraction step, before going into the details of polyline optimization.

10.4.1 Probabilistic Sensor Model

Both polyline extraction and polyline optimization strive to maximize the measurement probability. By measurement probability, we refer to the probability of a laser scan conditioned on a specific set of polylines. In order to describe this quantity mathematically, we need to define all necessary variables. We denote the scan by $Z := \{z_k\}$, where $k \in \{1, 2, \dots, K\}$ represents the index of the laser ray. A single laser measurement $z := \{a, b\}$ is composed of two two-element column vectors: the starting point of the ray a and the endpoint b . The set of polylines L consists of a total of I individual polylines. These polylines are ordered sets $l := \{v_j\}$, composed of at least two pairwise distinct vertices v . The vertices v , just like the ray starting points and endpoints a and b , are specified with respect to the coordinate system of the polyline map L . Note that no vertex can be part of multiple polylines:

$$\bigcap_{i=1}^I l_i = \emptyset.$$

Most lidar sensors exhibit approximately normally distributed noise in radial direction and relatively small angular noise. Consequently, we neglect angular noise and model the distribution of the measured ray radius given the polyline map as a Gaussian probability density function centered at the true radius. With the above definitions, we formulate the sensor model for a single ray as

$$p(z | L) = \mathcal{N}(r(z); \hat{r}(z, L), \Sigma), \quad (10.1)$$

with the measured ray radius $r(z) := \|b - a\|$. The function $\hat{r}(z, L) \in \mathbb{R}^+$ computes the distance between the starting point of the ray and the first intersection between its axis and the polyline map. The variance of the radial noise Σ is usually a function of multiple parameters such as the sensor device, the ray radius, the optical properties of the reflecting surface, and temperature. It can either be read off the datasheet of the sensor or determined experimentally.

By assuming independence between the individual laser rays of a scan, we extend equation (10.1) to compute the measurement probability of the whole scan as

$$p(Z | L) = \prod_{k=1}^K p(z_k | L).$$

This formula represents the measurement probability that both steps of our algorithm strive to maximize.

10.4.2 Polyline Extraction

Line extraction is always a compromise between memory requirements and accuracy of the produced lines. This compromise needs to be quantified. Embedded applications, for example, might focus on minimal memory footprint, while offline mapping systems might favor high accuracy at the expense of polylines that consist of hundreds or thousands of vertices. For this reason, every line extraction algorithm requires some kind of parameter. In the following, we choose this parameter to be the maximum number of vertices J_{\max} of the polyline set, because it allows direct control over the memory footprint of the result. Note, however, that our approach makes it easy to use arbitrary parameters, for example the maximum root mean squared error of the ray radii, the Akaike Information Criterion, the maximum difference in area between the extracted polylines and the polygon of the original scan endpoints, etc., as described further below.

Given a specific maximum number of vertices J_{\max} , the goal of the polyline extraction step is to find the set of polylines L^* with at most J_{\max} vertices that, among all other polyline maps with at most J_{\max} vertices, yields the highest measurement probability. Formally, we are confronted with the optimization problem

$$L^* = \operatorname{argmax}_L p(Z | L) \mid J(L) = J_{\max}, \quad (10.2)$$

where $J(L)$ denotes the number of vertices in L .

Solving (10.2) is primarily a combinatorial problem. Even if we knew the locations of the J_{\max} vertices, we still do not know the data associations, i.e. which vertices make up which polyline. Exhaustively searching the space of all data associations for the combination that maximizes the measurement probability quickly leads to combinatorial explosion even for small J_{\max} . For that reason, we use a greedy algorithm to solve the combinatorial part of (10.2).

In a nutshell, the algorithm first creates a polygon by connecting all neighboring scan endpoints. Starting from this initial map, it iteratively removes the vertex that reduces the measurement probability of the scan given the map by the least amount, until it reaches the desired number of vertices, or until another stopping criterion like one of those mentioned above is fulfilled.

Given the initial or any intermediate polyline map L , the problem of finding the vertex v_{j^*} that reduces the measurement probability the least

can be formulated as

$$\begin{aligned}
j^* &= \operatorname{argmax}_j p(Z \mid L \setminus v_j) \\
&= \operatorname{argmax}_j \log\{p(Z \mid L \setminus v_j)\} \\
&= \operatorname{argmin}_j \sum_{k=1}^K \frac{d^2(z_k, L \setminus v_j)}{\Sigma_k} \\
&= \operatorname{argmin}_j \left\{ \sum_{k=1}^K \frac{d^2(z_k, L \setminus v_j)}{\Sigma_k} - \sum_{k=1}^K \frac{d^2(z_k, L)}{\Sigma_k} \right\} \\
&= \operatorname{argmin}_j \sum_{k=1}^K \frac{d^2(z_k, L \setminus v_j) - d^2(z_k, L)}{\Sigma_k} \\
&= \operatorname{argmin}_j \underbrace{\sum_{k \in X(L, v_j)} \frac{d^2(z_k, L \setminus v_j) - d^2(z_k, L)}{\Sigma_k}}_{=: e_j} \\
&= \operatorname{argmin}_j \{e_j\}, \tag{10.3}
\end{aligned}$$

where we define $L \setminus v_j := \{l_i \setminus v_j \mid i = 1, 2, \dots, I\}$. Accordingly, $p(Z \mid L \setminus v_j)$ represents the probability density function of the measurements conditioned on the set of polylines with the vertex v_j removed. The function $d(z, L) \in \mathbb{R}$ determines the distance between the endpoint of a ray and its intersection with the map

$$d(z, L) := r(z) - \hat{r}(z, L).$$

Please note that the transition from the third to the fourth line of equation (10.3) is valid because the second sum $\sum_{k=1}^K d^2(z_k, L) \Sigma_k^{-1}$ is constant with respect to j . The function $X(L, v_j) \subseteq \{1, 2, \dots, K\}$ in the sixth line returns the indices of the rays that intersect any of the line segments that start or end at v_j . The variable e_j can be thought of as a measurement probability error term corresponding to the removal of the j -th vertex. Consequently, removing the vertex that decreases the measurement probability the least is equivalent to removing the vertex whose removal incurs the smallest error. For an illustration of the quantities that need to be determined in order to compute the errors, see figure 10.2.

Computing the error terms according to equation (10.3) is valid if the vertex in question has two neighbors. If the vertex marks the start or the end of the polyline, however, removing it means removing the corresponding line segment. In this case, it is not possible to compute the term $d(z_k, L \setminus v_j)$. Consequently, we resort to a heuristic: We introduce a constant parameter d_{rm} , which acts as a placeholder for $d(z_k, L \setminus v_j)$ in case the latter term

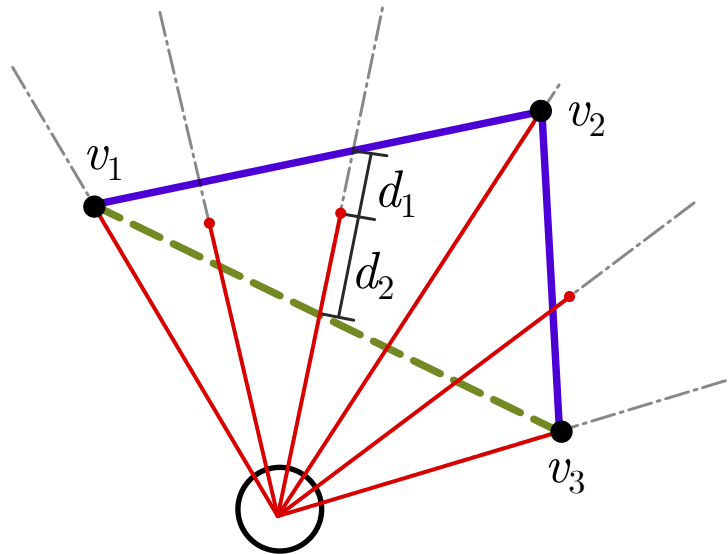


Figure 10.2: Illustration of the different terms in equation (10.3). The black circle represents the lidar sensor. It shoots laser rays in different directions, drawn as red lines with dots marking their measured endpoints. The blue polyline is the map L , its vertices are denoted v_1, v_2, v_3 . The green dashed line depicts $L \setminus v_2$, the polyline L with its middle vertex v_2 removed. When computing the error e_2 corresponding to the removal of v_2 , we need to determine two quantities for each ray z_k . The first one is the distance $d(z_k, L)$ between the measured endpoint and the intersection between the ray and the given polyline L , exemplarily shown for one ray and denoted d_1 in the graphic. The second one is the distance $d(z_k, L \setminus v_2)$ between the endpoint and the polyline with the vertex v_2 removed, denoted d_2 in the graphic.

is impossible to determine. The magnitude of d_{rm} controls how easily the algorithm crops lines. If chosen large, the algorithm rather keeps the line segments and prefers to remove the vertices in the middle of the polylines. If chosen small, it tends to crop lines and reluctantly removes two-neighbor vertices.

The pseudocode in listing 1 shows the workings of the algorithm in detail and delineates an efficient implementation. In lines 1 to 10, the algorithm forms the initial map. To that end, it connects all neighboring scan endpoints that satisfy two conditions. First, neither of the points in the pair that is to be connected is a maximum range reading (line 3). Maximum-range readings emerge when there is no object within the range of the lidar sensor, so removing the corresponding line segments is consequential. Second, the length of the connection between the points does not exceed a given maximum l_{max} (line 4). This step prevents the generation of long lines that are not sufficiently backed up by lidar data, for example connections between a short-range endpoint and a long-range endpoint, or connections between neighboring endpoints at large radii, far away from the sensor. After the resulting set of line segments have been merged (lines 9 to 10), L is either a polygon, if all endpoint pairs meet both conditions, or a set of polylines otherwise.

To reduce the number of vertices, the map L is then subjected to the greedy part of the algorithm, represented by lines 11 to 21. Lines 11 to 15 initially compute the error values corresponding to the removal of the individual vertices v_j in L . The loop spanning lines 16 to 21 then iteratively removes the vertex corresponding to the smallest change in measurement probability and updates the errors, until the desired number of vertices is reached. Note that in line 20, it is not necessary to recompute all errors in E . Only the errors corresponding to the immediate neighbors of v_{j^*} change. After the removal of vertex v_{j^*} , those are indexed by $j^* - 1$ and j^* .

If a stopping criterion other than the number of vertices is given, for example a maximum RMSE value, the condition in line 16 simply needs to be changed accordingly.

Algorithm 1 solves the combinatorial part of the optimization problem (10.2). It tells both which scan endpoints create which vertices, and which vertices form which polyline. It does not, however, alter the positions of the vertices in order to maximize the measurement probability. In polyline extraction, the vertex locations are limited to the locations of the scan endpoints. The next step, polyline optimization, relaxes this restriction.

Algorithm 1: Polyline Extraction

Data: $Z, r_{\max}, l_{\max}, J_{\max}$
Result: L

```

1  $L \leftarrow \{\}$ 
2 for all  $z_k$  in  $Z$  do
3   if  $(\max(r_k, r_{k+1}) \leq r_{\max})$ 
4      $\wedge (\|b_{k+1} - b_k\| \leq l_{\max})$  then
5        $l \leftarrow \{b_k, b_{k+1}\}$ 
6       add  $l$  to  $L$ 
7     end
8 end
9 merge all line segments  $l \in L$  to polylines so that
10    $\bigcap_{i=1}^I l_i = \emptyset$ 
11  $E \leftarrow \{\}$ 
12 for all  $v_j$  in  $L$  do
13   compute error  $e_j$  corresponding to removal of  $v_j$ 
14   add  $e_j$  to  $E$ 
15 end
16 while  $J(L) > J_{\max}$  do
17   find index  $j^*$  of smallest element in  $E$ 
18   remove  $v_{j^*}$  from  $L$ 
19   remove  $e_{j^*}$  from  $E$ 
20   recompute errors  $e_{j^*-1}$  and  $e_{j^*}$  in  $E$ 
21 end

```

10.4.3 Polyline Optimization

Having solved the combinatorial part of the optimization problem (10.2), we now turn to its numerical part: We take the vertex locations produced by the polyline extraction step and move them to the positions that maximize the measurement probability of the scan conditioned on the map $p(Z | L)$.

More formally, we want to solve

$$L^* = \operatorname{argmax}_L p(Z | L) = \operatorname{argmin}_L \sum_{k=1}^K \frac{d^2(z_k, L)}{\Sigma_k}, \quad (10.4)$$

which is a nonlinear, discontinuous, multivariate optimization problem in the coordinates of the polyline vertices. Its discontinuous nature, which results from the polylines' kinks in the vertices, requires appropriate direct search solvers, for example the Nelder-Mead Simplex Method [102].

Before starting to optimize, it is important to closely consider the search space of the problem formulated in equation (10.4). In the case of a closed polygon, this space simply becomes \mathbb{R}^{2J} , where J is the number of vertices in L . A corresponding candidate solution consists of the coordinates of all polygon vertices. In the case of a set of polylines, however, allowing all vertices to freely move around might lead to undesired effects: Vertices at the start or end of a polyline might drift off into unobserved regions. Consider vertices v_1 and v_3 in figure 10.2, for example. As long as they stay on the axis of the respective line segment, they can move indefinitely away from the observed region without affecting the measurement probability of the scan. To avoid this effect and to keep the search space as small as possible, we constrain the movement of vertices at the start or end of a polyline to the axis of the corresponding ray. Given a map consisting of I polylines, the dimensionality of the search space hence becomes $2(J - I)$.

10.5 Experiments

In order to assess the quality of the polyline maps produced by the presented method and to compare the results with those returned by existing approaches, we conduct two series of experiments. In the first, we evaluate the performance of every method on real-world 2-D lidar data. The data is composed of 13 datasets taken from Radish, the publicly available Robotics Data Set repository [54]. From each of those datasets, listed in table 10.1, we randomly choose 20 scans, leading to a total of 260 scans to evaluate. On average, each of the selected scans contains 264 rays. The second experiment series is based on the same number of simulated scans with 360 rays

Acapulco Convention Center
University of Texas, ACES3
Belgioioso Castle
Massachusetts Institute of Technology, CSAIL
Edmonton Convention Centre
FHW museum
University of Freiburg, 079
University of Freiburg, 101
University of Freiburg, campus
Infinite corridors
Intel Research Lab
Örebro University
University of Washington, Seattle

Table 10.1: Datasets taken from the Robotics Data Set Repository [54].

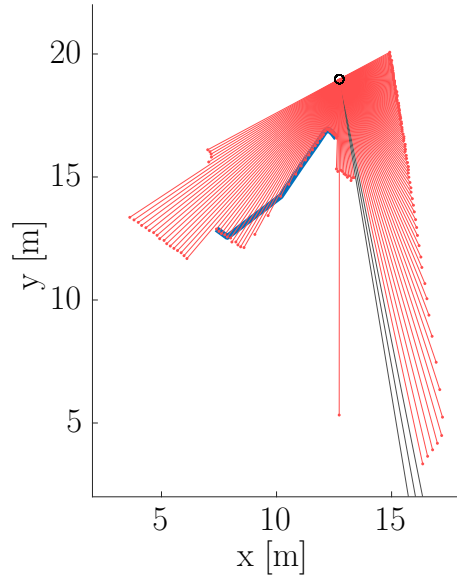
each. Simulation allows us to measure the accuracy of the obtained polyline maps not only with respect to the scan data, but also with respect to the underlying ground-truth map. To simulate a scan, we first create a random polygon with 3, 4, 5, 6, 12, 36, or 180 vertices. We then sample a noisy full-revolution, 360-ray scan from it by first applying normally distributed noise to the ray angles, then computing the true intersection points of scan and polygon, and by finally adding normally distributed noise to the corresponding ray radii. The standard deviations of angular and radial noise are 0.2° and 0.03 m, respectively.

