EFFICIENT 3D SEGMENTATION, REGISTRATION AND MAPPING FOR MOBILE ROBOTS

DISSERTATION

zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

DIRK HOLZ
aus Bergisch Gladbach

Bonn, March 2016

Angefertigt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Erster Gutachter	Prof. Dr. Sven Behnke
Zweiter Gutachter	Prof. Dr. Joachim Hertzberg
Tag der Promotion	05.05.2017
Erscheinungsjahr	2017

The family—that dear octopus from whose tentacles we never quite escape, nor, in our inmost hearts, ever quite wish to.
— Dodie Smith

Family is not an important thing. It's everything.
— Michael J. Fox

Dedicated to my wife Suzana and my parents Helga and Jürgen.
Without their support I could not have completed this process.

Für Suzana, Karl und Leonard

ABSTRACT

Sometimes simple is better! For certain situations and tasks, simple but robust methods can achieve the same or better results in the same or less time than related sophisticated approaches. In the context of robots operating in real-world environments, key challenges are perceiving objects of interest and obstacles as well as building maps of the environment and localizing therein. The goal of this thesis is to carefully analyze such problem formulations, to deduce valid assumptions and simplifications, and to develop simple solutions that are both robust and fast. All approaches make use of sensors capturing three-dimensional (3D) information, such as consumer color and depth (RGB-D) cameras. Comparative evaluations show the performance of the developed approaches.

For identifying objects and regions of interest in manipulation tasks, a real-time object segmentation pipeline is proposed. It exploits several common assumptions of manipulation tasks such as objects being on horizontal support surfaces (and well separated). It achieves real-time performance by using particularly efficient approximations in the individual processing steps, subsampling the input data where possible, and processing only relevant subsets of the data. The resulting pipeline segments 3D input data with up to 30 Hz.

In order to obtain complete segmentations of the 3D input data, a second pipeline is proposed that approximates the sampled surface, smooths the underlying data, and segments the smoothed surface into coherent regions belonging to the same geometric primitive. It uses different primitive models and can reliably segment input data into planes, cylinders and spheres. A thorough comparative evaluation shows state-of-the-art performance while computing such segmentations in near real-time.

The second part of the thesis addresses the registration of 3D input data, i.e., consistently aligning input captured from different view poses. Several methods are presented for different types of input data. For the particular application of mapping with micro aerial vehicles (MAVs) where the 3D input data is particularly sparse, a pipeline is proposed that uses the same approximate surface reconstruction to exploit the measurement topology and a surface-to-surface registration algorithm that robustly aligns the data. Optimization of the resulting graph of determined view poses then yields globally consistent 3D maps. For sequences of RGB-D data this pipeline is extended to include additional subsampling steps and an initial alignment of the data in local windows in the pose graph. In both cases, comparative evaluations show a robust and fast alignment of the input data.

ZUSAMMENFASSUNG

In den letzten Jahren und Jahrzehnten vollzog die Robotikforschung einen Wandel von der vorherrschenden Forschung an vorprogrammierten Robotern hinter Sicherheitszäunen hin zu autonomen intelligenten Systemen, die selbstständig in Alltagsumgebungen handeln können. Diese neuen Anwendungen öffnen zwar neue Märkte, bringen aber auch neue Fragen- und Problemstellungen mit sich. Von besonderer Bedeutung sind hierbei Fähigkeiten zur Wahrnehmung der Umgebung des Roboters wie zum Beispiel die Wahrnehmung der zu manipulierenden Objekte, der Hindernisse in der unmittelbaren Umgebung, oder von menschlichen Benutzern und Mitarbeitern. Für die meisten dieser Problemstellungen wurden hochentwickelte Lösungsansätze vorgestellt, die sich vor allem in ihrer Robustheit und ihrer Komplexität unterscheiden. Ein Schlüsselkriterium für den Erfolg eines Verfahrens ist jedoch nicht nur eine hohe Erfolgsrate ohne das Auftreten von Fehlern, sondern auch eine kurze Laufzeit, um Unterbrechungen im Arbeitsfluss des Roboters zu vermeiden. Für viele Situationen und Aufgaben können einfache aber robuste Verfahren effizienter als hochentwickelte Verfahren sein, indem sie in derselben oder weniger Zeit dieselben oder bessere Ergebnisse liefern. Ziel dieser Arbeit ist es, die erwähnten Problemstellungen genau zu untersuchen, Annahmen und Vereinfachungen abzuleiten, und solche simplen aber effizienten Lösungsansätze zu entwickeln. Die Effizienz der entwickelten Verfahren wird dabei in umfangreichen Vergleichen mit dem aktuellen Stand der Technik verglichen. In allen Verfahren werden 3D Sensoren verwendet, die ihre Umgebung räumlich abtasten. Der Fokus liegt dabei auf sogenannten RGB-D Kameras, die bei geringen Anschaffungskosten und geringer Größe, Farb- und Tiefenbilder mit hoher Bildrate aufnehmen. Vor allem die Topologie der Messungen wird dabei ausgenutzt, um geringe Laufzeiten zu erreichen.

Die erste Problemstellung in dieser Reihe ist die Wahrnehmung von Objekten in Manipulationsaufgaben. Um manipulierbare Objekte zu finden und weitere Verarbeitungsschritte auf relevante Regionen in den Daten fokussieren zu können, wird eine Echtzeit-Verarbeitungspipeline vorgestellt. Die Pipeline nutzt verschiedene Annahmen aus, wie die dass Objekte meist auf horizontalen Flächen stehen und einen gewissen Abstand voneinander haben. Zusätzlich wird Geschwindigkeit dadurch erzielt, dass in den einzelnen Schritten nur Untermengen der Eingabedaten oder nur Daten in relevanten Regionen verarbeitet werden. Ferner werden in den einzelnen Verarbeitungsschritten nur besonders schnelle Verfahren, wie zum Beispiel für die Berechnung von Oberflächennormalen, verwendet. Die resul-

tierende Pipeline segmentiert Eingabedaten in horizontale Flächen und potentielle Objekte darauf in Echtzeit (30 Hz).

Um 3D Eingabedaten vollständig in Ebenen und andere geometrische Primitive zu segmentieren, wird eine zweite Pipeline vorgestellt. Diese berechnet zunächst eine approximierte Oberflächenrekonstruktion, glättet die Daten, und segmentiert dann die geglättete Oberfläche in kohärente Regionen, die zu einem Primitiv gehören. Die Pipeline kann mehrere Primitive wie Ebenen, Zylinder und Kugeln unterscheiden. Eine ausführliche Vergleichsevaluation zeigt Segmentierungen vergleichbar mit dem Stand der Technik und Laufzeiten im Bereich der Echtzeitverarbeitung.