In both experiment series, we compare six different takes on polyline extraction, starting with Visvalingam line simplification (VVL) [20]. The method requires initial polylines, so we first connect the endpoints of the scan using the exact same procedure as described in algorithm 1, line 1 to 10, with l_{\max} set to 1 m. Visvalingam’s algorithm then simplifies the resulting initial polygon or set of polylines by iteratively removing the vertex whose removal is linked to the least perceptible change in the polyline. The popular iterative endpoint fit algorithm (IEF) [21] comes second in our comparison. As opposed to the top-down approach of VVL, which starts with the most detailed line and iteratively simplifies it, IEF builds polylines from bottom up. In short, IEF takes a set of range measurements, connects the first and the last point by a straight line, and then inserts the scan endpoint with the largest distance from the line as a vertex into the line. It repeats this process

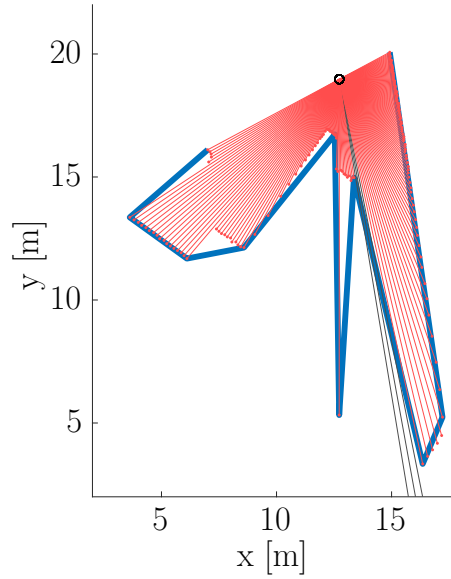
until it reaches the specified number of vertices. Split-and-merge (SAM) [22] is an extension of IEF. The only difference between the two algorithms is that in each iteration, SAM first fits the line estimate to the scan points by minimizing the squared distances between the points and the line. Both algorithms do not account for maximum-range readings, which is why we removed them from the laser scans before passing the scans to IEF or SAM. VB, the fourth method in our comparison, denotes the polyline learning algorithm proposed by Veeck and Burgard [3]. We call our approach probabilistic line extraction (PLE). If the vertices provided by PLE are optimized using the procedure described in section 10.4.3, we denote it by PLE+. Throughout all experimental runs, we set $l_{\max} = 1$ m and $d_{\text{rm}} = 0.5$ m. Furthermore, we assume that the radial variance Σ of all rays is the same. As a consequence, we do not have to specify any variance at all, because if constant, the term Σ_k vanishes from equations (10.3) and (10.4).

Figure 10.3 exemplarily illustrates the results obtained by applying the described methods to the same scan. Although the desired number of vertices was set to $J = 10$ for all methods, the returned polyline maps differ considerably.

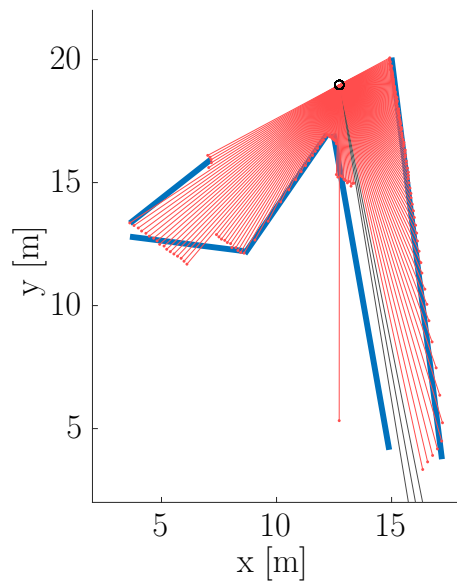
Figure 10.4 summarizes our findings pertaining to the accuracy of the investigated methods. It displays the evolution of the results over increasing memory footprint, encoded by the number of vertices J . For each method and each dataset, we evaluate the following values of J : 10, 20, 30, 40, 50. We employ three different metrics to look at the results from different perspectives. The root mean squared error (RMSE) of the ray radii assesses how closely the extracted polylines represent a scan. The E in RMSE is the distance between the measured endpoint of a lidar ray and its hypothetical intersection with the polyline, measured along the ray axis. For each scan, we determine one RMSE value by iterating over all rays. For every algorithm, we then average the RMSE over all scans to obtain the values shown in figure 10.4. The second metric, denoted by f , quantifies the match between the polyline map and the original scan in a different way. For each scan, f is the ratio of the number of rays hypothetically intersected by the polylines and the number of rays actually reflected in the measured scan. The f -values presented in figure 10.4 are again averaged over all scans. For the simulated experimental runs, we introduce the third metric a . It measures how well an extracted polyline map recovers the ground truth map. To determine the a -value, we transform the coordinates of the estimated polyline vertices to polar coordinates with respect to the sensor pose, order them counter-clockwise, and build a polygon out of them. We then compare how well the resulting polygon matches the one that represents the ground-truth. More



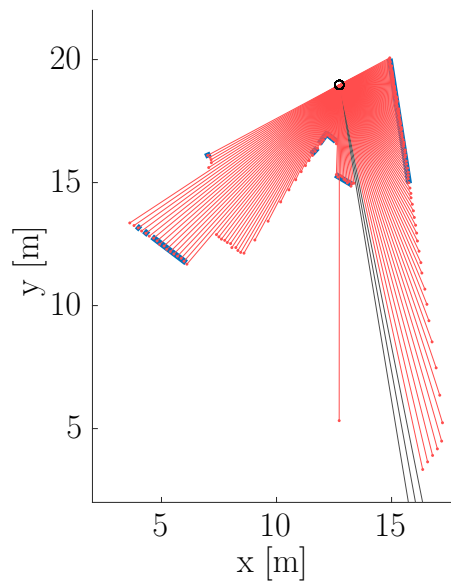
(a) Visvalingam's algorithm.



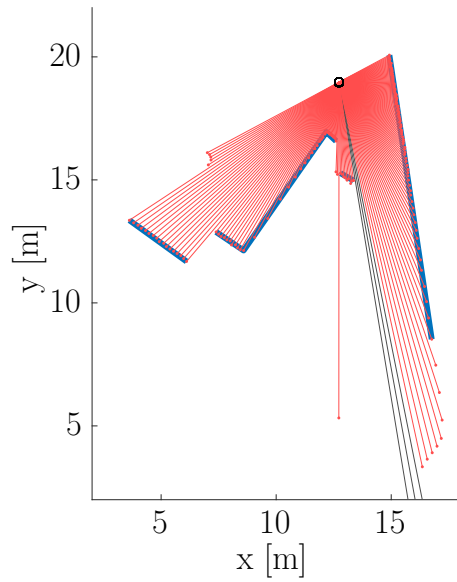
(b) Iterative endpoint fit.



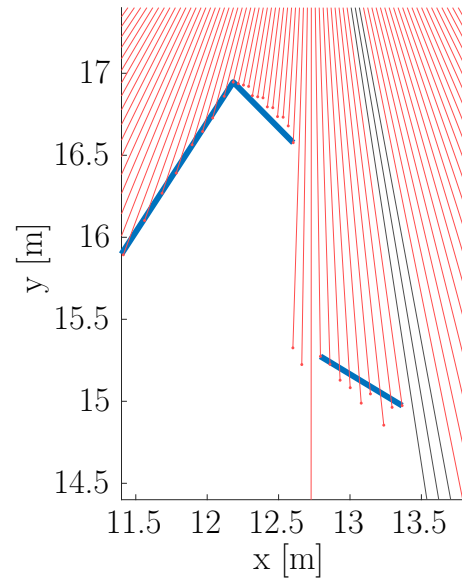
(c) Split-and-merge.



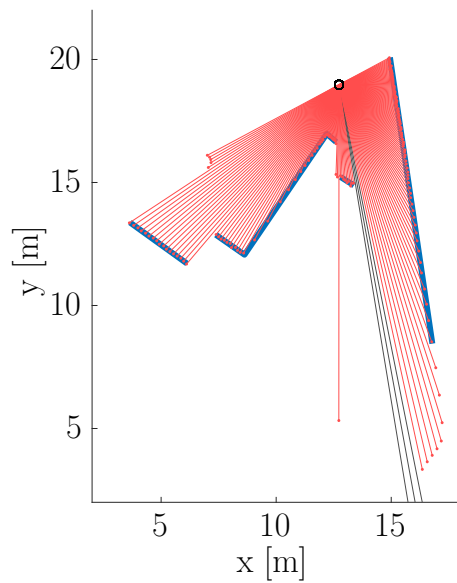
(d) Veeck and Burgard's method.



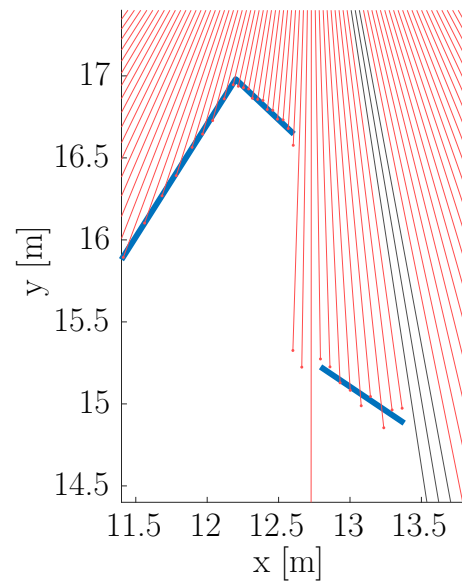
(e) Ours without optimization.



(f) Zoom into (e).



(g) Ours with optimization.



(h) Zoom into (g).

Figure 10.3: Exemplary results of the various polyline extraction methods applied to the same scan captured in an office. All methods respect the requirement of at most 10 vertices, except for VB, which produces 17 vertices.

specifically, we compute the ratio of areas

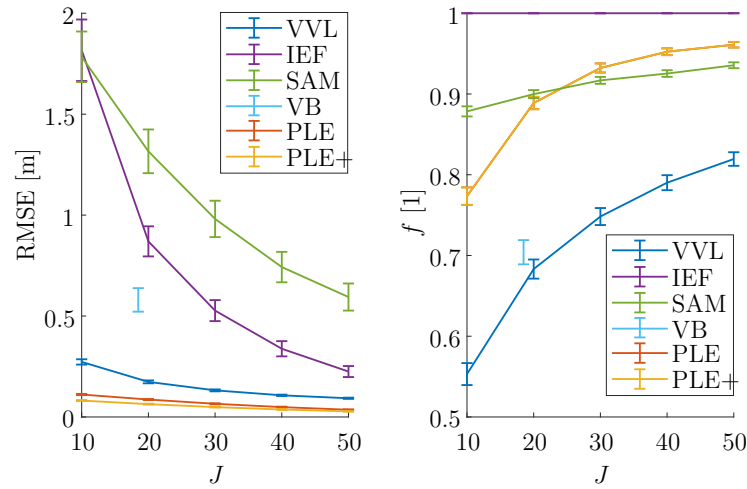
$$a := \frac{(a_{\text{GT}} \cup a_{\text{E}}) - (a_{\text{GT}} \cap a_{\text{E}})}{a_{\text{E}}},$$

where a_{GT} denotes the area of the ground-truth polygon, while a_{E} stands for the area of the estimated polygon.

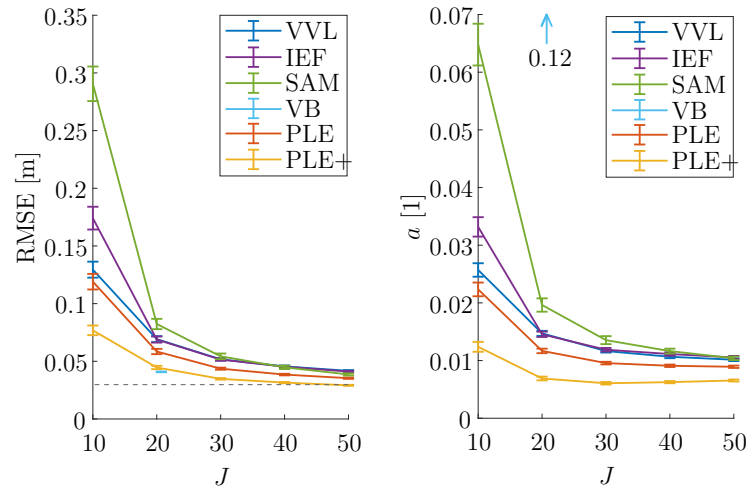
Figure 10.4 reveals that although very popular, the split-and-merge algorithm performs poorly relative to the other line extraction methods we selected. Even more surprisingly, its less elaborate predecessor, iterative endpoint fit, outperforms SAM with respect to every metric we evaluated. We tracked the reason for this behavior down to the line fitting process: Especially in the first iterations, when the number of polyline segments is by far too small to represent the structure of the environment, fitting often leads to a degradation of the line estimate. During the later iterations, when refining the polylines, the algorithm is not able to compensate this inaccurate prior. The described behavior is not only apparent in the RMSE, but also in the f -value. While IEF explains all measurements, SAM does not account for a significant amount of rays. So in contrast to the conclusion that Nguyen et al. [19] drew after experimentally comparing SAM and IEF on a dataset of 100 scans, we find that split-and-merge performs significantly poorer than iterative endpoint fit.

The results of the method developed by Veeck and Burgard are only given for $J \approx 20$. This is due to the fact that their approach does not allow to set the memory limits of the resulting polyline directly. Instead, one has to provide a target value for the Bayesian Information Criterion (BIC), which they use to balance the compromise between memory requirements and accuracy. Unfortunately, even large variations in the BIC value lead to similar vertex counts. For that reason, we are not able to evaluate VB over the whole range of J . The results demonstrate that VB achieves good accuracy, but only for the limited set of laser rays it explains. At 0.71, the f -value turns out comparatively low both on real data and in simulation. The reason is the grid map-based initialization of the polylines, which discards grid cells with low occupancy values. Figure 10.3d illustrates this behavior. In contrast to all other methods, VB is not able to extract the line that runs approximately through coordinate (10, 13). The occupancy probability along this line is simply too small to qualify as an initial polyline. As a consequence of this behavior, VB returns the most inaccurate polygons in simulation.

Although to our knowledge never evaluated in a robotics context, Visvalingam’s algorithm returns comparatively low RMSE values in both experimental settings. The characteristic it suffers from most is its small f -value – a consequence of the fact that the removal of straight lines comes at no



(a) Mean accuracy of extracted polylines on real-world lidar data.



(b) Mean accuracy of extracted polylines on simulated lidar data.

Figure 10.4: Experimentally determined accuracies of the investigated polyline extraction methods. RMSE denotes the root mean squared error between the measured laser ray endpoints and the hypothetical intersection with the extracted set of polylines, averaged over all scans. The variable f indicates the fraction of rays explained by the polyline map, whereas a denotes the relative area error between the polygon extracted from simulated data and the underlying ground-truth polygon. The error bars in the plots visualize the standard errors. The dashed line on the left side of plot (b) marks the standard deviation of the radial noise for the simulated laser scans.

cost. Hence, the algorithm discards any solitary line segment in order to decrease the vertex count. The exemplary output of the Visvalingam method in figure 10.3a shows exactly this behavior. The solitary line segments representing the long walls at the top and on the right side of the image had to make way for the nine vertices in the blue polyline. Some of them are hardly recognizable because the corresponding kinks in the line are so small. In simulation, where the scan represents a closed polygon, the described effect does not appear, resulting in an f -value of 1.

The line extractors proposed in this paper, PLE and PLE+, outperform all other methods on real data and in simulation. As shown in figure 10.4, both algorithms result in significantly smaller RMSE values than the other methods, except for VB, which exhibits a slightly lower RMSE on simulated data. However, VB is unable to accurately recover the simulated polygons, achieving an f -value of only 0.71, while PLE and PLE+ attain 1. PLE+ always exhibits smaller RMSE values than PLE, because the optimization minimizes exactly this metric. The superior a -values in figure 10.4b demonstrate that minimizing the RMSE also leads to an improved representation of the underlying ground-truth map. Note that in figure 10.4a, the f -values of PLE and PLE+ are exactly the same, because PLE+ does neither change the topology of the polylines extracted by PLE, nor does the optimization allow boundary rays to interfere with rays that account for measurements outside the polyline.

As expected, the RMSEs of all algorithms in figure 10.4b approach the standard deviation of the simulated radial sensor noise for large J . PLE+ even falls below this value, an effect that is due to the algorithm overfitting highly articulated polygons to the noise in the scans. Correspondingly, the area error increases slightly for large J .

Lastly, we report on the computational costs for all methods in table 10.2. Each algorithm ran in a single thread on an Intel Xeon CPU with 2.50 GHz. The bottom-up line simplification algorithms IEF and SAM exhibit slightly decreasing computation times for increasing J , because higher J -values mean less simplification steps. The repeated fitting steps in SAM turn out to be costly: In the worst case, SAM is 100 times slower than IEF. The reason for the constant timing of Visvalingam's algorithm lies in our implementation: At first, for every polyline in the map, we compute the incremental errors until the line has vanished. We then order the errors globally, i.e. over all polyline segments, and remove as many vertices as required to meet the specified vertex count. Despite this simplified implementation, VVL is at least ten times faster as the popular IEF. Our algorithms PLE and PLE+ are in the mid-range among the investigated methods. As a result of the optimization via direct search, we find that the complexity of PLE+ grows

	J	VVL	IEF	SAM	VB	PLE	PLE+
Real	20	0.056	0.27	28	0.28*	1.4	2.3
	50	0.050	1.12	27	–	1.3	6.9
Simulated	20	0.037	0.38	103	0.48**	2.3	11
	50	0.037	1.59	102	–	2.2	72

* $J = 18.5$ ** $J = 20.7$

Table 10.2: Mean computation times in seconds.

approximately quadratically in J . At the same time, the advantage gained by the optimization process decreases for high numbers of vertices, as can be read off figure 10.4. Therefore, we recommend to use PLE+ to extract only few, but highly accurate vertices. If memory requirements are less strict and timing becomes an issue, PLE is the right choice.

Both our MATLAB implementation of the presented line extraction approach and the scripts used to conduct and evaluate the experiments are publicly available under <https://github.com/acschaefer/ple>.