Der zweite Teil der Arbeit beschäftigt sich mit der Registrierung von 3D Eingabedaten, die von unterschiedlichen Beobachtuntspunkten und -orientierungen aufgenommen wurden, also dem korrekten Ausrichten der Daten zueinander. Für verschiedene Typen von Eingabedaten werden entsprechende Verfahren vorgestellt. Speziell für die Kartierung der Umgebung mit Leichtfluggeräten werden ein spezieller 3D Laser Scanner und ein Verfahren zur Registrierung der besonders spärlichen Daten des Scanners vorgestellt. Die Pipeline berechnet zunächst wieder approximierte Oberflächenrekonstruktionen für die aufgenommenen Laser Scans und registriert diese dann, indem die Oberflächen zueinander ausgerichtet werden. Optimierung der resultieren Graphen aus bestimmten Beobachtungsposen ermöglicht dann die Konstruktion einer konsistenten dreidimensionalen Karte der Umgebung. Für Sequenzen von RGB-D Daten, wird die Pipeline schließlich um weiteres Subsampling und initiale Registrierungen in lokalen Fenstern benachbarter Beobachtungsposen erweitert. In beiden Fällen (spärliche 3D Laser Scans und RGB-D Sequezen) zeigen Vergleichsevaluationen robuste und schnelle Registrierung der Eingabedaten.

ACKNOWLEDGMENTS

First and foremost, I would like to give special thanks to Prof. Dr. Sven Behnke whose advice, insightful criticisms, and patient encouragement aided the writing of this thesis in innumerable ways. Furthermore, I would like to thank David Droeschel, Matthias Nieuwenhuisen, Jörg Stückler, Michael Schreiber, Angeliki Topalidou-Kyniazopoulou, Max Schwarz and all the others in the Autonomous Intelligent Systems group for the time and the projects we shared in Bonn.

I would like to express my gratitude to Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich and Markus Vincze for providing the object segmentation database (OSD). Special thanks also to Ruwen Schnabel, Stefan Holzer and Radu B. Rusu for the collaboration on shape detection and normal estimation in the context of the fast object detection pipeline and for many fruitful discussions.

I thank Adam Hoover, Gillian Jean-Baptiste, Xiaoyi Jiang, Patrick J. Flynn, Horst Bunke, Dimitry B. Goldgof, Kevin Bowyer, David W. Eggert, Andrew Fitzgibbon, and Robert B. Fisher for providing the SegComp plane segmentation datasets and the corresponding evaluation framework. Furthermore, my thanks go to Bastian Oehler, Jörg Stückler and Sven Behnke for providing the Microsoft Kinect datasets and the modified SegComp evaluation tool.

I would also like to express my gratitude to Kristiyan Georgiev, Ross T. Creed, and Rolf Lakaemper as well as Alex J. B. Trevor, Suat Gedikli, Radu B. Rusu, and Henrik I. Christensen for providing the implementations of their plane segmentation algorithms.

I thank Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers for providing the RGBD SLAM datasets and benchmarking tools. I would also like to express my gratitude to Christian Kerl, Jürgen Sturm, and Daniel Cremers for providing the implementation of their RGB-D mapping approach DVO-SLAM.

The work presented in this thesis has been partially funded by the European Union's Seventh Framework Programme in the FP7 ICT-2007.2.2 project ECHORD (grant agreement 231 143) experiment ActReMa, the FP7 ICT-2013.2.2 project STAMINA (grant agreement 610 917), and the FP7 FoF-ICT-2013.7.1 project EuRoC (grant agreemant 608 849), as well as by the German Research Foundation (DFG) under grant BE 2556/7-1 (Mapping on Demand).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

2D	Two-dimensional
3D	Three-dimensional
2.5D	Two-and-a-half-dimensional, a two-dimensional (2D) surface containing information about the 3rd dimension, e.g., depth images
APS	Approximate Plane Segmentation
ATE	Absolute trajectory error (Sturm *et al.*, 2012)
BFGS	Broyden-Fletcher-Goldfarb-Shanno, algorithm
BRIEF	Binary robust independent elementary features (Calonder *et al.*, 2010)
CRF	Conditional random field
DoF	Degree-of-freedom
FAST	FAST corner detector (Rosten and Drummond, 2006; Rosten *et al.*, 2010)
FOVIS	Visual odometry library (Huang *et al.*, 2011)
FPFH	Fast point feature histogram (Rusu *et al.*, 2009a)
g2o	Generalized Graph Optimization (Kümmerle *et al.*, 2011)
GICP	Generalized-ICP (Segal *et al.*, 2009)
GPU	Graphics processing unit
ICP	Iterative Closest Point (Besl and McKay, 1992)
KF	Kalman Filter
LRF	Laser range finder
LUM	Lu/Milios scan matching (Lu and Milios, 1997)
MAV	Micro aerial vehicle
MLE	Maximum likelihood estimation
MRF	Markov Random Field
MRSMAP	Muli-resolution surfel map (Stückler and Behnke, 2014)
MSAC	M-estimator sample consensus, a RANSAC variant by Torr and Zisserman (2000)
NDT	Normal distributions transform by Biber and Straßer (2003) and extended to 3D by Magnusson *et al.* (2007)
ORB	Oriented FAST and Rotated BRIEF (Rublee *et al.*, 2011)
PCA	Principal Component Analysis
PCL	Point Cloud Library, `http://pointclouds.org`
PPS	Probabilistic Plane Segmentation
RANSAC	Random sample consensus (Fischler and Bolles, 1981)
RGB-D	Color and depth, cameras acquiring both a color image (RGB) and a depth image (D).
RMSE	Root mean square error
RPE	Relative pose error (Sturm *et al.*, 2012)
SIFT	Scale-invariant feature transform (Lowe, 2004)

SLAM	Simultaneous localization and mapping
SSD	Smooth Signed Distance
SURF	Speeded-up robust features (Bay *et al.*, 2008)
SVM	Support vector machine
ToF	Time-of-flight

LIST OF SYMBOLS

In general, the following convention for symbols is used:

a, b, c	Scalar values (lower case Latin letters)
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	Vectors (lower case bold Latin letters)
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	Matrices (upper case bold Latin letters)
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Sets (upper case calligraphic Latin letters)
α, β, γ	Angles (lower case Greek letters)
$\hat{\boldsymbol{X}}$	Coordinate axes, e.g., $\hat{\boldsymbol{X}}$-axis, $\hat{\boldsymbol{Y}}$-axis, and $\hat{\boldsymbol{Z}}$-axis

In particular, the following symbols are used in this thesis:

$\boldsymbol{p}, \boldsymbol{q}$	Points in $\mathbb{R}^n$
P, Q	Point clouds, i.e., sets of points $\boldsymbol{p} \in P$ and $\boldsymbol{q} \in Q$
ϵ	Threshold ($epsilon > 0$), e.g.,
ϵ_d	distance threshold and
ϵ_θ	angular threshold.
$\mathcal{C}$	Set of (point) correspondences between two point clouds.
$\boldsymbol{T}$	(Homogeneous) transformation matrix (4×4).
$\boldsymbol{R}$	Rotation matrix (3×3).
$\boldsymbol{t}$	Translation vector (3×1).
$\boldsymbol{q} \in \mathbb{H}$	Unit quaternion as a representation for rotation (4×1).
$\boldsymbol{\Sigma}$	Covariance matrix.
$\boldsymbol{\Sigma}_i^P$	Covariance matrix for point $\boldsymbol{p}_i$, computed in a local neighborhood of points in point cloud P.
$\mathcal{G}$	A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices and edges.
$\mathcal{V}$	Set of vertices in a graph.
$\mathcal{E}$	Set of edges in a graph.
$\boldsymbol{e}$	Measurement error ($n \times 1$ vector).
$\boldsymbol{H}$	Information matrix ($n \times n$ matrix), inverse of the covariance matrix.