10.6 Conclusion and Future Work

In order to extract polylines from a 2-D laser scan, one has to answer two questions: Which polyline reflects which scan endpoints? And where are the optimal locations of the polyline vertices? In the present paper, we answer the first question using a greedy algorithm that minimizes the decrease in measurement probability caused by representing individual scan endpoints by line segments. The answer to the second question is given by a direct search optimizer that moves the vertices in order to maximize the measurement probability. Extensive experiments on publicly available datasets and simulated data demonstrate that our approach clearly outperforms all four reference approaches.

Due to the promising results, we will build upon the presented approach in the future and extend the line extractor to three dimensions, resulting in a maximum-likelihood approach to extract planes from 3-D laser range scans.

Acknowledgments

We thank Michael Veeck for kindly supporting us with the implementation of his line extraction method, and Patrick Beeson, Mike Bosse, Dieter Fox,

Giorgio Grisetti, Dirk Hähnel, Nick Roy, and Cyrill Stachniss for providing the datasets.

Chapter 11

A Maximum-Likelihood Approach to Extract Finite Planes from 3-D Laser Scans

This contribution [103], authored by Alexander Schaefer, Johan Vertens, Daniel Büscher, and Wolfram Burgard, was accepted for and presented at the IEEE/RSJ International Conference on Robotics and Automation 2019, which took place in Montreal, Quebec, Canada, from May 20 to 24, 2019. The IEEE holds the copyright on the article: © 2019 IEEE. Reprinted, with permission, from “A Maximum-Likelihood Approach to Extract Finite Planes from 3-D Laser Scans”.

11.1 Abstract

Whether it is object detection, model reconstruction, laser odometry, or point cloud registration: Plane extraction is a vital component of many robotic systems. In this paper, we propose a strictly probabilistic method to detect finite planes in organized 3-D laser range scans. An agglomerative hierarchical clustering technique, our algorithm builds planes from bottom up, always extending a plane by the point that decreases the measurement likelihood of the scan the least. In contrast to most related methods, which rely on heuristics like orthogonal point-to-plane distance, we leverage the ray path information to compute the measurement likelihood. We evaluate our approach not only on the popular SegComp benchmark, but also provide a challenging synthetic dataset that overcomes SegComp’s deficiencies. Both our implementation and the suggested dataset are available at [104].

11.2 Introduction

The geometry of many man-made environments like factory floors, offices, and households can be described by a set of finite planes. Robots navigating these types of environments often rely on 3-D laser range finders, which capture up to millions of reflections per second. Plane extraction methods take these highly redundant raw sensor measurements and reduce them to the parameters of the underlying planes, thus reducing the computational effort and the memory footprint required for processing the sensor data. Plane extraction may also increase accuracy in tasks like scan matching and sensor calibration, and it enables applications like model reconstruction and object detection in the first place.

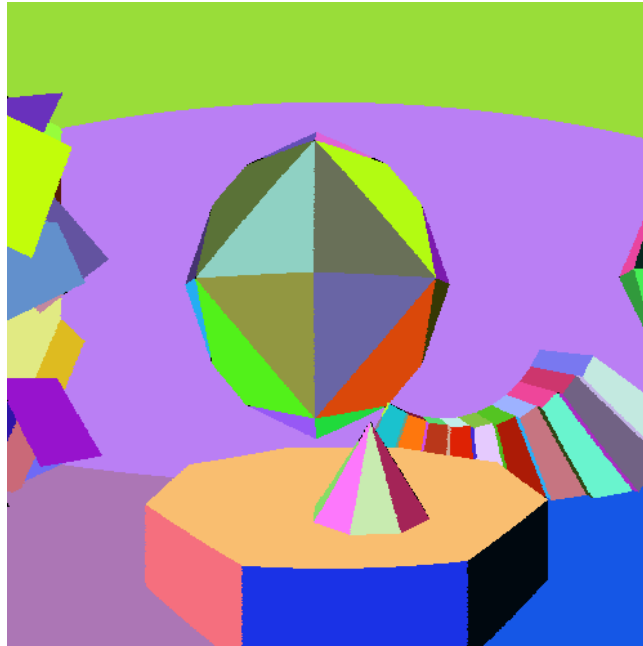
The presented method, dubbed probabilistic plane extraction (PPE), extends our recent work on polyline extraction from 2-D laser range scans [2] to three dimensions. Essentially, PPE is a maximum-likelihood approach based on agglomerative hierarchical clustering. In the beginning, PPE represents the scan by a large set of planes – one plane for every reflection – and then iteratively merges them, in each step choosing the subset whose merger maximizes the measurement likelihood of the whole scan, until a specified stopping criterion is met. Figure 11.1b shows an exemplary segmentation result.

Our approach distinguishes itself from the large body of related work in two respects. First, all methods surveyed in the following resort to heuristics like orthogonal distance between ray endpoint and plane when estimating the measurement likelihood of a scan conditioned on a set of planes. Instead, PPE accounts for the true ray path from start to end. This more accurate sensor model leads to more accurate results, as demonstrated by our experiments. Second, due to its probabilistic formulation, PPE requires only one robust parameter to control the granularity of the extracted planes. In contrast, some of the surveyed methods need up to a dozen carefully tuned parameters in order to obtain reasonable results.

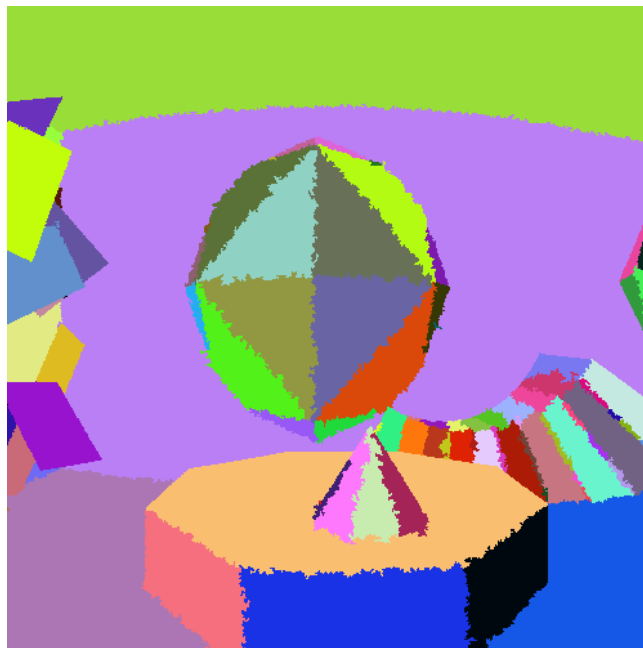
11.3 Related Work

This section provides an overview over the state of the art considering plane extraction from 3-D lidar scans. We distinguish four classes of approaches: region growing, clustering, random sample consensus (RANSAC), and the Hough transform.

In a nutshell, region growing first selects some seed points from the input point cloud, which are then grown into regions by iteratively adding all



(a) Ground-truth segmentation.



(b) PPE segmentation.

Figure 11.1: Ground-truth segmentation of an organized 500×500 point cloud taken from the suggested SynPEB dataset and segmentation result of PPE, the proposed method.

neighboring points that pass a set of criteria. Hoover et al. [23], for instance, select the points with the highest local planarity score as seeds. During the growing process, they add all adjacent points to the regions that do not exceed a specified maximum difference of normals, Euclidean distance, and orthogonal distance. In contrast, Hähnel et al. [11] choose seeds at random and grow planar polygons by including all neighboring points that do not push the mean squared error of the resulting plane over a given limit.

Deschaud et al. [105] propose an adaption of region growing to large noisy datasets. They compensate for noise by introducing a filter that improves the estimation of endpoint normals, select seeds based on local planarity, and employ a voxel-based variant of region growing. Nurunnabi et al. [106], in turn, address noise by computing endpoint features like normals and curvature via a robust variant of principal component analysis (PCA). In another take on plane extraction from noisy point clouds, Dong et al. [107] combine region growing with energy optimization, where the energy is defined as the sum of geometric errors, spatial coherence, and the total number of planes.

Holz et al. [108] focus on plane extraction for time-sensitive applications. Their method computes normal and curvature estimates not directly based on the point cloud, but based on an approximate mesh. CAPE, an algorithm developed by Proença et al. [109], achieves even higher plane extraction rates at the expense of reduced accuracy. First, the algorithm creates a low-resolution grid, pools the points in each cell, and applies PCA to each cell. CAPE then grows regions composed of cells based on their PCA features.

Inspired by the observation that every line-shaped sequence of points in a laser scan is caused by a planar surface, Jiang et al. [110], Hoover et al. [23], and Cabo et al. [111] apply region growing to line segments instead of points.

As opposed to region growing, clustering extracts planes without the need to find suitable seed points. Trevor et al. [112], for example, assign the same label to adjacent points of an organized range scan if the difference of their normals and their orthogonal distance falls below a given threshold, and subsequently extract planes by clustering points with the same labels. Feng et al. [113] present a clustering algorithm that extracts planes from an organized point cloud with minimal latency. It divides the point cloud uniformly into rectangular point groups, discards all non-planar groups, and subjects the remaining groups to agglomerative hierarchical clustering, using the mean squared orthogonal point-to-plane fitting error as clustering metric. Eventually, it refines the extracted coarse planes by region growing. Marriott et al. [114] also cluster groups of coplanar points based on mean squared error, but instead of using a regular grid to define initial point groups, they propose an expectation-minimization algorithm that fits a Gaussian mixture model to the points.

Pham et al. [115] combine clustering and region growing. They use region growing to oversegment the point cloud and then merge the resulting plane hypotheses via clustering, in each step minimizing an energy function that favors mutually parallel or orthogonal plane pairs.

RANSAC, initially developed by Fischler et al. [116], is a versatile iterative model fitting algorithm. When applied to plane extraction, it selects three laser endpoints at random, fits a plane to them, searches for all points within a certain orthogonal distance, and determines the plane's fitness based on the corresponding point-to-plane distances. This process is repeated until the algorithm finds a plane that satisfies a given minimal fitness. Several works improve on standard RANSAC to overcome its deficiencies. The robust estimator formulated by Gotardo et al. [117] counteracts RANSAC's tendency to disregard small regions. Gallo et al. [118] address the problem of RANSAC often connecting nearby patches that are actually unconnected, for example at steps and curbs. By combining RANSAC with conformal geometric algebra, Sveier et al. [119] perform the least squares fitting necessary to assess the fitness of a plane hypothesis analytically instead of numerically. Alehdaghi et al. [120] present a highly parallelized GPU implementation of RANSAC for plane extraction.

Another general model fitting method, the Hough transform computes for each point in the discretized space of model parameters the fitness of the associated model instance given the data. Vosselman et al. [121] describe how to apply this method to the problem of plane extraction from 3-D point clouds. Oehler et al. [122] present a multi-resolution approach based on both the Hough transform and RANSAC. For a review of further flavors of Hough transform-based plane extraction, the reader is referred to the review composed by Borrmann et al. [123].

11.4 Approach

In this work, we present probabilistic plane extraction (PPE), an approach to extract finite planes from organized 3-D lidar scans. PPE is a maximum-likelihood estimation technique based on agglomerative hierarchical clustering. As a maximum-likelihood estimation technique, it searches for the set of planes that maximize the measurement probability of the given laser scan. As an agglomerative clustering method, it attempts to find this set by creating a plane for each reflection first. This plane explains the corresponding reflection perfectly. PPE then reduces the number of planes by iteratively merging the set of adjacent planes whose merger maintains the highest measurement likelihood of the scan. Clustering ends as soon as a given stopping

criterion is met.

In the following, we first introduce the probabilistic sensor model, on the basis of which we then formulate plane extraction as a maximum-likelihood estimation problem. We describe in detail how our agglomerative hierarchical clustering algorithm solves this optimization problem, and finally explain the pseudocode.

11.4.1 Probabilistic Sensor Model

The sensor model tells the measurement probability of a 3-D lidar scan given a set of planes. We denote the scan $Z := \{z_k\}$, where $k \in \{1, 2, \dots, K\}$ represents the index of a laser ray. A single laser measurement $z := \{s, v, r\}$ is composed of two three-element Cartesian vectors and a scalar: the starting point s of the ray, the normalized direction vector v , and the ray length r . The set of finite planes $L := \{l_j\}$ extracted from the scan consists of a total of J elements. Each plane is represented by a three-element Cartesian support vector x , a three-element Cartesian normal vector n , and a set Q of ray indices: $l := \{x, n, Q\}$. While x and n define the location and orientation of the plane, Q determines its extent. This representation can not only handle convex planes, but also concave planes or planes with holes.

Most lidar sensors exhibit approximately normally distributed noise in radial direction and relatively small angular noise. Consequently, we neglect angular noise and model the distribution of the measured length of a single ray conditioned on a set of planes as a Gaussian probability density function centered at the true ray length:

$$p(z \mid L) = \mathcal{N}(r; \hat{r}(s, v, L), \sigma^2). \quad (11.1)$$

Here, the function $\hat{r}(s, v, L) \in \mathbb{R}^+$ computes the distance between the starting point of the ray and the first intersection of its axis and all planes in L . The standard deviation σ of the radial noise is a function of multiple parameters such as sensor device, reflecting surface, and temperature, but usually not range.

By assuming independence between the individual laser rays, we can derive the measurement probability of the whole scan from equation (11.1) as

$$p(Z \mid L) = \prod_{k=1}^K p(z_k \mid L).$$

To our knowledge, we are the first to apply the above sensor model to plane extraction. Most surveyed works model the measurement probability

of a ray as a zero-centered normal distribution over the shortest distance between the measured ray endpoint and the nearest plane. This heuristic does not account for the ray path, which leads to two undesired effects. First, the nearest plane is not always the one that intersects the ray. Second, the accuracy of the computed distance strongly depends on the incidence angle of the ray.

11.4.2 Maximum-Likelihood Estimation

With the above sensor model, we formulate plane extraction as the following maximum-likelihood estimation problem: Find the set of planes L^* that maximizes the measurement probability of the whole scan $p(Z | L)$. The solution is trivial: For each reflection in the laser scan, create a tiny plane that is not parallel to the ray and that intersects the ray at the measured ray length r . This solution, however, is merely a different representation of the raw lidar data. In order to extract meaningful planes from the scan, we need to reduce the number of planes by constraining the optimization problem. For the following derivation, we choose the maximum number of planes J_{\max} as constraint parameter. Note, however, that our approach allows us just as well to use arbitrary metrics like the maximum mean squared error of the ray radii or the Akaike Information Criterion [124]. Formally, we are confronted with the constrained least squares optimization problem

$$\begin{aligned}
 L^* &= \operatorname{argmax}_L p(Z | L) \Big|_{J(L) \leq J_{\max}} \\
 &= \operatorname{argmin}_L -\log \left(p(Z | L) \right) \Big|_{J(L) \leq J_{\max}} \\
 &= \operatorname{argmin}_L \sum_{k=1}^K \left(r_k - \hat{r}(s_k, v_k, L) \right)^2 \Big|_{J(L) \leq J_{\max}} \\
 &=: \operatorname{argmin}_L E(Z, L) \Big|_{J(L) \leq J_{\max}}
 \end{aligned} \tag{11.2}$$

where $J(L)$ is a function that determines the number of planes in L . The transition from the second to the third line implies our assumption that all rays exhibit the same radial noise. Hereafter, we will refer to E simply as the error of the set of planes L .

Solving (11.2) is primarily a combinatorial problem. Even if we knew the parameters $\{x_j\}$ and $\{n_j\}$ of the planes, we would still not know the data associations $\{Q_j\}$, i.e. which rays belong to which plane. Exhaustively searching the space of all data associations for the combination that maximizes the measurement probability quickly leads to combinatorial explosion

even for small J_{\max} . PPE solves this problem via agglomerative hierarchical clustering.

11.4.3 Agglomerative Hierarchical Clustering

In its generic form, agglomerative hierarchical clustering builds clusters from bottom up: The algorithm first assigns each observation its own cluster and then iteratively merges adjacent pairs of clusters. In each iteration, it decides which pair to merge based on a greedy strategy, always optimizing a specific metric.

Transferred to our case, observations correspond to reflected laser rays, clusters correspond to planes, and the metric the algorithm strives to maximize is the measurement probability $p(Z | L)$, which is equivalent to minimizing the error $E(Z, L)$. Consequently, in the first step, which assigns each observation its own cluster, we assign each laser reflection its own plane. As mentioned above, this plane is not parallel to the ray and intersects the ray at its measured length r . In the following, we call such a plane atomic. We define the support vector of an atomic plane as the endpoint $s + rv$ of the corresponding ray and the normal vector as the ray direction vector v . As opposed to atomic planes, regular planes represent not one, but three or more rays. Therefore, their parameters need to be fitted to the data.

Starting from this trivial maximum-likelihood solution, PPE iteratively reduces the number of planes to J_{\max} by merging adjacent planes. With each merger, the measurement likelihood of the whole scan $p(Z | L)$ decreases, whereas the error $E(Z, L)$ increases by

$$e := E(Z, L'') - E(Z, L') \geq 0, \quad (11.3)$$

where L' and L'' denote the set of planes before and after the merger. Greedy as it is, PPE always opts for the merger that incurs the least error increment, which is equivalent to maintaining maximum measurement likelihood.

Due to ambiguities in the decision process, the formulation above will not yield the desired result yet: The error increment corresponding to merging two or three atomic planes is always zero, because every pair or triple of reflections can be perfectly explained by a single plane. Therefore, given multiple atomic planes, PPE cannot decide which pair or triple to merge. Creating a regular plane out of four atomic planes, however, leads to an overdetermined system of equations, hence a regular plane must be fitted to the four reflections, and the corresponding fitting error constitutes the error increment

$$e_{\text{crt}}(Z, Q) := \min_{x, n} E(\{z_q\}, \{x, n, Q\}), \quad (11.4)$$

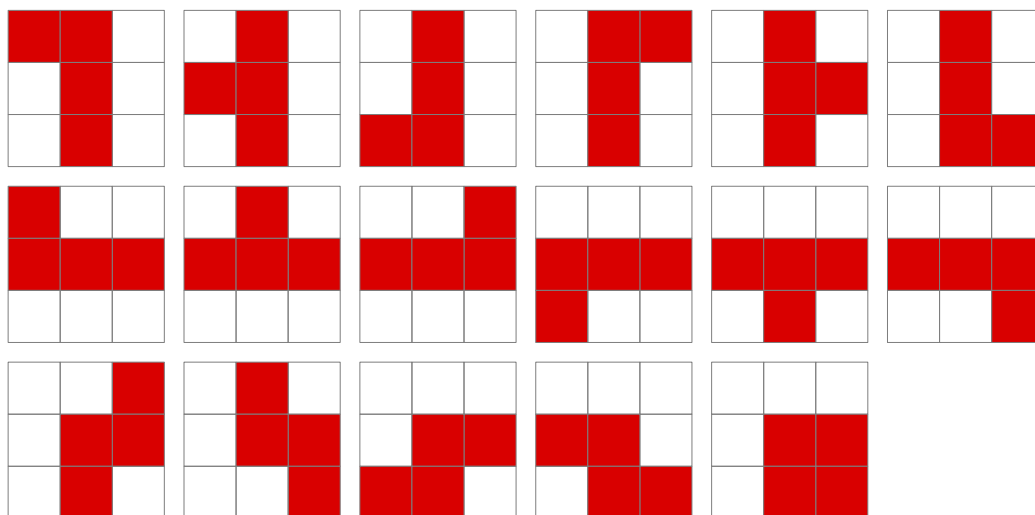


Figure 11.2: All 17 valid tetrominoes that can form a regular plane composed of four atomic planes. The tetrominoes are represented by the red fields. The white fields are neighboring rays that are not assigned to any plane.

where Q denotes the set of ray indices, and where $q \in Q$.

In order to find the combination of four atomic planes that yields the minimum error increment, PPE needs to assess the fitting errors corresponding to all possible combinations. For an atomic plane that resides somewhere in the middle of the grid of laser rays, there are 17 valid ways to combine it with three of its 4-connected neighbors, forming so-called tetrominoes: one O-shaped, four T-shaped, four Z-shaped, and eight L-shaped tetrominoes. Figure 11.2 depicts them all. The I-shaped tetromino is invalid, because fitting a plane to four endpoints in a straight line again yields ambiguous results.

Once the first regular planes emerge, we can identify two more classes of clustering actions apart from merging tetrominoes: extending a regular plane by an atomic plane, and merging two regular planes. Figure 11.3 illustrates all three classes. In each clustering step, PPE must determine the error increment of every possible action, find the one that incurs the least error increment, and merge the respective planes. The error increment of extending a regular plane by an atomic plane amounts to the difference

$$e_{\text{ext}}(Z, Q, k) := e_{\text{crt}}(Z, Q \cup k) - e_{\text{crt}}(Z, Q),$$

where Q denotes the indices of the rays of the regular plane, and where k is the index of the ray corresponding to the atomic plane. Merging two regular

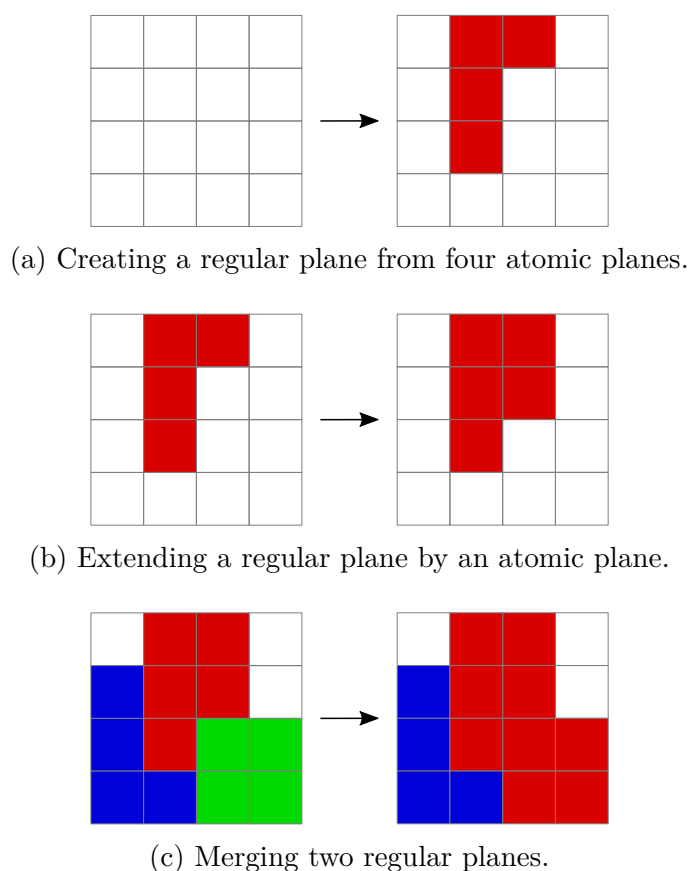


Figure 11.3: Exemplary instances of all three classes of actions PPE can take during clustering in order to reduce the number of planes. White fields stand for atomic planes, fields of the same color except white denote regular planes.

planes indexed i and j adds

$$e_{\text{mrg}} := e_{\text{crt}}(Z, Q_i \cup Q_j) - e_{\text{crt}}(Z, Q_i) - e_{\text{crt}}(Z, Q_j)$$

to the total error $E(Z, L)$.

11.4.4 Probabilistic Plane Extraction

Algorithm 2 provides the PPE pseudocode. Line 1 initializes the set of atomic planes. The function $\text{crt}(Z, L)$ in line 2 loops over all valid tetrominoes of atomic planes and returns the minimum error e_{crt} along with the associated indices Q_{crt} . As there are no regular planes which could be extended or merged at this point, line 3 sets the corresponding error increments e_{ext} and e_{mrg} to infinity. After these initializations, the algorithm starts iteratively

Algorithm 2: Probabilistic Plane Extraction

Data: Z, J_{\max} **Result:** L

```

1  $L \leftarrow \{s_k + r_k v_k, v_k, k\}, k \in \{1, 2, \dots, K\}$ 
2  $(e_{\text{crt}}, Q_{\text{crt}}) \leftarrow \text{crt}(Z, L)$ 
3  $e_{\text{ext}} \leftarrow e_{\text{mrg}} \leftarrow \infty$ 
4 while  $J(L) > J_{\max}$  do
5   if  $e_{\text{crt}} = \min(e_{\text{crt}}, e_{\text{ext}}, e_{\text{mrg}})$  then
6      $L \leftarrow L \cup \text{fit}(Z, Q_{\text{crt}})$ 
7      $L \leftarrow \text{rma}(L, Q_{\text{crt}})$ 
8   else if  $e_{\text{ext}} = \min(e_{\text{crt}}, e_{\text{ext}}, e_{\text{mrg}})$  then
9      $L_j \leftarrow \text{fit}(Z, Q_j \cup k)$ 
10     $L \leftarrow \text{rma}(L, \{k\})$ 
11   else
12      $L_j \leftarrow \text{fit}(Z, Q_i \cup Q_j)$ 
13      $L \leftarrow L \setminus L_i$ 
14   end
15 end
16 end
17  $(e_{\text{crt}}, Q_{\text{crt}}) \leftarrow \text{crt}(Z, L)$ 
18  $(e_{\text{ext}}, j, k) \leftarrow \text{ext}(Z, L)$ 
19  $(e_{\text{mrg}}, i, j) \leftarrow \text{mrg}(Z, L)$ 
20 end

```

reducing the number of planes. In the first iteration, it always creates a regular plane out of four atomic ones. This means it first adds the new plane to the map (line 6) and then removes the merged atomic planes (line 7). Here, the function $\text{fit}(Z, Q)$ fits a plane l^* to the rays indexed by Q :

$$l^* := \text{fit}(Z, Q) := \left\{ \underset{x,n}{\text{argmin}} E(\{z_q\}, \{x, n, Q\}), Q \right\},$$

whereas $\text{rma}(L, Q)$ removes the atomic planes indexed by Q from L and returns the updated plane set. After every manipulation of the plane map, lines 17 to 19 recompute the error increments of all merging options. To that end, $\text{ext}(Z, L)$ iterates over all possible extensions of all regular planes in L and finds the minimum error increment e_{ext} associated with extending plane j by ray k . Similarly, mrg evaluates for all pairs of neighboring regular planes the hypothetical error increments incurred by merging them and returns the minimum e_{mrg} , which corresponds to merging planes i and j . Lines 9 and 10 update the map during an extension step, while lines 12 and 13 come into play when two regular planes are merged.

For the sake of clarity, algorithm 2 is not optimized. For an optimized version of PPE, please refer to our MATLAB implementation [104], which features several algorithmic optimizations, optional GPU acceleration, multiple stopping criteria, and a geometric outlier filter.

11.5 Experiments

In order to compare PPE with the state of the art, we conduct two series of experiments. In the first series, we evaluate PPE using the popular SegComp plane extraction benchmark [23]. The deficiencies of this dataset motivated us to create SynPEB, the first publicly available synthetic plane extraction benchmarking dataset, on which we base the second experiment series.

SegComp comprises two collections of organized point clouds, which depict compositions of polyhedral objects on a tabletop. They were recorded by an ABW structured light sensor and by a Perceptron laser scanner, respectively. Due to the fact that our measurement model, defined in equation (11.1), is specifically designed for laser sensors, we evaluate our method on the Perceptron collection only. This dataset is divided into 10 training scans and 30 testing scans. We use the former to determine the optimum values of e and d , the two parameters of the specific PPE version we use in both experiment series. The parameter e , defined in equation (11.3), denotes the maximum admissible error increment in a clustering step and serves as stopping criterion. In order to compensate for the high level of noise present

in all Perceptron scans, we incorporate a geometric outlier filter in PPE, which prevents clustering neighboring points if their Cartesian distance exceeds a certain threshold d . To find suitable values for both parameters, we maximize the fraction of correctly segmented planes over a regular grid in e and d .

The upper part of table 11.1 shows the corresponding experimental results for PPE and compares them to all previous works evaluated on the Perceptron dataset using the performance metrics defined by Hoover et al. [23]. In order to increase the relevance of the results, we suggest two additional metrics: the k -value and the RMSE. The k -value is defined as

$$k := \frac{\sum_{j=1}^{J(L)} \hat{K}(l_j)}{K}, \quad (11.5)$$

where $\hat{K}(l)$ is a function that takes an extracted plane l as input, checks if this plane is correctly segmented using the 80% threshold proposed by Hoover et al., and returns the number of points of the corresponding ground truth plane. If the input plane is not correctly segmented, the function returns zero. In this way, k indicates the portion of the point cloud that the algorithm correctly segments into planes. The root mean squared error RMSE, defined as

$$\text{RMSE} := \sqrt{\frac{E(Z, L)}{J(L)}}, \quad (11.6)$$

complements k by providing an estimate of how accurately the extracted planes represent the point cloud.

In addition to quoting the numbers of previous works and stating our results for PPE, we evaluate MSAC and PEAC. MSAC is a baseline approach based on the RANSAC variant proposed by Torr et al. [125]. Beginning with the input point cloud, this method iteratively detects a plane and removes the inlier points from the cloud until a specified fraction of the original number of points remains. PEAC – plane extraction using agglomerative clustering – refers to the open-source implementation [126] Feng et al. provide to complement their paper [113]. We are not able to exactly replicate the SegComp results they quote in their paper. Nevertheless, we state our findings for SegComp in order to establish comparability between our PEAC results across both experiment series. Analogously to PPE, we determine the optimum parameters for MSAC and PEAC via grid search. The exact parameter sets for all methods can be found at [104]. Even with these parameters, both MSAC and PEAC return a single false plane detection when processing all

testing scans of SegComp, which leads to exploding RMSE-values. To mitigate this effect, the RMSE-values in table 11.1 are based on all planes with $\text{RMSE} \leq 10$ m each.

Although PPE is designed for maximum accuracy, our method achieves only average results on SegComp. The reasons lie in the peculiarities of the dataset. Figure 11.4a reveals that the rays that hit an object face at an obtuse angle are much more strongly affected by noise than rays with acute incidence angles, creating the impression that faces with obtuse incidence angles extend in a curved fashion beyond their borders. Another issue becomes apparent when closely inspecting the ground plane: Labeling is based on the geometry of the underlying objects, not on the output of the miscalibrated sensor. The khaki tabletop plane and the purple topside of the octagon in the point cloud in figure 11.4a, for example, exhibit kinks due to systematic errors in the lidar calibration. The labelers, knowing that these planes were flat, labeled both as contiguous planes. PPE, without knowledge about the real scene, splits each plane into two. Although desirable, this behavior results in the highest oversegmentation rate among all methods and decreases both the percentage of correctly segmented planes and the k -value.

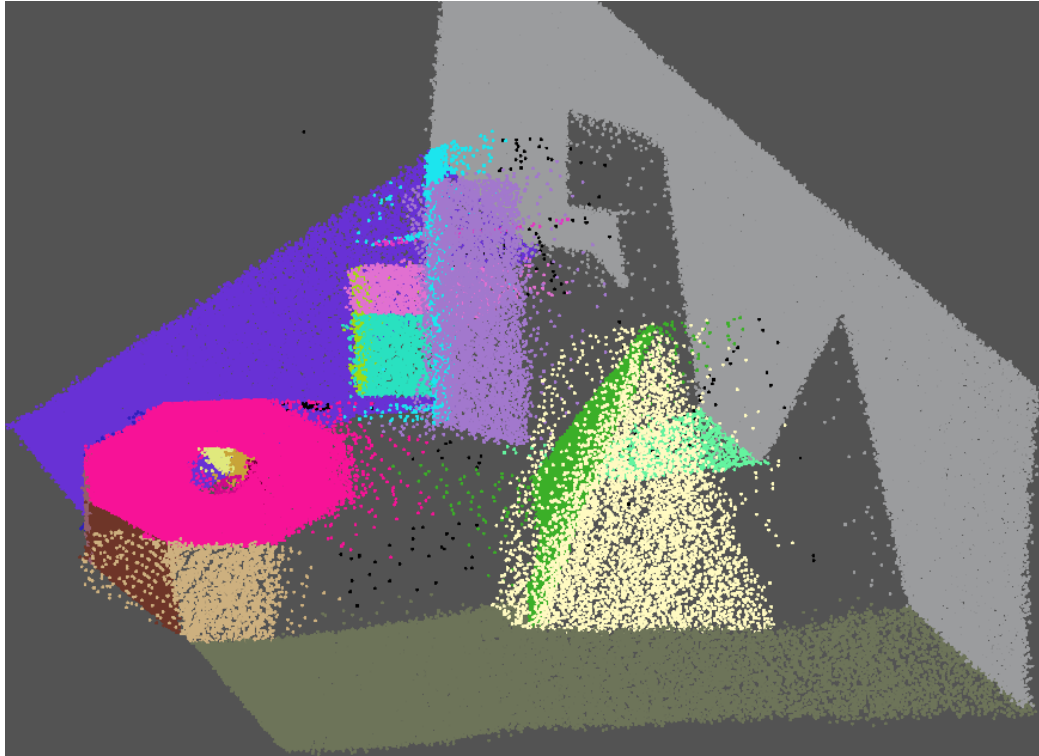
In order to prove that the ground-truth labeling of SegComp is indeed not optimal, we compare the RMSE-values of the ground-truth segmentation to those of PPE. This time, PPE is configured to extract as many planes from a scan as there are present in the ground truth. On average, the resulting RMSE-values are 3.2 % lower than those corresponding to ground truth. A t -test over all scans yields a p -value of 12.9 %, which means that the probability of PPE returning a more accurate segmentation than ground truth is as high as 87.1 %.

Similarly to the ground-truth segmentation, the ground-truth angles between adjacent planes were presumably determined based on the underlying data, too: They are provided as integers rather than as floating-point numbers.