1 INTRODUCTION

Everything should be made as simple as possible, but no simpler.[1]

— Albert Einstein (1879–1955)

A problem well stated is a problem half solved.

— Charles Franklin Kettering (1876–1958)

The past decades have seen tremendous progress in robotics research. While we are still far away from the highly capable fictional robots and androids as imagined in novels and movies, real robotic systems have already revolutionized many domains especially automated manufacturing. Moreover, in recent years, a challenging new market is addressed, namely that of service robots. In contrast to controlled factory and laboratory environments, service robots operate in everyday environments, possibly in the vicinity of humans or even in collaboration with a human operator or co-worker. Applications in the health care domain, for example, envision autonomous robotic systems assisting the elderly to allow them to stay at home and remain independent past the point they would usually be unable to live alone. In other applications, robots help people with disabilities and take over duties and responsibilities of professional caregivers. The common goal of these applications is to increase the user's life quality and, at the same time, entail cost-effectiveness to the public expenditures on healthcare. However, operating in uncontrollable everyday environments comes at a price: it is far more challenging than operating an industrial robot behind fences that replays pre-programmed instructions and gets stopped in case of potential risks, e.g., when a human enters the workspace of the robot.

Robots operating in uncontrollable environments need advanced perception and cognition abilities in order to understand the situation and to react accordingly. In particular, autonomous mobile robots need a rough understanding of their surroundings in order to act in a goal-directed manner and plan actions effectively. Understanding the scene means to

- *perceive* spatial and visual information about the scene,
- *extract* spatial composition and geometry of rooms and environmental structures,

1 From *"It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience."* (Einstein, 1934)

- *segment* the scene into distinct objects, and
- *classify* them into distinct classes like walls, doors, tables, chairs or people.

Moreover, robots need a representation or *map* of their surroundings and the abilities to

- *localize* objects and themselves in such maps and to
- *build* and update such maps during operation.

Naturally, requirements for robotic applications are both 1) being fast to avoid longer interruptions in the workflow of the robot and 2) being robust to reliably provide correct results while avoiding errors such as false positives. Especially the last decade has brought up a myriad of sophisticated approaches addressing the aforementioned tasks. They vary in complexity—both algorithmic complexity and runtime/memory requirements—as well as in robustness and reliability. For many situations and tasks, however, a simple but robust approach can achieve the same or even better results than sophisticated related approaches in the same or even less time. That is, well-engineered simple solutions can be far more efficient (in terms of computation time and achievable results) than state-of-the-art solutions, especially if task and situation are constrained. If, for example, certain assumptions can be made about the task and the environment, exploiting these assumptions can considerably simplify the problem and make an approach for solving the problem less complex. In other words, one can overfit the approach to the problem to gain efficiency. While this comes at the price of loosing generality, it can allow simple but well-engineered solutions to outperform sophisticated general solutions. Investing such simple but efficient approaches is an essential part of this thesis.

This thesis focuses on environment perception and extracting relevant information for mobile manipulation and mapping. As the main sensing modalities, sensors capturing the three-dimensional (3D) geometry of the surroundings are used. These include 3D laser scanners, time-of-flight (ToF) cameras and consumer color and depth (RGB-D) cameras. The goal of this thesis is to develop simple but efficient approaches for problems in autonomous mobile robot perception and mapping as well as investigating how they compare to state-of-the-art approaches.

1.1 LIST OF CONTRIBUTIONS

Real-time RGB-D plane segmentation and object detection: A wide range of tasks in the domain of (domestic) service robots involves the manipulation of objects and pick and place tasks in particular. In order to avoid interruptions in the robot's operation, both perception and planning components need to be fast and ideally exhibit real-time applicability.

A segmentation pipeline specifically designed for ToF cameras is presented that efficiently detects horizontal support surfaces, objects and the shape of objects (Holz *et al.*, 2010). For object detection in real-time, another efficient pipeline is developed that makes use of several heuristics to achieve reliable detection of potential object candidates in table scenes (Holz *et al.*, 2011). Experimental results show reliable detections with the update rate of the used RGB-D camera.

The approach finds application in various other projects. In a joint work with Jörg Stückler and Ricarda Steffens, it is combined with an efficient approximate grasp planner in order to detect and grasp objects in table scenes and plan grasps in less than a second (Stückler *et al.*, 2013b). In a joint work with Angeliki Topalidou-Kyniazopoulou and Jörg Stückler, another perception and grasping pipeline was developed that allows for picking automotive parts from pallets with particularly low cycle times of only several seconds (Holz *et al.*, 2015a; Holz *et al.*, 2015b).

Fast and accurate segmentation of RGB-D and range images: A key aspect of scene understanding in the context of mobile robotics is the decomposition of the scene into environmental structures such as walls, floor and ceiling. In man-made environments, these structures are mainly composed of planes and other simple geometric primitives such as cylinders.

Focusing on planes to detect dominant planes and environmental structures, an efficient approach to range image segmentation is developed (Holz and Behnke, 2012; Holz and Behnke, 2014a). The approach first approximately reconstructs the sensed surface in RGB-D point clouds. The obtained mesh is then used to efficiently cache neighborhoods and relations, compute local features, smooth the underlying data, and segment the measured points into dominant planes and other geometric primitives.

In experiments, the proposed approach shows state-of-the-art performance for range image segmentation on publicly available datasets and data acquired using RGB-D cameras while being considerably faster than related approaches.

Fast registration and mapping of sparse 3D point clouds: In contrast to RGB-D cameras with dense depth and color images, the point clouds acquired by (continuously rotating) 3D laser scanners are sparse, especially when rotated fast. Registration of such data with standard registration algorithms suffers from effects resulting from the different point densities within and between individual scan lines of the point cloud.