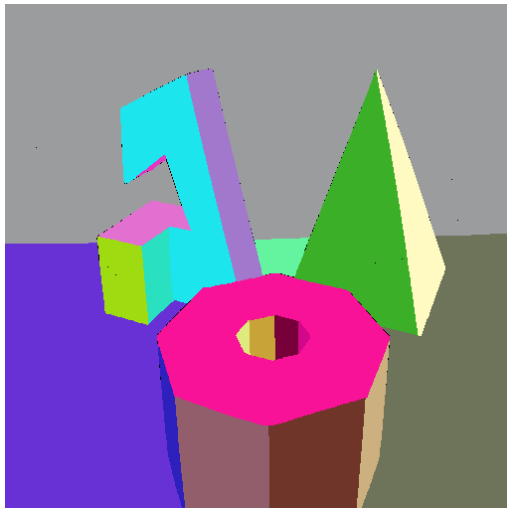
As the aforementioned problems with SegComp bias the evaluation and because there is no publicly available alternative, we created a synthetic plane extraction benchmarking dataset, in short SynPEB, which we use as the basis of the second experiment series. Like PPE and the implementation of all our experiments, the SynPEB scans and the corresponding sampling engine can be downloaded at [104]. The SynPEB world consists of a room of approximately $6 \text{ m} \times 7 \text{ m} \times 3 \text{ m}$ populated with various polyhedral objects, resulting in 42.6 planes of different shapes and sizes per scan. Analogously to SegComp, we divide the dataset into 10 training scans and 30 testing scans, provided as organized point clouds of 500×500 measurements. These scans are affected by normally distributed angular noise with standard deviation

Method	f [%]	k [%]	RMSE [mm]	α [°]	n_o	n_u	n_m	n_s
SegComp Perceptron dataset								
USF [23]	60.9	–	–	2.7	0.4	0.0	5.3	3.6
WSU [23]	40.4	–	–	3.3	0.5	0.6	6.7	4.8
UB [23]	65.7	–	–	3.1	0.6	0.1	4.2	2.8
UE [23]	68.4	–	–	2.6	0.2	0.3	3.8	2.1
UFPR [117]	75.3	–	–	2.5	0.3	0.1	3.0	2.5
Oehler et al. [122]	50.1	–	–	5.2	0.3	0.4	6.2	3.9
Holz et al. [108]	75.3	–	–	2.6	0.4	0.2	2.7	0.3
RPL-GMR [114]	72.4	–	–	2.5	0.3	0.3	3.0	2.0
Feng et al. [113]	60.9	–	–	2.4	0.2	0.2	5.1	2.1
PEAC [126]	48.6	91.3	2.6	2.6	0.0	0.1	7.1	2.0
MSAC [125]	18.5	76.7	3.4	3.9	0.1	0.2	11.3	3.4
PPE (proposed)	60.7	61.2	2.9	2.8	1.4	1.1	1.5	2.3
SynPEB dataset								
PEAC [126]	29.1	60.4	28.6	–	0.7	1.0	26.7	7.4
MSAC [125]	7.3	35.6	34.3	–	0.3	1.0	36.3	10.9
PPE (proposed)	73.6	77.9	14.5	–	1.5	1.1	7.1	16.5

Table 11.1: Results of both experiment series. The header variables f and α denote the fraction of correctly segmented planes and the mean angular deviation, averaged over all testing scans, while n_o , n_u , n_m , and n_s represent the absolute numbers of oversegmented, undersegmented, missing, and spurious planes compared to the ground-truth segmentation. The metrics k and RMSE are defined in equation (11.5) and (11.6), respectively. On average, each scan of the SegComp dataset contains 14.6 ground-truth planes, while each scan of the SynPEB dataset is composed of 42.6 planes.



(a) 3-D point cloud colored according to ground-truth segmentation.



(b) Ground-truth segmentation.



(c) PPE segmentation.

Figure 11.4: Point cloud and segmentation images of scan `perc.test.23` of the SegComp dataset. Outliers are colored black.

$\sigma_{\text{ang}} = 1$ mdeg and by normally distributed radial noise with $\sigma_{\text{rad}} = 20$ mm. Figure 11.1 conveys an intuition of what a SynPEB scan looks like.

The lower part of table 11.1 shows the plane extraction results for PEAC, MSAC, and PPE on SynPEB. For the other approaches, there is no working implementation publicly available. When comparing the results across datasets, we observe that the fraction of planes detected by both PEAC and MSAC is considerably lower on SynPEB than on SegComp. The high numbers of missed planes indicate that the most likely cause is the challenging nature of SynPEB: At almost identical resolutions, SynPEB contains almost three times as many planes per scan as SegComp. Nevertheless, both the percentage of correctly identified planes and the k -value of the PPE results have increased significantly. The RMSE-value for PPE is 28% lower than the radial sensor noise, demonstrating that the method is able to leverage the high number of data points per plane to accurately reconstruct the underlying data.

PPE’s high accuracy comes at a price: On average, processing a 500×500 scan using our open-source implementation takes 1.6 h on a single core of an Intel Xeon CPU with 2.6 GHz, while our MATLAB implementation of MSAC needs 1.1 s. As a method specifically developed to enable real-time plane extraction, PEAC runs at approximately 30 Hz on an Intel i7-7700K processor.

11.6 Conclusion and Future Work

Many authors have investigated the problem of extracting planes from 3-D laser scans and proposed solutions. The present paper sets itself apart in two ways. First, it proposes PPE, an approach to plane extraction that builds upon an accurate probabilistic sensor model instead of the conventional point-to-plane distance heuristic. Our experiments demonstrate that the accuracy of the sensor model translates to superior plane reconstruction results. Second, motivated by the deficiencies of the popular plane extraction benchmark SegComp, we suggest an alternative benchmark, dubbed SynPEB. Both the implementation of the proposed algorithm and the suggested dataset are available online [104].

Due to the promising results, we plan several extensions of PPE. First of all, we will decrease the runtime to enable online plane extraction. In addition, we will relax the requirement that the point cloud is organized, and investigate whether leveraging laser remission intensity information can further improve the results.

Chapter 12

Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans

The work replicated below, written by Alexander Schaefer, Daniel Büscher, Johan Vertens, Lukas Luft, and Wolfram Burgard was accepted for the European Conference on Mobile Robotics 2019, and presented during the conference, which took place in Prague, Czech Republic, from September 4 to 6, 2019. The IEEE holds the copyright on the article: © 2019 IEEE. Reprinted, with permission, from “Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans”.

12.1 Abstract

Due to their ubiquity and long-term stability, pole-like objects are well suited to serve as landmarks for vehicle localization in urban environments. In this work, we present a complete mapping and long-term localization system based on pole landmarks extracted from 3-D lidar data. Our approach features a novel pole detector, a mapping module, and an online localization module, each of which are described in detail, and for which we provide an open-source implementation [27]. In extensive experiments, we demonstrate that our method improves on the state of the art with respect to long-term reliability and accuracy: First, we prove reliability by tasking the system with localizing a mobile robot over the course of 15 months in an urban area based on an initial map, confronting it with constantly varying routes, differ-

ing weather conditions, seasonal changes, and construction sites. Second, we show that the proposed approach clearly outperforms a recently published method in terms of accuracy.

12.2 Introduction

Intelligent vehicles require accurate and reliable self-localization systems. Accurate, because an exact pose estimate enables complex functionalities such as automatic lane following or collision avoidance in the first place. Reliable, because the quality of the pose estimate must be maintained independently of environmental factors in order to ensure safety.

Satellite-based localization systems like RTK-GPS or DGPS seem to be an efficient solution, since they achieve centimeter-level accuracy out of the box. However, they lack reliability. Especially in urban areas, buildings that obstruct the line of sight between the vehicle and the satellites can decrease accuracy to several meters [127, 5]. Localization on the basis of dense maps like grid maps, point clouds, or polygon meshes represents a more reliable alternative [128]. On the downside, dense approaches require massive amounts of memory that quickly become prohibitive for maps on larger scales. This is where landmark maps come into play: By condensing billions of raw sensor data points into a comparably small number of salient features, they can decrease the memory footprint by several orders of magnitude [129].

In this work, we present an approach to long-term 2-D vehicle localization in urban environments that relies on pole landmarks extracted from mobile lidar data. Poles occur as parts of street lamps, traffic signs, as bollards and tree trunks. They are ubiquitous in urban areas, long-term stable and invariant under seasonal and weather changes. Since their geometric shape is well-defined, too, poles are well suited to serve as landmarks that enable accurate and reliable localization.

Our localization process is subdivided into a mapping and a localization phase. During mapping, we use the pole detector presented below to extract pole landmarks from lidar scans and register them with a global map via a given ground-truth vehicle trajectory. During localization, we employ a particle filter to estimate the vehicle pose by aligning the pole detections from live sensor data with those in the map. Figure 12.1 shows an exemplary localization result.

We are not the first ones to propose this kind of localization technique: The next section provides an overview over the numerous related works. However, to the best of our knowledge, we are the first ones to present a pole detector that does not only consider the laser ray endpoints, but also the

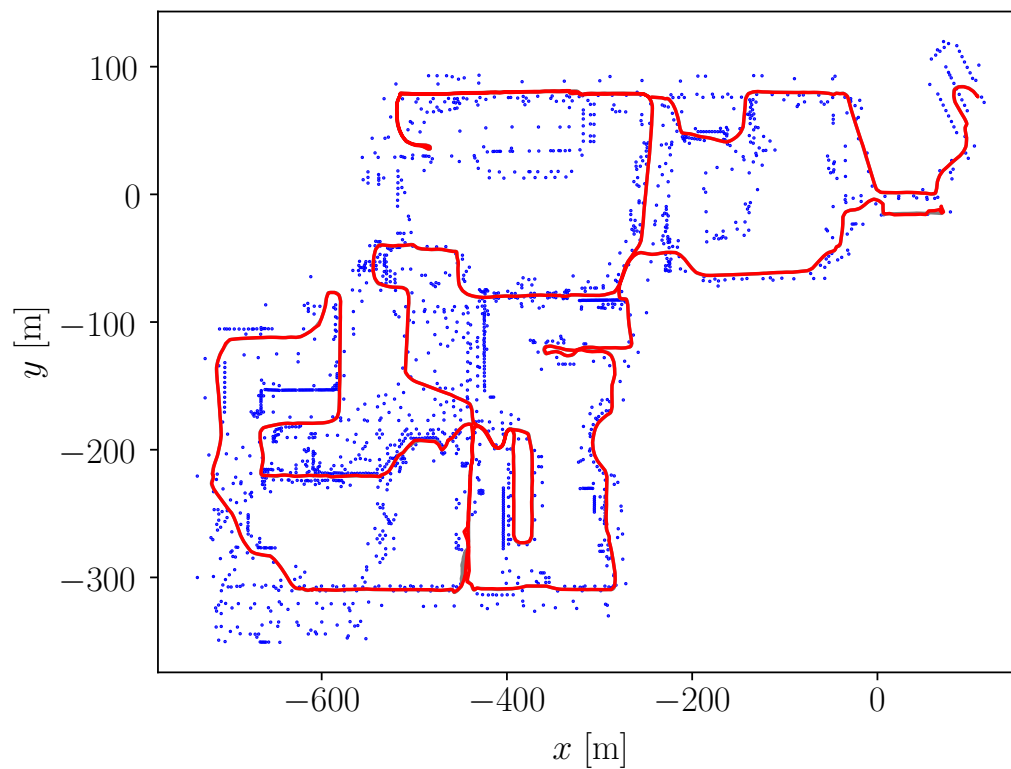


Figure 12.1: Pole landmark map created from the NCLT dataset [5] and trajectory of an experimental run 15 months after map creation. The blue dots represent the landmarks. The gray line corresponds to the ground-truth trajectory. Most of it is covered by the red line, which represents the estimate produced by the presented method. The mean position difference between both trajectories, formally defined in section 12.5.1, amounts to 0.31 m.

free space in between the laser sensor and the endpoints, and to demonstrate reliable and accurate vehicle localization based on a map of pole landmarks on large time scales. While related works usually evaluate localization performance on a short sample trajectory of at most a few minutes length, we successfully put our approach to the test on a publicly available long-term dataset that contains 35 hours of data recorded over the course of 15 months – including varying routes, construction zones, seasonal and weather changes, and lots of dynamic objects. Additional control experiments show that the presented method is not only reliable, but significantly outperforms a recently published state-of-the-art approach in terms of accuracy, too.

12.3 Related Work

In recent years, a number of authors have addressed the specific question of vehicle localization via pole landmarks extracted from lidar scans. Any solution to this question consists of at least two parts: a pole detector and a landmark-based pose estimator. The detector developed by Weng et al. [24], for example, tessellates the space around the lidar sensor and counts the number of laser reflections per voxel. Poles are then assumed to be located inside contiguous vertical stacks of voxels that all exceed a reflection count threshold. In order to extract the pole parameters from these clusters, the detector fits a cylinder to all points in a stack via RANSAC [116]. For 2-D pose estimation, the authors employ on a particle filter with nearest-neighbor data association. Sefati et al. [25] present a pole detector that removes the ground plane from a given point cloud, projects the remaining points onto a horizontal regular grid, clusters neighboring cells based on occupancy and height, and fits a cylinder to each cluster. Like Weng et al., Sefati et al. obtain their 2-D localization estimate from a particle filter that performs nearest-neighbor data association. Kümmerle et al. [129] make use of Sefati et al.'s pole detector, but to further refine the localization estimate, they also fit planes to building façades in the laser scans and lines to lane markings in stereo camera images. Like the above works, their pose estimator relies on a Monte Carlo method to solve the data association problem, but uses optimization to compute the most likely pose. More specifically, in the data association stage, it builds a local map by accumulating the landmarks detected over the past timesteps based on odometry. It then samples multiple poses around the current GPS position, uses these pose hypotheses to project the local map into the global map, and identifies the most probable hypothesis via a handcrafted landmark matching metric. Given the resulting data associations, it refines the current vehicle pose estimate via nonlinear least

squares optimization over a graph of past vehicle poses and landmarks.

Spangenberg et al. [130] extract pole landmarks not from lidar scans, but from stereo camera images. In order to estimate the vehicle pose, they feed wheel odometry, GPS data, and online pole detections to a particle filter.

While the approaches above all provide a complete localization system consisting of a pole extractor and a landmark-based localization module, there exist a variety of research papers that focus solely on pole extraction. Extracting poles from lidar data is a common problem in road infrastructure maintenance and urban planning. In this domain, researchers are not only interested in fitting geometric primitives to the data and determining pole coordinates, but also in precise point-wise segmentation. Brenner [131], Cabo et al. [132], Tombari et al. [133], and Rodriguez et al. [134] present different methods to extract pole-like objects from point clouds, i.e. without accounting for free space information. The approaches of Yu et al. [135] and Wu et al. [136] specifically target street lamp poles, while Zheng et al. [137] provide a solution to detect poles that are partially covered by vegetation. Yokoyama et al. [138] not only extract poles, but they classify them as lamp posts, utility poles, and street signs. Ordóñez et al. [139] build upon the pole detector proposed by Cabo et al. [132] and classify the results into six categories, including trees, lamp posts, traffic signs, and traffic lights. Li et al. [140] take classification one step further by decomposing multifunctional structures, for example a light post carrying traffic signs, into individual elements.

Poles are not the only landmarks suitable for vehicle localization. Qin et al. [141] investigate Monte Carlo vehicle localization in urban environments based on curb and intersection features. As demonstrated by the works of Schindler [142] and Schreiber et al. [143], road markings as landmarks can also yield high localization accuracy. Hata and Wolf [144] feed both curb features and road markings to their particle filter. Welzel et al. [145] explore the idea of using traffic signs as landmarks. Although traffic signs occur less frequently in urban scenarios compared to other types of road furniture like road markings or street lamp poles, they offer the advantage of not only encoding a position, but also an unambiguous ID. Finally, Im et al. [146] explore urban localization based on vertical corner features, which appear at the corners of buildings, in monocular camera images and lidar scans.

12.4 Approach

The proposed 2-D vehicle localization system consists of three modules: the pole extractor, the mapping module, and the localization module. During

the initial mapping phase, the pole extractor reduces a given set of lidar scans to pole landmarks. The mapping module then uses the ground-truth sensor poses to build a global reference map of these landmarks. During the subsequent localization phase, the pole extractor processes live lidar data and passes the resulting landmarks to the localization module, which in turn generates a pose estimate relative to the global map. In the following, we detail each of these modules and their interactions.

12.4.1 Pole Extraction

The pole extraction module takes a set of registered 3-D lidar scans as input and outputs the 2-D coordinates of the centers of the detected poles with respect to the ground plane, along with the estimated pole widths. To that end, it builds a 3-D occupancy map of the scanned space, applies a pole feature detector to every voxel, and regresses the resulting pole map to a set of pole position and width estimates.