For the registration of sparse laser scans acquired by an autonomous micro aerial vehicle (MAV), an efficient approach is

developed that makes use of information about the underlying surface to compensate for the different point densities (Holz and Behnke, 2014b). As in the range image segmentation approach, the surface information is obtained using approximate reconstruction, that is adapted to sparse laser scans. In order to build allocentric 3D maps, the registration algorithm is used in a complete simultaneous localization and mapping (SLAM) pipeline using graph-based pose optimization (Holz and Behnke, 2014c). By using multiple point correspondences between view poses instead of a single pose estimate, accurate 3D maps can be built even with inaccurate registration results.

In experiments, it is demonstrated that accurate 3D maps can be built from the acquired sparse laser scans in near real-time and without aggregation of the data (Holz and Behnke, 2015a).

Fast registration and mapping for RGB-D Cameras: Streams of RGB-D images pose quite different challenges on registration pipelines than the non-uniform density 3D laser scans: they are dense and arrive at high frame rates.

For the efficient registration of RGB-D images and mapping with RGB-D cameras, the registration and mapping approach is extended by aligning new images in local windows of the pose graph to minimize draft and by different subsampling steps to reduce the amount of data, e.g., the processed points per frame and the edges between connected frames (Holz and Behnke, 2015b).

Experiments show that the extended pipeline can reliably register streams of RGB-D images and build 3D maps with a moving RGB-D camera. Moreover, the approach can keep up with the performance of related approaches.

In a last series of experiments, the extensions are applied to the non-uniform density 3D laser scans again, showing an increase in performance for both initial alignments and final registrations (Razlaw *et al.*, 2015).

1.2 PUBLICATIONS

Parts of this thesis have been published in journals and conference proceedings. The publications are provided in chronological order:

- Holz, D., R. Schnabel, D. Droeschel, J. Stückler, and S. Behnke (2010). "Towards Semantic Scene Analysis with Time-of-Flight Cameras." In: *Proceedings of the RoboCup International Symposium*. Vol. 6556. Lecture Notes in Computer Science. Singapore, Singapore: Springer, pp. 121–132.

- Holz, D., S. Holzer, R. B. Rusu, and S. Behnke (2011). "Real-Time Plane Segmentation using RGB-D Cameras." In: *Proceedings of the 15th RoboCup International Symposium*. Vol. 7416. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, pp. 307–317.
- Holz, D. and S. Behnke (2012). "Fast Range Image Segmentation and Smoothing using Approximate Surface Reconstruction and Region Growing." In: *Proceedings of the 12th International Conference on Intelligent Autonomous Systems (IAS)*. Jeju Island, Korea.
- Stückler, J., R. Steffens, D. Holz, and S. Behnke (2013b). "Efficient 3D object perception and grasp planning for mobile manipulation in domestic environments." In: *Robotics and Autonomous Systems* 61.10, pp. 1106–1115.
- Holz, D. and S. Behnke (2014a). "Approximate triangulation and region growing for efficient segmentation and smoothing of range images." In: *Robotics and Autonomous Systems* 62.9, pp. 1282–1293. ISSN: 0921-8890.
- Holz, D. and S. Behnke (2014b). "Registration of Non-Uniform Density 3D Point Clouds using Approximate Surface Reconstruction." In: *Proceedings of the 45th International Symposium on Robotics (ISR) and 8th German Conference on Robotics (ROBOTIK)*. Munich, Germany.
- Holz, D. and S. Behnke (2014c). "Mapping with Micro Aerial Vehicles by Registration of Sparse 3D Laser Scans." In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS)*. Padova, Italy.
- Holz, D., A. Topalidou-Kyniazopoulou, M. R. P. Francesco Rovida, V. Krüger, and S. Behnke (2015a). "A Skill-Based System for Object Perception and Manipulation for Automating Kitting Tasks." In: *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Luxemburg.
- Holz, D., A. Topalidou-Kyniazopoulou, J. Stückler, and S. Behnke (2015b). "Real-Time Object Detection, Localization and Verification for Fast Robotic Depalletizing." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, pp. 1459–1466.
- Holz, D. and S. Behnke (2015a). "Registration of Non-Uniform Density 3D Laser Scans for Mapping with Micro Aerial Vehicles." In: *Robotics and Autonomous Systems* 74, Part B, pp. 318–330.
- Holz, D. and S. Behnke (2015b). "Approximate Surface Reconstruction and Registration for RGB-D SLAM." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Lincoln, UK.

- Razlaw, J., D. Droeschel, D. Holz, and S. Behnke (2015). "Evaluation of Registration Methods for Sparse 3D Laser Scans." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Lincoln, UK.

1.3 OUTLINE

The thesis is structured as follows:

SECTION 2: *Efficient Perception for Object Manipulation* presents mobile manipulation as a first application of real-time perception approaches. After discussing relevant related work, the developed real-time object segmentation pipeline is described. The section also presents the conducted experimental evaluation and discusses the achievable results and applications of the approach.

SECTION 3: *Efficient Segmentation of RGB-D Images* addresses the problem of segmenting 3D point clouds and identifying environmental structures. After a discussion of related work, it presents the proposed segmentation algorithm based on approximate surface reconstruction. The section also features an extensive comparative evaluation of the proposed approach and a variety of related works.

SECTION 4: *Registration and Mapping with Micro Aerial Vehicles* presents the problem of simultaneous localization and mapping (SLAM) as another important problem domain. It features both the presentation of the proposed approach to laser scan registration using approximate surface information and the presentation of the full SLAM approach for building allocentric 3D maps using registered laser scans. The section also features a discussion of related work and a comparative evaluation of different approaches and error metrics for both SLAM and the registration problem.

SECTION 5: *Registration and Mapping for RGB-D Cameras* presents the extensions of the SLAM pipeline to streams of RGB-D images including initial alignments in local windows and efficient subsampling. The section also features a discussion of related work and a comparative evaluation showing that the resulting pipeline compares well to related works for both RGB-D mapping and mapping with micro aerial vehicles.

SECTION 6: *Discussion and Conclusion* concludes the thesis, discusses the achieved results and outlines applications of the proposed approaches as well as future works.

Chapters 2 to 5 are written self-contained with each chapter containing an introduction, an individual discussion of related works, experiments and results.

Part I

PERCEPTION AND SEGMENTATION

... in which robotic environment sensors are introduced and the central question is how to process the acquired data efficiently. In this thesis, two applications are distinguished: 1. perceiving certain parts of the environment that are relevant for a particular task like mobile manipulation and 2. complete segmentations of input data into coherent parts for further processing and applications like scene understanding.

Perception for (mobile) manipulation aims at extracting task-relevant regions in the input data such as finding the object to manipulate and its support surface as well as detecting obstacles in the surroundings of the object and the robot that may restrict applicable manipulation motions.