To describe these three steps mathematically, we denote a single laser measurement – a ray – by $z := \{u, v\}$, where u and v represent its Cartesian starting point and endpoint, respectively. All measurements $Z := \{z_i\}$ are assumed to be registered with respect to the map coordinate frame, whose x - y plane is aligned with the ground plane. The measurements can be taken at different points in time, but the timespan between the first and the last measurement needs to be sufficiently small in order not to violate our assumption that the world is static. Now, we tessellate the map space, trace the laser rays, and model the posterior probability that the j -th voxel reflects an incident laser ray according to Luft et al. [45] by

$$p(\mu_j | Z) = \text{Beta}(h_j + \alpha, m_j + \beta).$$

Here, h_j and m_j denote the numbers of laser reflections and transmissions in the j -th cell, whereas α and β are the parameters of the prior reflection probability $p(\mu_j) = \text{Beta}(\alpha, \beta)$, which we determine in accord with [45] by

$$\alpha = -\frac{\gamma(\gamma^2 - \gamma + \delta)}{\delta},$$

$$\beta = \frac{\gamma - \delta + \gamma\delta - 2\gamma^2 + \gamma^3}{\delta},$$

where $M := \{h_j(h_j + m_j)^{-1}\}$ denotes the maximum-likelihood reflection map, and where $\gamma := \mathbf{E}[M]$, $\delta := \text{var } M$ represent its mean and variance, respectively. Please note that $\{p(\mu_j | Z)\}$ is a full posterior map: In contrast to M , which assigns each voxel the most probable reflection rate, it yields a posterior distribution over every reflection rate possible.

Since we want to extract poles based on occupancy probability, not on reflection rate, we convert $\{p(\mu_j | Z)\}$ to an occupancy map $O := \{o_j\}$. Assuming that a cell is occupied if its reflection rate exceeds a threshold μ_o , we formulate the occupancy probability by integration:

$$o_j := \int_{\mu_o}^1 p(\mu_j | Z) d\mu_j.$$

Next, a pole feature detector transforms O to a 2-D map of pole scores S in the ground plane. Each pixel of S encodes the probability that a pole is present at the corresponding location. The transformation from O to S follows a set of heuristics that are based on the definition of a pole as a vertical stack of occupied voxels with quadratic footprint, laterally surrounded by a hull of free voxels. First, we create a set of intermediate 3-D score maps of the same size as O , each denoted by $Q_a := \{q_{a,j}\}$. Every cell $q_{a,j}$ tells how probable it is that this portion of space is part of a pole with edge length a , where $a \in \mathbb{N}^+$ is measured in units of grid spacing:

$$q_{a,j} := \max_{k \in \text{inside}(j,a)} \left(\frac{\sum_{l \in \text{inside}(k,a)} o_l}{a^2} - \max_{l \in \text{outside}(k,a,f)} o_l \right).$$

Here, $\text{inside}(j, a)$ and $\text{outside}(j, a, f)$ are functions that, given a map index j and a pole width a , return a set of indices into voxels in the same horizontal map slice as j . While the former outputs the indices of all voxels inside the pole, the latter returns the indices corresponding to the supposedly free region around the pole with thickness $f \in \mathbb{N}^+$. Both functions assume that the lower left lateral walls of the pole are aligned with the lower left lateral sides of the j -th voxel. With these definitions, the argument of the enclosing maximum operator amounts to the difference between the mean occupancy value inside the pole and the maximum occupancy value of the volume of free space around the pole. The resulting score lies in the interval $[-1, 1]$: the higher the score, the greater the probability that the corresponding partition of space is part of a pole. Second, we regress from the resulting 3-D maps $\{Q_a\}$ to 2-D by merging them into a single map $Q := \{q_j\} := \{\max_a q_{a,j}\}$ and by determining for each horizontal position in Q the contiguous vertical stack of voxels that all surpass a given score threshold q_{\min} . After discarding all stacks that fall below a certain height threshold h_{\min} and computing the mean score for each of the remaining stacks, we obtain the desired 2-D score map S .

Finally, we convert this discrete score map to a set of continuous pole position and width estimates. We identify the pole positions as the modes of S , which we determine via mean shift [147] with a Gaussian kernel and with the local maxima of S as seed points. The width estimate of each pole

is computed as the weighted average over all pole widths a , where for every a , the weight is the mean of all cells in Q_a that touch the pole.

The presented algorithm differs from other pole extractors in the fact that it is based on ray tracing. By considering not only the scan endpoints, but also the starting points, it explicitly models occupied and free space. In contrast, most other methods assume the space around the sensor to be free as long as it does not register any reflections. The absence of reflections, however, can have two reasons: The respective region is in fact free, or the lidar sensor did not cover region due to objects blocking its line of sight or its limited range.

12.4.2 Mapping

In theory, the global reference map could be built by simply applying the pole extractor to a set of registered laser scans that cover the area of interest. In practice, the high memory complexity of grid maps and laser scans often renders this naïve approach infeasible. To create an arbitrarily large landmark map with limited memory resources, we partition the mapping trajectory into shorter segments of equal length and feed the lidar measurements taken along each segment to the pole extractor one by one. For the sake of consistency, we take care that the intermediate local grid maps are aligned with the axes of the global map and that all of them have the same raster spacing. The intermediate maps, whose sizes are constant and depend on the sensor range, usually fit into memory easily. Processing all segments provides us with a set of pole landmarks. If the length of a trajectory segment is smaller than the size of a local map, the local maps overlap, a fact that can lead to multiple landmarks representing a single pole. In order to merge these ambiguous landmarks, we project all poles onto the ground plane, yielding a set of axis-aligned squares. If multiple squares overlap, we reduce them to a single pole estimate by computing a weighted average over their center coordinates and widths. Each weight equals the mean pole score, which we determine by averaging over the scores of all voxels that touch the pole in all score maps Q_a . If there is no overlap, we integrate the corresponding pole into the global reference map without further ado.

As a side benefit, this mapping method allows us to filter out dynamic objects at the landmark level using a sliding-window approach: A local landmark is integrated into the reference map only if it was seen at least c times in the past w local maps, where $c \leq w$; $c, w \in \mathbb{N}^+$. Correspondences between landmarks are again determined via checking for overlapping projections in the ground plane.

12.4.3 Localization

During online localization, we continuously update the vehicle pose based on the collected odometry measurements and periodically correct the estimate by matching online pole landmarks, which we extract from the most recent local map, against the reference map. We build the local map by accumulating laser scans along a segment of the trajectory and by registering them via odometry. To filter out dynamic objects, we apply the sliding-window approach described in the previous section.

A particle filter is well suited for the localization task [10], because it can not only maintain multiple pose hypotheses in parallel, but also handle global localization. At time t , each particle corresponds to a 2-D vehicle pose hypothesis, represented by the 3×3 homogeneous transformation matrix X_t . To perform the motion update, we assume Gaussian motion noise Σ and sample from a trivariate normal distribution in χ :

$$X_t = \text{transform}(\xi) X_{t-1} \quad \Big| \quad \xi \sim \mathcal{N}(\chi, \Sigma),$$

where $\chi := [x, y, \phi]^\top$ denotes the latest relative odometry measurement, with x , y , and ϕ representing the translation and the heading of the vehicle, respectively. The function $\text{transform}([x, y, \phi]^\top)$ converts the input vector to the corresponding 3×3 transformation matrix. In each measurement update, we determine the data associations between the online landmarks $\Lambda := \{\lambda_k\}$ and the landmarks in the reference map $L := \{l_n\}$ via nearest-neighbor search in a k -D tree, assume independence between the elements of Λ , and update the particle weights according to the measurement probability

$$p(\Lambda \mid X, L) = \prod_k p(\lambda_k \mid X, l_{n(k)}),$$

where $n(k)$ is the data association function that tells the index of the reference landmark associated with the k -th online landmark. To evaluate the above equation, we need to define a measurement model

$$p(\lambda_k \mid X, l_{n(k)}) := \mathcal{N}(\|X\lambda_k - l_{n(k)}\|, \sigma) + \epsilon,$$

with the reference and online landmarks represented by homogeneous 2-D position vectors $[x, y, 1]^\top$, and where we assume isotropic position uncertainty σ of the reference landmarks. The constant addend $\epsilon \in \mathbb{R}^+$ accounts for the probability of discovering a pole that is not part of the map. This probability can be estimated by generating a global map from one run, generating a set of local maps from data recorded on the same trajectory in a second run, and computing the numbers of matched and unmatched landmarks.

12.5 Experiments

In order to evaluate the proposed localization system, we perform two series of experiments. The complete implementation is publicly available [27]. In the first series, we assess the system’s long-term localization reliability and accuracy on the NCLT dataset [5]. While these experiments provide profound insights into the performance of the developed method, the results are absolute and do not allow direct comparisons with other methods, because to the best of our knowledge, we are the first to test landmark-based localization on NCLT. For this reason, we base the second experiment series on the KITTI dataset [26]. That allows us to repeat the experiments performed by the authors of another state-of-the-art localization method, only that this time, we use the system presented above.

12.5.1 Localization on the NCLT Dataset

The NCLT (North Campus Long-Term) dataset [5] was acquired with a two-wheeled Segway robot on one of the campuses of the University of Michigan, USA. The data is perfectly suited for testing the capabilities of any system that targets long-term localization in urban environments: Equipped with a Velodyne HDL-32E lidar, GPS, IMU, wheel encoders, and a gyroscope, among others, the robot recorded 27 trajectories with an average length of 5.5km and an average duration of 1.3h over the course of 15 months. The recordings include different times of day, different weather conditions, seasonal changes, indoor and outdoor environments, lots of dynamic objects like people and moving furniture, and two large construction projects that evolve constantly. Although the routes differ significantly between sessions, the trajectories have a large overlap.

The main difference between NCLT and the data used to evaluate all other pole-based localization methods we surveyed lies in its extent: While related works briefly demonstrate the plausibility of their approaches by evaluating localization performance on datasets with durations between 46 s and 30 min, we focus on long-term reliability and accuracy and process 35 h of data spread over more than one year.

Before localizing, we build a reference map of the poles on the campus. To that end, we feed the laser scans and the ground-truth robot poses of the very first session to our mapping module. Unfortunately, the ground truth provided by NCLT is not perfect. It consists of optimized poses spaced in intervals of 8m, interpolated by odometry. Consequently, point clouds accumulated over a few meters exhibit considerable noise, as illustrated in figure 12.2. For that reason, we set the distance of the trajectory segments to

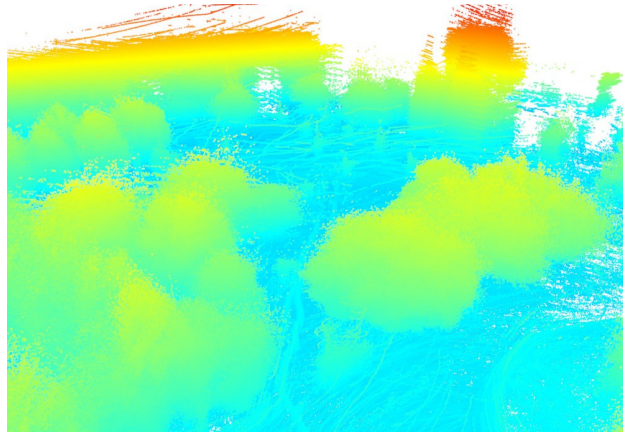
build local maps to 1.5 m, the raster spacing of the grid maps to 0.2 m, and the occupancy threshold to $\mu_o = 0.2$. During mapping and localization, the pole extractor discards all poles below a minimum pole height of $h_{\min} = 1$ m and below a minimum pole score of $q_{\min} = 0.6$. The extent of the local maps is chosen $30 \text{ m} \times 30 \text{ m} \times 5 \text{ m}$ in x , y , and z of the map frame, respectively. Figure 12.3 illustrates the corresponding results.

Although the first session covers most of the campus, the robot occasionally roams into unseen regions during later sessions. For that reason, we iterate over all subsequent sessions, too, but add landmarks to the global map only if the corresponding laser scans are recorded at a minimum distance of 10 m from all previously visited poses. Table 12.1 shows that after the second session, the fractions of scans per session that contribute to the map drop to $f_{\text{map}} \leq 5.5\%$.

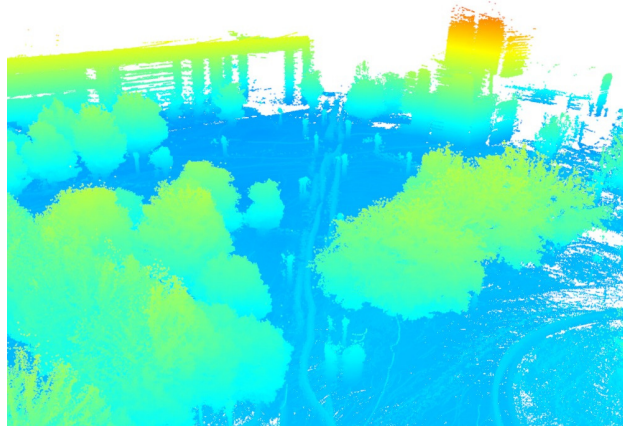
During localization, odometry mean and covariance estimates are generated by fusing wheel encoder readings, gyroscope, and IMU data in an extended Kalman filter. The particle filter contains 5000 particles, which we initialize by uniformly sampling positions in a circle with radius 2.5 m around the earliest ground-truth pose. The headings are uniformly sampled in $[-5^\circ, 5^\circ]$. To maximize reliability, we inflate the motion noise by a factor of four, which corresponds to doubled standard deviation, define the position uncertainty of the poles in the global map as $\sigma = 1 \text{ m}^2$, and set the addend in the measurement probability density to $\epsilon = 0.1$. We resample particles whenever the number of effective particles $n_{\text{eff}} := (\sum_i w_i^2)^{-1} < 0.5$, where w_i is the weight of the i -th particle, via low-variance resampling as described by Thrun et al. [10]. In order to obtain the pose estimate, we select the best 10% of the particles and compute the weighted average of their poses.

Table 12.1 presents for each of the 27 sessions the corresponding position and heading errors. To generate these values, we run the localization module ten times per session, evaluate the deviation of our estimate from ground truth every 1 m along the ground-truth trajectory, compute the means and RMSEs, and average these metrics over the ten sessions. The results demonstrate that the proposed method achieves both high reliability and accuracy, even if the data used for mapping and for localization lie 15 months apart: The particle filter never even partially diverges, except for one late session discussed below. Furthermore, despite the inaccuracies in ground truth, which affect both the global map and the evaluation, it achieves a mean positioning accuracy over all sessions of 0.284 m. Looking at the evolution of the errors over time, we observe slightly increasing magnitudes. This is due to changes in campus infrastructure accumulating over time and rendering the initial map more and more outdated.

In session 2012-02-23, these changes eventually cause the localization



(a) Registration via the original NCLT ground truth.



(b) Refined registration.

Figure 12.2: The same set of point clouds taken from a short sequence of an NCLT session, registered using different ground-truth robot poses. The colors encode the point height above ground: Blue represents the ground plane, whereas green, yellow, and red indicate increasing height. The upper image shows the result of the registration based on the original NCLT ground truth poses, which we use throughout our experiments. To illustrate the inaccuracy of the original ground truth, the lower image presents a refined registration that we generated via pose-graph optimization. While the original ground truth leads to a blurry point cloud, the refined version significantly improves point alignment and results in crisp details. The mean positional error between both ground truth versions is approximately 0.25 m on average, which leads us to believe that the original NCLT ground truth is off by a similar amount. This fact impedes the generation of an accurate reference pole map and negatively affects our localization results.

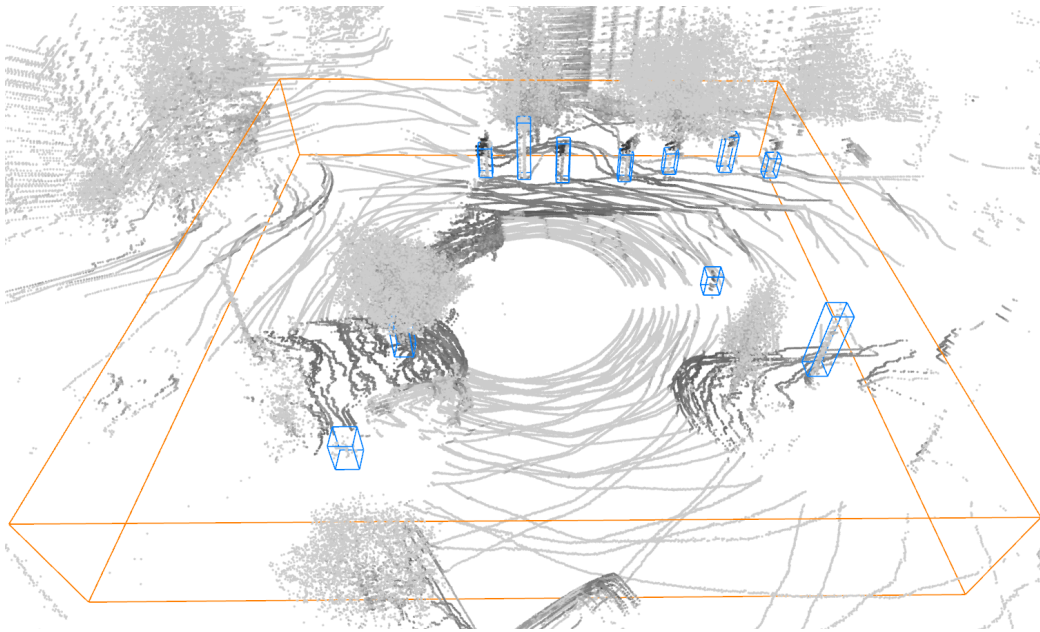


Figure 12.3: Exemplary pole extraction result for a point cloud from the NCLT dataset. The gray values of the points correlate with the intensity values returned by the lidar sensor. The orange wireframe represents the boundaries of the local map, while the blue wireframes represent the extracted poles. The pole extractor is triggered by different kinds of pole-shaped objects like traffic signs, street lamps, and tree trunks.

Session date	f_{map} [%]	Δ_{pos} [m]	RMSE_{pos} [m]	Δ_{ang} [°]	RMSE_{ang} [°]
2012-01-08	100.0	0.130	0.178	0.663	0.857
2012-01-15	8.5	0.156	0.225	0.760	0.999
2012-01-22	5.1	0.172	0.222	0.939	1.291
2012-02-02	0.4	0.155	0.205	0.720	0.975
2012-02-04	0.1	0.144	0.195	0.684	0.903
2012-02-05	0.5	0.148	0.210	0.691	0.947
2012-02-12	0.8	0.269	1.005	0.802	1.040
2012-02-18	0.8	0.149	0.221	0.699	0.938
2012-02-19	0.0	0.148	0.194	0.704	0.944
2012-03-17	0.0	0.149	0.191	0.830	1.062
2012-03-25	0.0	0.200	0.262	1.418	1.836
2012-03-31	0.0	0.143	0.184	0.746	0.973
2012-04-29	0.0	0.170	0.251	0.829	1.079
2012-05-11	5.5	0.161	0.225	0.773	0.998
2012-05-26	0.4	0.158	0.217	0.690	0.889
2012-06-15	0.4	0.180	0.238	0.659	0.879
2012-08-04	0.3	0.210	0.340	0.884	1.143
2012-08-20	3.8	0.189	0.264	0.711	0.941
2012-09-28	0.3	0.206	0.311	0.731	0.952
2012-10-28	1.4	0.217	0.338	0.693	0.919
2012-11-04	2.5	0.257	0.456	0.746	0.996
2012-11-16	2.7	0.403	0.722	1.467	2.031
2012-11-17	0.4	0.243	0.377	0.686	0.959
2012-12-01	0.0	0.266	0.492	0.674	0.930
2013-01-10	0.0	0.217	0.278	0.689	0.911
2013-02-23	0.0	2.470	5.480	1.083	1.769
2013-04-05	0.0	0.365	0.920	0.654	1.028

Table 12.1: Results of our experiments with the NCLT dataset, averaged over ten localization runs per session. The variables Δ_{pos} and Δ_{ang} denote the mean absolute errors in position and heading, respectively, RMSE_{pos} and RMSE_{ang} represent the corresponding root mean squared errors, while f_{map} denotes the fraction of lidar scans per session used to build the reference map.

module to temporarily lose track of the exact robot position. The diverging behavior reproducibly occurs when the robot drives along a row of construction barrels that fence a large construction site. When the global map was built, these barrels were located on the footpath. Just before the session in question, however, the barrels were moved laterally by a few meters, while maintaining their longitudinal positions. Since the barrels are the only landmarks in the corresponding region, the localizer “corrects” the robot position so that the incoming pole measurements match the map. Having passed the construction site, the localizer is confident about its wrong position estimate, which is why it takes some time until the particle cloud diverges and the robot relocalizes. The positioning error over all sessions except 2012-02-23 amounts to 0.200 m.

Lastly, we describe the runtime requirements of our method stochastically. On a 2011 quad-core PC with dedicated GPU, we measure an average 1.33 s for pole extraction with our open-source Python implementation [27], which corresponds to processing 0.5 million laser data points per seconds. The measurement step with data association requires a mean computation time of 0.09 s. These two steps pose by far the highest computational requirements and make others, like the measurement update, negligible.

12.5.2 Localization on the KITTI Dataset

As delineated in section 12.3, Kümmerle et al. [129], Weng et al. [24], and Sefati et al. [25] present methods for vehicle localization with pole landmarks extracted from 3-D lidar data. While the former two use small proprietary datasets – a fact that makes a direct comparison infeasible – Sefati et al. evaluate their method on sequence number 9 of the publicly available KITTI dataset [26]. This sequence is a short recording of 46 s along a simple L-shaped trajectory. Trajectories in KITTI have hardly any overlap, which is why Sefati et al. use the sequence data for both mapping and localization. Consequently, their results have limited significance as to real localization performance: They could theoretically localize the vehicle based on dynamic landmarks only, and they would still obtain accurate results with respect to their map, although it is extremely unlikely that they will encounter the same constellation of dynamic objects ever again. The same is true for Weng et al., who also use a single trajectory of 3.5 km for mapping and localization. Nevertheless, we repeat Sefati et al.’s experiment with the localization system proposed in this paper and compare accuracies in table 12.2. This time, we set the grid spacing for the pole extractor to 0.1 m, because the quality of the ground-truth robot poses is higher than in NCLT. Furthermore, we adjust the parameters of our localizer to match the values Sefati et al. apparently

used – 2000 particles, 3 m initial positioning variation, $\pm 5^\circ$ heading variation – and average our results over 50 experimental runs. As shown in table 12.2, our localization system outperforms the reference method by reducing the RMSEs in position and heading by 54% and 69%, respectively. For qualitative analysis, table 12.2 also includes the results Kümmerle et al. and Weng et al. obtained after processing their respective proprietary datasets.

12.6 Conclusion and Future Work

We presented a complete landmark-based 2-D localization system that relies on poles extracted from 3-D lidar data, that is able to perform long-term localization reliably, and that outperforms current state-of-the-art approaches in terms of accuracy. The implementation is publicly available [27].

For the future, we have two major extensions in mind. First, we plan to fuse the separated mapping and localization modules into a single SLAM module. Second, we would like to explore pole-based localization in different sensor modalities.

12.7 Acknowledgements

We thank Arash Ushani for his kind support with the NCLT dataset.

Approach	Δ_{pos} [m]	RMSE _{pos} [m]	Δ_{lat} [m]	σ_{lat} [m]	Δ_{lon} [m]	σ_{lon} [m]	Δ_{ang} [°]	σ_{ang} [°]	RMSE _{ang} [°]
Kümmerle et al. [129]	0.12	-	0.07	-	0.08	-	0.33	-	-
Weng et al. [24]	-	-	-	0.082	-	0.164	-	0.329	-
Sefati et al. [25]	-	0.24	-	-	-	-	-	-	0.68
Ours	0.096	0.111	0.061	0.075	0.060	0.067	0.133	0.188	0.214

Table 12.2: Comparison of the accuracies of Sefati et al.’s method and the proposed localization approach on the KITTI dataset. The results of Weng et al. and Kümmerle et al. are not directly comparable and are stated for qualitative analysis only.

Part IV

Conclusion and Outlook

This dissertation explored different aspects of mapping and localization for mobile robots using lidar data: sensor models, map representations, feature extraction, and feature-based mapping and localization. In the following, we summarize the key insights gained in the scope of these works.

The first important finding in all our works is as simple as that: Leveraging all tangible information always improves performance compared to using only part of the information at hand. The decay-rate model, for example, outperforms related approaches like the reflection model in terms of localization accuracy because it does not only account for whether or not a ray hits a particular voxel, but also what distance the ray covers within that voxel. Full-posterior maps outperform maximum-likelihood maps because for each map cell, they maintain full probability distributions over all possible map values instead of condensing them to their modes. Our polyline and plane extraction approaches outperform related methods in terms of accuracy because in contrast to the latter, they account for the ray path information contained in the sensor data.

The second key insight refers to the theoretical basis of an algorithm: the stronger its probabilistic foundation, the higher the accuracy of the corresponding results. Our works on polyline and plane feature extraction demonstrate this finding when being compared to related heuristic methods. Moreover, it is because of this observation that we are convinced our approach to pole extraction could be further refined by basing it on a more probability-theoretic formulation.

In order to apply these insights to future research in lidar-based mapping and localization, we derive a set of guiding principles from them. First, use as little algorithmic simplifications as possible. As can be seen in the comparison of full-posterior maps versus maximum-likelihood maps, avoiding simplifications can lead to significant improvements in accuracy.

Second, always make use of the full ray path information. Although popular, correlation-based lidar models, which only take lidar endpoints as input, ignore a significant portion of the measurement information output by the sensor. Without knowing the location of the sensor, they cannot process any information about the free space. This clearly leads to poor performance compared to methods that leverage all of this information.

Third, do not discard no-return measurements. Correlation-based models may not be able to process them, but they carry valuable information about where the space around the robot is free.

The above recommendations target maximum accuracy, not computational efficiency. When these two goals conflict with each other, it may not be possible to fully comply with our advice due to computational requirements. But often, they do not conflict, as demonstrated in our works intro-

ducing decay-rate maps and closed-form full-posterior maps: Both decay-rate maps and full-posterior maps achieve higher accuracy than previous methods without increasing computational complexity at all.

When analyzing the state of the art, one realizes that these recommendations are often violated. For example, there are only few approaches to grid mapping that do without the independence assumption between the cells of the map, although this assumption clearly discards a possibly significant portion of the original information provided by the sensor. Similarly, most approaches to robot localization make use of the Markov assumption, which also simplifies the information at hand. For us, gradually overcoming these simplifications has great potential to improve system performance, to push the boundary of the state of the art in robot mapping and localization further, and eventually to render robots capable of navigating even the most challenging environments.

Bibliography

- [1] M. G. Jadidi, J. V. Miro, and G. Dissanayake, “Gaussian processes autonomous mapping and exploration for range-sensing mobile robots,” *Autonomous Robots*, vol. 42, no. 2, pp. 273–290, 2018.
- [2] A. Schaefer, L. Luft, and W. Burgard, “DCT maps: compact differentiable lidar maps based on the cosine transform,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1002–1009, April 2018.
- [3] M. Veeck and W. Burgard, “Learning polyline maps from range scan data acquired with mobile robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, September 2004, pp. 1065–1070.
- [4] A. Schaefer, D. Büscher, L. Luft, and W. Burgard, “A maximum likelihood approach to extract polylines from 2-D laser range scans,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2018, pp. 4766–4773.
- [5] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, “University of Michigan north campus long-term vision and lidar dataset,” *International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2015.
- [6] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard, “Long-term urban vehicle localization using pole landmarks extracted from 3-D lidar scans,” in *European Conference on Mobile Robots*, September 2019.
- [7] United States of America Patent 6 809 490, 2001.
- [8] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conratt, K. Daniilidis *et al.*, “Event-based vision: a survey,” *arXiv preprint arXiv:1904.08405*, 2019.

-
- [9] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun, “Sonar-based mapping of large-scale mobile robot environments using EM,” in *Sixteenth International Conference on Machine Learning*, 1999, pp. 67–76.
- [10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
- [11] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, “Map building with mobile robots in dynamic environments,” in *IEEE International Conference on Robotics and Automation*, vol. 2, September 2003, pp. 1557–1563.
- [12] H. Moravec, “Sensor fusion in certainty grids for mobile robots,” *AI Magazine*, vol. 9, no. 2, pp. 61–74, July 1988.
- [13] A. Elfes, “Occupancy grids: a probabilistic framework for robot perception and navigation,” Ph.D. dissertation, Carnegie Mellon University, 1989.
- [14] P. Biber and W. Strasser, “The normal distributions transform: a new approach to laser scan matching,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, October 2003, pp. 2743–2748.
- [15] S. T. O’Callaghan and F. T. Ramos, “Gaussian process occupancy maps,” *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.
- [16] F. T. Ramos and L. Ott, “Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent,” in *Robotics: Science and Systems*, July 2015.
- [17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [18] A. Atoyán and J. Patera, “Properties of continuous Fourier extension of the discrete cosine transform and its multidimensional generalization,” *Journal of Mathematical Physics*, vol. 45, no. 6, pp. 2468–2491, 2004.
- [19] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, “A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005, pp. 1929–1934.

- [20] M. Visvalingam and J. D. Whyatt, "Line generalisation by repeated elimination of points," *The Cartographic Journal*, vol. 30, no. 1, pp. 46–51, 1993.
- [21] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112–122, October 1973.
- [22] T. Pavlidis and S. L. Horowitz, "Segmentation of plane curves," *IEEE Transactions on Computers*, vol. C-23, no. 8, pp. 860–870, August 1974.
- [23] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. 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*, vol. 18, no. 7, pp. 673–689, July 1996.
- [24] L. Weng, M. Yang, L. Guo, B. Wang, and C. Wang, "Pole-based real-time localization for autonomous driving in congested urban scenarios," in *IEEE International Conference on Real-time Computing and Robotics*, August 2018, pp. 96–101.
- [25] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther, and A. Kampker, "Improving vehicle localization using semantic and pole-like landmarks," in *IEEE Intelligent Vehicles Symposium*, June 2017, pp. 13–19.
- [26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [27] A. Schaefer and D. Büscher. Long-term urban vehicle localization using pole landmarks extracted from 3-D lidar scans. [Online]. Available: <https://github.com/acschaefer/polex>
- [28] L. Luft, A. Schaefer, T. Schubert, and W. Burgard, "Detecting changes in the environment based on full posterior distributions over real-valued grid maps," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1299–1305, April 2018.
- [29] A. Schaefer*, L. Luft*, and W. Burgard, "An analytical lidar sensor model based on ray path information," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1405–1412, July 2017.

-
- [30] S. Thrun, “Learning occupancy grid maps with forward sensor models,” *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, September 2003.
- [31] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas, “Semantic localization via the matrix permanent,” in *Robotics: Science and Systems*, vol. 2, July 2014.
- [32] K. Y. K. Leung, F. Inostroza, and M. Adams, “Generalizing random-vector SLAM with random finite sets,” in *IEEE International Conference on Robotics and Automation*, May 2015, pp. 4583–4588.
- [33] M. Yguel, O. Aycard, and C. Laugier, “Efficient GPU-based construction of occupancy grids using several laser range-finders,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, May 2006, pp. 105–110.
- [34] A. Elfes, “Using occupancy grids for mobile robot perception and navigation,” *Computer*, vol. 22, no. 6, pp. 46–57, June 1989.
- [35] S. Thrun, “A probabilistic on-line mapping algorithm for teams of mobile robots,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.
- [36] F. Ferri, M. Gianni, M. Menna, and F. Pirri, “Dynamic obstacles detection and 3D map updating,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2015, pp. 5694–5699.
- [37] M. Bennewitz, C. Stachniss, S. Behnke, and W. Burgard, “Utilizing reflection properties of surfaces to improve mobile robot localization,” in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 63–68.
- [38] J. Ahtiainen, T. Stoyanov, and J. Saarinen, “Normal distributions transform traversability maps: LIDAR-only approach for traversability mapping in outdoor environments,” *Journal of Field Robotics*, vol. 34, no. 3, pp. 600–621, 2017.
- [39] R. Limosani, L. Y. Morales, J. Even, F. Ferreri, A. Watanabe, F. Cavallo, P. Dario, and N. Hagita, “Long-term human affordance maps,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2015, pp. 5748–5754.
- [40] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard, “Gaussian beam processes: a nonparametric Bayesian measurement model for range finders,” in *Robotics: Science and Systems*, June 2007.

-
- [41] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, February 1992.
- [42] T. De Laet, J. De Schutter, and H. Bruyninckx, "Rigorously Bayesian range finder sensor model for dynamic environments," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 2994–3001.
- [43] J. Mullane, M. D. Adams, and W. S. Wijesoma, "Robotic mapping using measurement likelihood filtering," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 172–190, 2009.
- [44] R. P. Mahler, *Statistical multisource-multitarget information fusion*. Artech House Norwood, MA, USA, 2007, vol. 685.
- [45] L. Luft*, A. Schaefer*, T. Schubert, and W. Burgard, "Closed-form full map posteriors for robot localization with lidar sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2017, pp. 6678–6684.
- [46] T. K. Marks, A. Howard, M. Bajracharya, G. W. Cottrell, and L. Matthies, "Gamma-SLAM: using stereo vision and variance grid maps for SLAM in unstructured environments," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 3717–3724.
- [47] A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 176–183.
- [48] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," in *The International Journal of Robotics Research*, vol. 32, no. 14, December 2013, pp. 1662–1678.
- [49] K. P. Murphy, "Bayesian map learning in dynamic environments," in *Advances in Neural Information Processing Systems*. MIT Press, 2000, pp. 1015–1021.
- [50] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos, "The SPmap: a probabilistic framework for simultaneous localization and map building," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 948–952, October 1999.