The problem of segmentation is to (completely) partition the input data into coherent regions or clusters. A prominent example which is also the main problem addressed in this thesis is planar segmentation, i.e., partitioning input data into planar segments. Such a partition forms a simplification of the scene while still containing the relevant information about dominant environmental structures such as walls, floor, ceiling and table surfaces. Segmentation into planes and other geometric primitives forms a fundamental pre-processing step in a large variety of applications such as estimating the three-dimensional (3D) scene geometry, building planar environment maps or semantic scene classification.

Both perception for manipulation and segmentation share that the amount of data is typically large and that the involved computations are expensive. On the contrary, in order to avoid longer interruptions in the workflow of the robot, the data needs to be processed fast. The following chapters particularly focus on deducing and exploiting simplifying assumptions that allow for fast yet reliable perception pipelines.

2 EFFICIENT PERCEPTION FOR OBJECT MANIPULATION

Real-time 3D perception of the surrounding environment is a crucial precondition for the reliable and safe application of mobile service robots in domestic environments. In this chapter, we present a system for acquiring and processing 3D (semantic) information at frame rates of up to 30 Hz. It allows a mobile robot to reliably detect obstacles as well as to segment graspable objects and supporting surfaces.

After a brief introduction and a discussion of related works, we first present a table top perception pipeline that was specifically designed for time-of-flight (ToF) cameras. In contrast to other approaches working with highly detailed and accurate 3D laser scans, the pipeline features several processing steps for coping with the special measurement characteristics of ToF cameras. For being able to acquire semantic information from ToF camera data, we 1. pre-process the data including outlier removal, filtering and phase unwrapping for correcting erroneous distance measurements, and 2. apply a randomized algorithm for detecting shapes such as planes, spheres, and cylinders (e.g., for planning possible grasps and to obtain a parameterized representation of the scene). We present experimental results that show that the robustness against noise and outliers of the underlying random sample consensus (RANSAC) paradigm allows segmenting objects in 3D ToF camera data captured in natural mobile manipulation setups.

In the second part, we then present several extensions to the original pipeline that make it particularly efficient and robust for object detection with color and depth (RGB-D) cameras. By restructuring the pipeline and replacing several processing steps with highly efficient approximations, the resulting pipeline can detect potential object candidates in table top scenes at 30 Hz, i.e., with the frame-rate of the RGB-D camera. The system is tested in different setups in a real household environment. The results show that the system is capable of reliably detecting objects and obstacles at high frame rates, even in case of objects that move fast or do not considerably stick out of the ground. The segmentation of all planes in the 3D data even allows for correcting characteristic measurement errors and for reconstructing the original scene geometry in far ranges. We present experimental results that show that the extended pipeline can reliably segment horizontal support surfaces and potential object candidates thereon in real-time.

Finally, we present two applications of the real-time segmentation pipeline: 1. real-time object perception and grasp planning where the segmented regions are used efficiently grasp objects without prior

knowledge about the objects, and 2. mobile robot depalletizing where the pipeline is used to detect objects on transport pallets and grasp them. For both applications we present proof-of-concept experiments and results. The performance of the approaches is measured using two criteria: the success rate for detecting and grasping parts, and the cycle time, i.e., the time it takes for detecting and grasping the part including motion planning and execution.

2.1 INTRODUCTION

Autonomous mobile robots need environment models in order to plan actions and navigate effectively. Two-dimensional metric maps, built from two-dimensional (2D) laser range scans, became the de-facto standard to tackle navigation problems such as path planning and localization of the robot platform. For planning arm motions and grasps, however, 3D (semantic) information becomes crucial since:

1. Objects need to be detected in the presence of other objects (e.g., on a cluttered table).
2. The robot needs to determine whether or not an object is graspable (e.g., with respect to its size).
3. The robot needs to determine the 3D pose of the object as a goal for its end-effector.
4. The robot needs to determine the 3D pose (and boundaries) of neighboring objects in order to plan an obstacle-free path.

The context of the work presented here are different projects on mobile manipulation including larger European projects and the RoboCup@Home league. The latter addresses service robot applications and focuses on navigation (and SLAM) in dynamic environments, mobile manipulation and human-robot-interaction. A typical task for a mobile (service) robot is the manipulation of objects like, for instance, retrieving objects for a human user. Besides the interaction with the user, these tasks require the robot to be capable of safely navigating in cluttered and dynamic environments, reliably perceiving the involved objects under varying conditions, and planning and executing motions for manipulating the objects.

Mobile manipulation in domestic environments has seen a lot of progress recently and many different platforms have been developed and presented in the past years. A very prominent example is the PR2 developed by Willow Garage (Meeussen *et al.*, 2010). It is equipped with two 7 degree-of-freedom (DoF) compliant arms and a parallel gripper with touch sensor matrices on the gripper tips. Leeper *et al.* (2012) use the system in a tele-operated setting. Besides directly controlling the robot's end effector, the user can follow different strategies

for grasping objects. In one of the strategies, the user selects a grasp from a set of feasible poses suggested by the planner of Hsiao *et al.* (2010). Bohren *et al.* (2011) used the PR2 platform for opening a refrigerator, grasping beverages, and delivering them to human users. Beetz *et al.* (2011) presented a demonstration where a PR2 collaboratively prepared pancakes with another custom built robot.

Another popular (research) platform is the Baxter robot developed and marketed by Rethink Robotics[1]. It is designed for manipulation tasks in close collaboration with human users. The arms and grippers, for example, can be easily moved by users to teach motions to the robot. The robot has two 7-DoF arms and different grippers like two-finger grippers and suction cups. It also features a display, e.g., for communicating with the user or visualizing internal information.

The German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt; DLR) has developed Rollin' Justin, a mobile manipulator with a particularly compliant whole-body motion design. Bäuml *et al.* (2011) presented a demonstration with Rollin' Justin where the robot prepared coffee for human users. The task involved opening and closing the pad drawer, grasping coffee pads and putting them into the pad machine.

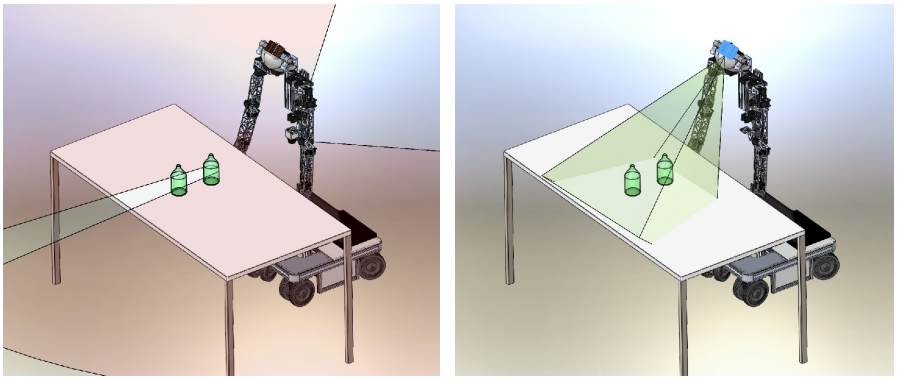
Fraunhofer IPA has developed the Care-O-Bot platform and its successors (Hans *et al.*, 2002). The platform is designed for both mobile manipulation and intuitive interaction.

Jain and Kemp (2010) presented the robot platform EL-E, a mobile manipulator that shall assist the impaired. The user can draw the robot's attention to objects on tables and the floor by pointing on the objects with a laser pointer. The robot then picks the object up and delivers it to the user.

Srinivasa *et al.* (2008) presented a mobile tray that delivers mugs to a statically mounted manipulator. The tray navigates through visual ceiling markers to the predefined position of the manipulator. The manipulator then grasps the mugs from the tray and loads them into a dishwasher rack. The mugs are perceived using a real-time vision system specifically designed for the mugs. For the mobile manipulator HERB, Srinivasa *et al.* (2010) extended the vision system to include a more general object recognition approach.

The above mentioned examples primarily include different platforms for mobile manipulation. However, it is not the design of a platform which solves a task but the robot's abilities to perceive its environment, to reason about the task, its knowledge and what it has observed, and to act (and react) in a reasonable way. In this chapter, we focus on a crucial part of the perception of objects namely detecting potential object candidates in table top scenes. This problem is also often referred to as *object discovery*. We present a pipeline for finding

1 http://www.rethinkrobotics.com/baxter



(a) 2D laser perception (b) 3D camera perception

Figure 2.1: Perceiving objects on the table. With the trunk laser scanner (a), the second object is occluded and not perceived. Using a 3D camera (b), mounted in the robot's head allows perceiving both objects.

potential object candidates (and grasping them) without the need for a special vision system or object models known a priori.

In previous work (Stückler and Behnke, 2009), we used a 2D laser scanner for detecting possible object locations on tables. The 2D positions of the object candidates were then projected into the image plane of a color camera for feature-based object recognition. In case of a successful recognition (i.e., the object to deliver was among the candidates), grasping of the object was initiated. The drawback of this approach (as illustrated in Figure 2.1) is that objects can occlude each other in the two-dimensional measurement plane of the laser range scanner. In this chapter, we focus on detecting potential object candidates with 3D cameras including ToF cameras and RGB-D cameras.

The two pipelines presented in this chapter were first published and presented at the International RoboCup Symposium in 2010 (Holz *et al.*, 2010) and 2011 (Holz *et al.*, 2011).

2.2 RELATED WORK

In typical household environments objects are usually constrained in their position to well defined parts like, for instance, table tops, shelves and other horizontal support planes. This natural restriction of space is exploited in the majority of approaches to object perception and search. A common processing scheme (Rusu *et al.*, 2009b; Holz *et al.*, 2010) and perception pipeline for detecting and recognizing objects in depth images and 3D point clouds is to

1. detect the horizontal support planes,
2. extract and cluster the measurements on top of these planes, and

3. perform further processing, e.g., recognizing, classifying or tracking the found clusters.

Differing in related works are, most notably, the used methods for the individual processing steps that determine, amongst others, the robustness, speed, and runtime requirements of the overall system.

Rusu *et al.* (2009b) propose to segment point clouds into objects on planar surfaces. They suggest to use RANSAC to detect planes and to extract shape primitives on the objects. Remaining points are described by meshes. Schnabel *et al.* (2007) decompose noisy point cloud data into geometric shape primitives with an efficient multi-resolution approach. In this chapter, we combine planar pre-segmentation as done by Rusu *et al.* (2009b) with efficient object modeling using shape primitives (Schnabel *et al.*, 2007), and make the approaches applicable to the measurements of ToF cameras. Amongst others, we present techniques to cope with the specific error sources of the cameras, to speed up processing by exploiting the image-like data organization, and for detecting geometric primitives in the found object clusters.

Rusu *et al.* (2009c) extract shape primitives and use them as obstacles for a motion planner. Sucan *et al.* (2010) extend this approach to identify areas of a scan that are occluded by the robot. They maintain these areas from a sequence of scans while the robot is moving. In this way, the robot can still avoid obstacles that it occludes during its motion. Muja *et al.* (2011) present a complete open source framework for detecting, recognizing, and localizing objects using intensity images. It allows for combining different detection and recognition methods in order to achieve reliable results in varying settings.

For discovering objects in 2D images a rich literature of algorithms is available. While we focus on RGB-D object discovery, we refer to the overview and comparison of Tuytelaars *et al.* (2010) for object discovery approaches in 2D images.

Collet *et al.* (2011) present a framework for object segmentation that combines and fuses color and depth information. They first segment the scene into regions based on color and depth and then estimate for each region how much structure is contained. This estimation is based on simple image and range features and separates objects from the mostly planar background. In experiments, they can show that the highest ranked regions w.r.t. to the measure of *structureness* belong to the most dominant objects in the scene, e.g., objects on a table top.

For oversegmenting RGB-D images, Weikersdorfer *et al.* (2012) present an extension of superpixels from the 2D image domain to include depth. Papon *et al.* (2013) present a similar supervoxel-based oversegmentation of RGB-D images that uses different voxel relationships in order to produce accurate oversegmentations that are fully consistent with the 3D geometry of the scene. Such oversegmentations are used, for example, by Stein *et al.* (2014) who first oversegment RGB-D images and then connect clusters to objects primarily by focusing on convex-

ity. That is, they make the assumptions that objects are convex and that segments belonging to the same object belong completely to that object and show relationships of convexity to other segments of the object. Moosmann *et al.* (2009) also use a local convexity criterion in a graph-based approach for segmenting the ground and objects on the ground in 3D laser data.

Richtsfeld *et al.* present a multi-stage pipeline that also starts with an oversegmentation of the input image and the estimation of surface patches using planes and non-uniform rational B-splines (Richtsfeld *et al.*, 2012; Richtsfeld *et al.*, 2014). In the next step, a graph is constructed representing pairwise relationships between surface patches. Support vector machines (SVMs) are used to classify the relations and to derive the grouping of patches. The approach can reliably segment objects in cluttered scenes, even when the objects are stacked or touching one another. In our pipeline, we primarily aim for efficiency and only address detecting regions of interest, rather than accurately segmenting the regions into individual objects.

Karpathy *et al.* (2013) also start from a pre-segmentation of the scene and then estimate for each segment a measure of *objectness*, i.e., how likely the segment belongs to an object and not to the background or other environmental structures. This estimation is based on different intrinsic shape measures such as compactness, convexity, symmetry and smoothness. Especially the symmetry criterion helps in distinguishing man-made objects from cluttered backgrounds as was shown, for example, by Palmer (1985). Triebel *et al.* (2010) pre-segment RGB-D images using a graph-based clustering in both geometric space and feature space, and use a conditional random field (CRF) to label extracted regions. In an extension of the approach, previously unknown objects are extracted from cluttered scenes based on whether or not they appear multiple times in an image (Shin *et al.*, 2010). In our approach we do not make assumptions about the reappearance of objects or assumptions about the convexity, symmetry or smoothness of the objects' shapes.