- [51] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, “Recursive decentralized collaborative localization for sparsely communicating robots,” in *Robotics: Science and Systems*, June 2016.
- [52] S. Thrun, “Robotic mapping: a survey,” in *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann, 2002.
- [53] D. Valcarce, J. Parapar, and A. Barreiro, “Additive smoothing for relevance-based language modelling of recommender systems,” in *4th Spanish Conference on Information Retrieval*, New York, NY, USA, 2016, pp. 9:1–9:8.
- [54] A. Howard and N. Roy, “The robotics data set repository (Radish),” 2003. [Online]. Available: <http://radish.sourceforge.net/>
- [55] J. Ryde and M. Brünig, “Non-cubic occupied voxel lists for robot maps,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2009, pp. 4771–4776.
- [56] H. Moravec and A. Elfes, “High resolution maps from wide angle sonar,” in *IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 116–121.
- [57] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, “Terrain mapping for a roving planetary explorer,” in *IEEE International Conference on Robotics and Automation*, vol. 2, May 1989, pp. 997–1002.
- [58] R. Triebel, P. Pfaff, and W. Burgard, “Multi-level surface maps for outdoor terrain mapping and loop closing,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006, pp. 2276–2282.
- [59] I. Dryanovski, W. Morris, and J. Xiao, “Multi-volume occupancy grids: an efficient probabilistic 3D mapping model for micro aerial vehicles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2010, pp. 1553–1559.
- [60] D. Meagher, “Geometric modeling using octree encoding,” *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982.
- [61] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: a probabilistic, flexible, and compact 3D map representation for robotic systems,” in *ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

- [62] P. Payeur, P. Hebert, D. Laurendeau, and C. M. Gosselin, “Probabilistic octree modeling of a 3D dynamic environment,” in *IEEE International Conference on Robotics and Automation*, vol. 2, April 1997, pp. 1289–1296.
- [63] K. Pathak, A. Birk, J. Poppinga, and S. Schwertfeger, “3D forward sensor modeling and application to occupancy grid based sensor fusion,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2007, pp. 2059–2064.
- [64] N. Fairfield, G. Kantor, and D. Wettergreen, “Real-time SLAM with octree evidence grids for exploration in underwater tunnels,” *Journal of Field Robotics*, vol. 24, no. 1-2, pp. 3–21, 2007.
- [65] T. Krajník, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, “Spectral analysis for long-term robotic mapping,” in *IEEE International Conference on Robotics and Automation*, May 2014, pp. 3706–3711.
- [66] J. Ryde and H. Hu, “3D mapping with multi-resolution occupied voxel lists,” *Autonomous Robots*, vol. 28, no. 2, p. 169, September 2009.
- [67] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal, “Normal distributions transform occupancy maps: application to large-scale online 3D mapping,” in *IEEE International Conference on Robotics and Automation*, May 2013, pp. 2233–2238.
- [68] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, “Normal distributions transform Monte-Carlo localization (NDT-MCL),” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2013, pp. 382–389.
- [69] T. Stoyanov, J. Saarinen, H. Andreasson, and A. J. Lilienthal, “Normal distributions transform occupancy map fusion: simultaneous mapping and tracking in large scale dynamic environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2013, pp. 4702–4708.
- [70] R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal, “Localization in highly dynamic environments using dual-timescale NDT-MCL,” in *IEEE International Conference on Robotics and Automation*, May 2014, pp. 3956–3962.

- [71] M. Yguel, C. T. M. Keat, C. Brailon, C. Laugier, and O. Aycard, “Dense mapping for range sensors: efficient algorithms and sparse representations,” in *Robotics: Science and Systems*, June 2007.
- [72] D. Fridovich-Keil, E. Nelson, and A. Zakhor, “AtomMap: a probabilistic amorphous 3D map representation for robotics and surface reconstruction,” in *IEEE International Conference on Robotics and Automation*, May 2017, pp. 3110–3117.
- [73] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, “6D SLAM – 3D mapping outdoor environments,” *Journal of Field Robotics*, vol. 24, no. 8–9, pp. 699–722, 2007.
- [74] 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,” *Robotics and Autonomous Systems*, vol. 45, no. 3, pp. 181–198, 2003.
- [75] D. M. Cole and P. M. Newman, “Using laser range data for 3D SLAM in outdoor environments,” in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 1556–1563.
- [76] R. Chatila and J. Laumond, “Position referencing and consistent world modeling for mobile robots,” in *IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 138–145.
- [77] M. Montemerlo, D. Hähnel, D. Ferguson, R. Triebel, W. Burgard, S. Thayer, W. Whittaker, and S. Thrun, “A system for three-dimensional robotic mapping of underground mines,” Carnegie-Mellon University Pittsburgh, PA, School of Computer Science, Tech. Rep., 2002.
- [78] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, “The Digital Michelangelo Project: 3D scanning of large statues,” in *27th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2000, pp. 131–144.
- [79] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinect-Fusion: real-time dense surface mapping and tracking,” in *10th IEEE International Symposium on Mixed and Augmented Reality*, October 2011, pp. 127–136.

-
- [80] S. O’Callaghan, F. T. Ramos, and H. Durrant-Whyte, “Contextual occupancy maps using Gaussian processes,” in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 1054–1060.
- [81] S. Dragiev, M. Toussaint, and M. Gienger, “Gaussian process implicit surfaces for shape estimation and grasping,” in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2845–2850.
- [82] F. T. Ramos and L. Ott, “Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent,” *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.
- [83] C. E. Rasmussen and H. Nickisch, “Gaussian processes for machine learning (GPML) toolbox,” *Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, December 2010.
- [84] A. Garulli, A. Giannitrapani, A. Rossi, and A. Vicino, “Mobile robot SLAM for line-based environment representation,” in *44th IEEE Conference on Decision and Control*, December 2005, pp. 2041–2046.
- [85] D. Rodriguez-Losada, F. Matia, and R. Galan, “Building geometric feature based maps for indoor service robots,” *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 546–558, 2006.
- [86] Y.-H. Choi, T.-K. Lee, and S.-Y. Oh, “A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot,” *Autonomous Robots*, vol. 24, no. 1, pp. 13–27, January 2008.
- [87] J. Lv, Y. Kobayashi, A. A. Ravankar, and T. Emaru, “Straight line segments extraction and EKF-SLAM in indoor environment,” *Journal of Automation and Control Engineering*, vol. 2, no. 3, 2014.
- [88] C. Berger, “Toward rich geometric map for SLAM: online detection of planes in 2D lidar,” *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 7, 2013.
- [89] D. Sack and W. Burgard, “A comparison of methods for line extraction from range data,” in *5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, vol. 33, 2004.
- [90] G. A. Borges and M. J. Aldon, “A split-and-merge segmentation algorithm for line extraction in 2D range images,” in *15th International Conference on Pattern Recognition*, vol. 1, 2000, pp. 441–444.

-
- [91] L. J. Latecki and R. Lakaemper, "Polygonal approximation of laser range data based on perceptual grouping and EM," in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 790–796.
- [92] S. T. Pfister and J. W. Burdick, "Multi-scale point and line range data algorithms for mapping and localization," in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 1159–1166.
- [93] J. S. Berrio, S. O. Ordoñez, and E. C. Bravo, "Lines extraction in laser scans through the integration of the Hough transform and SEF," in *Workshop on Engineering Applications*, May 2012, pp. 1–6.
- [94] A. Harati and R. Siegwart, "A new approach to segmentation of 2D range scans into linear regions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2007, pp. 2083–2088.
- [95] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *IEEE International Conference on Robotics and Automation*, vol. 1, September 2003, pp. 1304–1311.
- [96] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.
- [97] D. Navarro, G. Benet, and M. Martinez, "Line based robot localization using a rotary sonar," in *IEEE Conference on Emerging Technologies and Factory Automation*, September 2007, pp. 896–899.
- [98] D. Navarro, G. Benet, and F. Blanes, "Line-based incremental map building using infrared sensor ring," in *IEEE International Conference on Emerging Technologies and Factory Automation*, September 2008, pp. 833–838.
- [99] G. D. Tipaldi and K. O. Arras, "FLIRT – interest regions for 2D range data," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 3616–3622.
- [100] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

- [101] —, “Keypoint design and evaluation for place recognition in 2D lidar maps,” *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211–1224, 2009.
- [102] J. Lagarias, J. A. Reeds, M. H. Wright, and P. Wright, “Convergence properties of the Nelder–Mead Simplex Method in low dimensions,” *SIAM Journal on Optimization*, vol. 9, pp. 112–147, 12 1998.
- [103] A. Schaefer, J. Vertens, D. Büscher, and W. Burgard, “A maximum likelihood approach to extract finite planes from 3-D laser scans,” in *IEEE International Conference on Robotics and Automation*, May 2019, pp. 72–78.
- [104] A. Schaefer, J. Vertens, and D. Büscher. (2019) Probabilistic Plane Extraction. [Online]. Available: <https://github.com/acschaefer/ppe>
- [105] J.-E. Deschaud and F. Goulette, “A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing,” in *5th International Symposium on 3D Data Processing, Visualization and Transmission*, 2010.
- [106] A. Nurunnabi, D. Belton, and G. West, “Robust segmentation in laser scanning 3D point cloud data,” in *International Conference on Digital Image Computing Techniques and Applications*, December 2012, pp. 1–8.
- [107] Z. Dong, B. Yang, P. Hu, and S. Scherer, “An efficient global energy optimization approach for robust 3D plane segmentation of point clouds,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 137, pp. 112–133, 2018.
- [108] D. Holz and S. Behnke, “Fast range image segmentation and smoothing using approximate surface reconstruction and region growing,” in *Intelligent Autonomous Systems 12*. Springer, 2013, pp. 61–73.
- [109] P. F. Proença and Y. Gao, “Fast cylinder and plane extraction from depth cameras for visual odometry,” *Computing Research Repository*, vol. abs/1803.02380, 2018.
- [110] X. Jiang and H. Bunke, “Fast segmentation of range images into planar regions by scan line grouping,” *Machine Vision and Applications*, vol. 7, no. 2, pp. 115–122, June 1994.

-
- [111] C. Cabo, S. G. Cortés, and C. Ordoñez, “Mobile laser scanner data for automatic surface detection based on line arrangement,” *Automation in Construction*, vol. 58, pp. 28–37, 2015.
- [112] A. J. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, “Efficient organized point cloud segmentation with connected components,” *Semantic Perception Mapping and Exploration*, 2013.
- [113] C. Feng, Y. Taguchi, and V. R. Kamat, “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering,” in *IEEE International Conference on Robotics and Automation*, May 2014, pp. 6218–6225.
- [114] R. T. Marriott, A. Pashevich, and R. Horaud, “Plane-extraction from depth-data using a Gaussian mixture regression model,” *Computing Research Repository*, vol. abs/1710.01925, 2017.
- [115] T. T. Pham, M. Eich, I. Reid, and G. Wyeth, “Geometrically consistent plane extraction for dense indoor 3D maps segmentation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2016, pp. 4199–4204.
- [116] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.
- [117] P. F. U. Gotardo, O. R. P. Bellon, and L. Silva, “Range image segmentation by surface extraction using an improved robust estimator,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2003, pp. II–33.
- [118] O. Gallo, R. Manduchi, and A. Rafii, “CC-RANSAC: fitting planes in the presence of multiple surfaces in range data,” *Pattern Recognition Letters*, vol. 32, no. 3, pp. 403–410, 2011.
- [119] A. Sveier, A. L. Kleppe, L. Tingelstad, and O. Egeland, “Object detection in point clouds using conformal geometric algebra,” *Advances in Applied Clifford Algebras*, vol. 27, no. 3, pp. 1961–1976, September 2017.
- [120] M. Alehdaghi, M. A. Esfahani, and A. Harati, “Parallel RANSAC: speeding up plane extraction in RGBD image sequences using GPU,”

- in *International Conference on Computer and Knowledge Engineering*, October 2015, pp. 295–300.
- [121] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani, “Recognising structure in laser scanning point clouds,” in *Proceedings of the IS-PRS working group VIII/2: laser scanning for forest and landscape assessment*, M. Thies, B. Koch, H. Spiecker, and H. Weinacher, Eds. University of Freiburg, October 2004, pp. 33–38.
- [122] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke, “Efficient multi-resolution plane segmentation of 3D point clouds,” in *International Conference on Intelligent Robotics and Applications*. Springer, 2011, pp. 145–156.
- [123] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, “The 3D Hough transform for plane detection in point clouds: a review and a new accumulator design,” *3D Research*, vol. 2, no. 2, p. 3, November 2011.
- [124] H. Akaike, “Information theory and an extension of the maximum likelihood principle,” in *Selected papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.
- [125] P. H. Torr and A. Zisserman, “MLESAC: a new robust estimator with application to estimating image geometry,” *Computer vision and image understanding*, vol. 78, no. 1, pp. 138–156, 2000.
- [126] Mitsubishi Electric Research Laboratories. (2018) PEAC. [Online]. Available: <http://www.merl.com/research/?research=license-request&sw=PEAC>
- [127] M. Modsching, R. Kramer, and K. ten Hagen, “Field trial on GPS accuracy in a medium size city: the influence of built-up,” in *3rd Workshop on Positioning, Navigation and Communication*, 2006, pp. 209–218.
- [128] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps,” in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 4372–4378.
- [129] J. Kümmerle, M. Sons, F. Poggenhans, T. Kühner, M. Lauer, and C. Stiller, “Accurate and efficient self-localization on roads using basic geometric primitives,” in *IEEE International Conference on Robotics and Automation*, May 2019, pp. 5965–5971.

- [130] R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios," in *IEEE International Conference on Intelligent Robots and Systems*, October 2016, pp. 2161–2166.
- [131] C. Brenner, "Global localization of vehicles using local pole patterns," in *Pattern Recognition*, J. Denzler, G. Notni, and H. Süße, Eds. Springer Berlin Heidelberg, 2009, pp. 61–70.
- [132] C. Cabo, C. Ordóñez, S. Garcia-Cortes, and J. Martínez-Sánchez, "An algorithm for automatic detection of pole-like street furniture objects from mobile laser scanner point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 47–56, 01 2014.
- [133] F. Tombari, N. Fioraio, T. Cavallari, S. Salti, A. Petrelli, and L. D. Stefano, "Automatic detection of pole-like structures in 3D urban environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2014, pp. 4922–4929.
- [134] B. Rodríguez-Cuenca, S. García-Cortés, C. Ordóñez, and M. C. Alonso, "Automatic detection and classification of pole-like objects in urban point cloud data using an anomaly detection algorithm," *Remote Sensing*, vol. 7, no. 10, pp. 12 680–12 703, 2015.
- [135] Y. Yu, J. Li, H. Guan, C. Wang, and J. Yu, "Semiautomated extraction of street light poles from mobile LiDAR point-clouds," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 3, pp. 1374–1386, 2015.
- [136] F. Wu, C. Wen, Y. Guo, J. Wang, Y. Yu, C. Wang, and J. Li, "Rapid localization and extraction of street light poles in mobile lidar point clouds: a supervoxel-based approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 292–305, February 2017.
- [137] H. Zheng, F. Tan, and R. Wang, "Pole-like object extraction from mobile lidar data," *The International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 41, 2016.
- [138] H. Yokoyama, H. Date, S. Kanai, and H. Takeda, "Detection and classification of pole-like objects from mobile laser scanning data of urban environments," *International Journal of CAD/CAM*, vol. 13, no. 2, pp. 31–40, 2013.

-
- [139] C. Ordóñez, C. Cabo, and E. Sanz-Ablanedo, “Automatic detection and classification of pole-like objects for urban cartography using mobile laser scanning data,” *Sensors*, vol. 17, no. 7, p. 1465, 2017.
- [140] F. Li, S. Oude Elberink, and G. Vosselman, “Pole-like road furniture detection and decomposition in mobile laser scanning data based on spatial relations,” *Remote Sensing*, vol. 10, no. 4, 2018.
- [141] B. Qin, Z. Chong, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, “Curb-intersection feature based Monte Carlo localization on urban roads,” in *IEEE International Conference on Robotics and Automation*, May 2012, pp. 2640–2646.
- [142] A. Schindler, “Vehicle self-localization with high-precision digital maps,” in *IEEE Intelligent Vehicles Symposium*, June 2013, pp. 141–146.
- [143] M. Schreiber, C. Knöppel, and U. Franke, “LaneLoc: lane marking based localization using highly accurate maps,” in *IEEE Intelligent Vehicles Symposium*, June 2013, pp. 449–454.
- [144] A. Y. Hata and D. F. Wolf, “Feature detection for vehicle localization in urban environments using a multilayer lidar,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 420–429, February 2016.
- [145] A. Welzel, P. Reisdorf, and G. Wanielik, “Improving urban vehicle localization with traffic sign recognition,” in *IEEE International Conference on Intelligent Transportation Systems*, September 2015, pp. 2728–2732.
- [146] J.-H. Im, S.-H. Im, and G.-I. Jee, “Vertical corner feature based precise vehicle localization using 3D lidar in urban area,” *Sensors*, vol. 16, no. 8, p. 1268, 2016.
- [147] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, January 1975.