A popular approach for object discovery is to use the biologically inspired concept of saliency. Leroy *et al.* (2015) first segment RGB-D images into supervoxels and then use bottom-up saliency maps to identify how salient the individual voxels are. A similar approach is followed by Frintrop *et al.* (2014) who compute both a segmentation of the images and a saliency map. Thresholding in the saliency map then allows for identifying salient regions in the segmented image. The approaches find similar results as our pipeline for objects on tables and shelves whereas we explicitly make assumptions about these horizontal surface instead of using saliency.

Herbst *et al.* (2011a) exploit changes in the environment in a sequence of RGB-D images in order to detect and model moving objects. Herbst *et al.* (2011b) then extend to approach to use a multi-scene

Markov Random Field (MRF) as well as shape, visibility, and color cues. In a similar fashion Mason *et al.* (2012) detect objects by disappearance, i.e., when a certain region of the image is no longer present and does no longer occlude other parts of the scene. Ma and Sibley (2014) present a complete framework for estimating the camera motion in a scene as well as segmenting, tracking, and modeling moving objects. In our pipeline, we do not assume and require objects to move or disappear, but instead segment single RGB-D images.

2.3 TABLE TOP SEGMENTATION WITH TOF CAMERAS

One of the first applications in robotics considering ToF cameras as an alternative to laser scanning has been presented by Weingarten *et al.* (2004) who evaluated a SwissRanger SR-2 camera in terms of basic obstacle avoidance and local path-planning capabilities. Another early application was human-assisted 3D mapping in the context of the RoboCup Rescue league as presented by Sheh *et al.* (2006). Ohno *et al.* (2006) used a SwissRanger SR-2 camera for estimating the trajectory of a mobile robot and reconstructing the surface of the environment. More recently, May *et al.* (2009) presented and evaluated different approaches for registering multiple range images of a SwissRanger SR-3000 camera in the context of a fully autonomous 3D mapping system.

All the aforementioned approaches have shown that ToF cameras require taking care of their complex error model (May *et al.*, 2009). The specific characteristics of ToF cameras can cause spurious measurements that do not correspond to any object in the real physical environment. Several works address the sensor characteristics of ToF cameras as well as methods for calibrating the cameras (Fuchs and Hirzinger, 2008) or filtering noisy and erroneous measurements (May *et al.*, 2006; Fuchs and May, 2007; Pathak *et al.*, 2008). In our pipeline we include pre-processing steps that address these problems. In a post-processing step, we use the approach of Schnabel *et al.* (2007) to detect geometric shape primitives in the detected object clusters. The rest of the pipeline resembles a standard so-called *table-top segmentation* pipeline (Rusu *et al.*, 2009b).

2.3.1 *Pipeline Overview*

The pipeline is split into three stages: pre-processing, segmentation, and post-processing. Referring to Figure 2.2, input point clouds are first filtered to cope with the special characteristics of ToF cameras. Using a probabilistic graphical model, measurements larger than the maximally measurable distance are reconstructed by projecting them into the correct distance interval (distance unwrapping). A jump edge filter then removes spurious measurements between occluding and

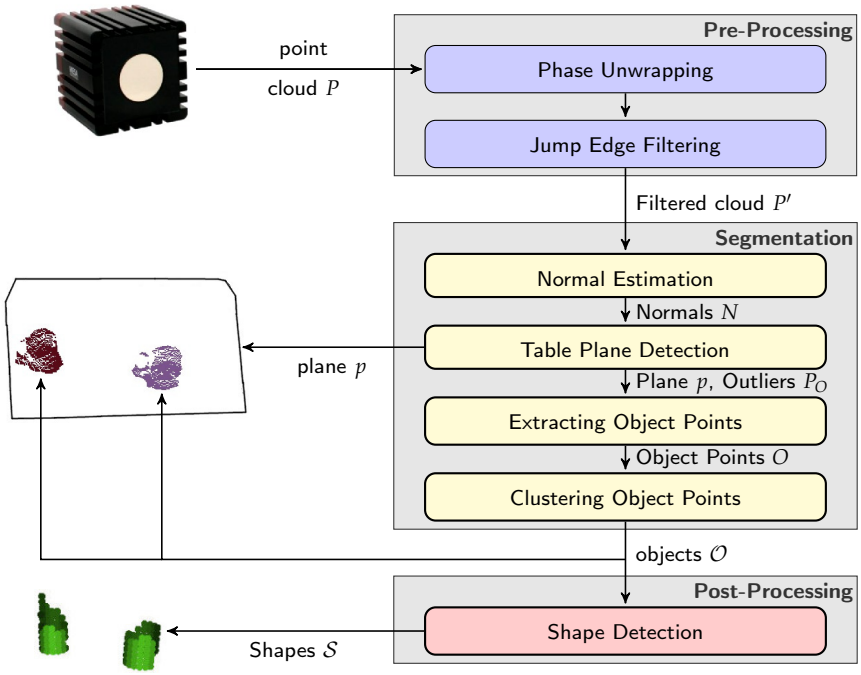


Figure 2.2: Overview of the object detection pipeline: input point clouds are filtered to cope with the special characteristics of ToF cameras. In the filtered point clouds, we first compute local point features and then detect the support plane and cluster the points above the plane. In a post-processing step, we detect geometric shape primitives and build the final parameterized model of the scene.

occluded surfaces. For the actual segmentation of the filtered point cloud, we first compute local features such as surface normal and curvature which are used in the various processing steps thereafter. The segmentation then continues to detect the dominant (horizontal) planes in the scene. The plane inliers are extracted and for the remaining points it is checked whether or not they lie above the detected plane. The points that are found to lie above then plane are then clustered to obtain potential object candidates and regions for further processing steps. As one application of the found object clusters, we detect geometric shape primitives in a post-processing step and build a parameterized environment representation for mobile manipulation. This representation models the support plane and its boundary as well as the geometric shape of objects on the support plane.

The results of the pipeline are 1. a model for the support plane p that includes the plane equation in Hesse normal form and its contour (we use the convex hull of the plane inliers) and 2. the set of objects $\mathcal{O}$, where every object $o_i \in \mathcal{O}$ is represented by a subset of the original filtered point cloud P' of points belonging to the object as well as the centroid of the object, its principal axes and the dimensions along the principal axes using Principal Component Analysis (PCA). In addition, it provides for every object $o_i \in \mathcal{O}$ the set of detected shapes $\mathcal{S}_i$ and

the set of remaining points R where no shape was detected (usually empty or containing few outliers).

2.3.2 *Preprocessing*

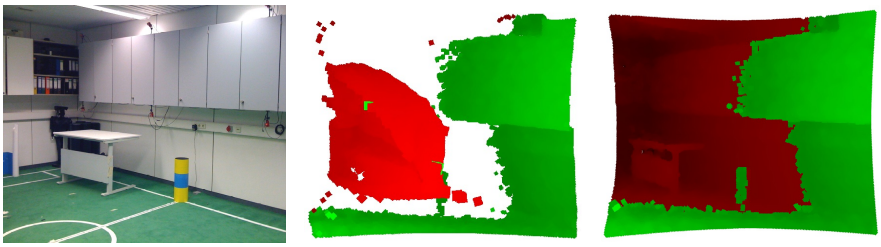
Besides a large variety of systematic and non-systematic errors, ToF cameras show two problems that are characteristic for their measurement principle. Both problems cause spurious measurements that do not correspond to any object in the real physical environment. The first problem is the ambiguity of distance measurements. The second problem is that the acquired point clouds contain phantom measurements occurring at distance discontinuities, i.e., at the boundaries of surfaces partially occluding each other. By means of phase unwrapping and jump edge filtering both problems are addressed when pre-processing the acquired point clouds.

2.3.2.1 *Phase Unwrapping*

ToF cameras illuminate the environment by means of an array of LEDs that emit amplitude-modulated near-infrared light. The reflected light is received by a CCD/CMOS chip. Depth information is gained for all pixels in parallel by measuring the phase shift between the emitted and the reflected light. This phase shift is proportional to the object's distance to the sensor modulo the wavelength of the modulation frequency. This characteristic results in a distance ambiguity. That is, objects farther away from the sensor than the maximum measurable distance $d_{\max}$ are, respectively, *wrapped* and projected into the interval $[0, d_{\max}]$. An example of a range image that contains such wrapped distances can be seen in Figure 2.3.

A common way to handle these distance ambiguities is to neglect measurements based on the ratio of measured distance and intensity as done by May *et al.* (2009). The amplitude of the reflected signal decreases quadratically with the measured distance. Sorting out points not following this scheme, e.g., points with a low intensity at a short distance, removes the majority of wrapped measurements but also valid measurements on less reflective surfaces.

In contrast to these approaches, we correct the wrapped measurements instead of neglecting them. We apply phase unwrapping techniques to reconstruct depth measurements behind the sensor's non-ambiguity range. The goal of phase unwrapping is to infer a number of phase jumps from the wrapped signal. Under the assumption that neighboring measurements are more likely close to each other than farther apart, relative phase jumps between neighboring pixels can be extracted. The signal can be unwrapped by integrating these phase jumps into the wrapped signal. We use a probabilistic approach based on the work of Frey *et al.* (2001) that relies on discontinuities in the image to infer these phase jumps. In addition to depth discontinuities,



(a) Scene (b) Wrapped cloud (c) Unwrapped cloud

Figure 2.3: Principle of phase unwrapping. By correcting the depth image at detected phase jumps, we can obtain valid measurements from objects being farther away from the sensor than the maximum measurable distance $d_{\max}$. Here the red measurements need to be corrected, the green points naturally lie in the interval $[0, d_{\max}]$.

we incorporate the intensity of the reflected signal, since it depends on the object's distance and can indicate inconsistencies between a measured and the corresponding real distance. An example of an unwrapped range image can be seen in Figure 2.3. For more details on the approach, we refer to the original paper by Droeschel *et al.* (2010a). An extension of the approach is to use multiple different modulation frequencies in order to reliably identify wrapped distance measurements (Droeschel *et al.*, 2010b).

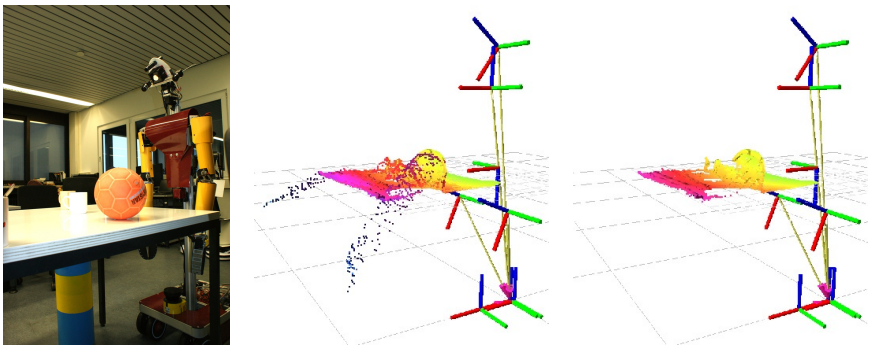
2.3.2.2 *Jump Edge Filtering*

Jump edges are known to cause spurious measurements that should either be corrected or neglected when processing ToF depth information. For simply neglecting these measurements, sufficient results are achieved by examining local neighborhood relations. From a set of 3D points $P = \{\boldsymbol{p}_i \in \mathbb{R}^3 | i = 1, \cdots, N_p\}$, jump edges J can be determined by comparing the opposing angles $\theta_{i,n}$ of the triangle spanned by the focal point $\boldsymbol{f} = \boldsymbol{0}$, point $\boldsymbol{p}_i$ and its eight neighbors $\mathcal{P}_i(n) = \{\boldsymbol{p}_{i,n} | i = 1, \cdots, N_p : n = 1, \cdots, 8\}$ with a threshold θ_{th}:

$$\theta_i = \max \arcsin \left(\frac{||\boldsymbol{p}_{i,n}||}{||\boldsymbol{p}_{i,n} - \boldsymbol{p}_i||} \sin \varphi \right), \tag{2.1}$$

$$J = \{\boldsymbol{p}_i | \theta_i > \theta_{th}\}, \tag{2.2}$$

where φ is the apex angle between two neighboring pixels. That is, neighboring points that lie on a common line-of-sight to the focal point $\boldsymbol{f}$ are, respectively, removed from the point cloud and marked as being invalid. A typical result of applying this filter is shown in Figure 2.4. As can be seen, the table occluding the ground planes causes spurious measurements in between the two surfaces. The jump edge filter reliably removes these points just like the spurious measurements between the objects on the table and the table plane.



(a) Scene (side view) (b) Unfiltered cloud (c) Filtered cloud

Figure 2.4: Sorting out measurements at jump edges. Shown are a photo of an example scene (a), the captured unfiltered point cloud (b) and the filtered cloud (c). It can be seen that the majority of erroneous measurements caused by jump edges, e.g., between table and floor in (b), are sorted out in the filtered cloud (c).

The detection of tables and objects in the filtered point clouds is conducted in three steps: We first compute local surface normals and variations for all points. This information is then used to detect larger horizontal planes and fitting corresponding planar models into the data. Points above these planes but inside their boundaries are then clustered in order to form the object candidates for further processing.

2.3.3 *Normal and Curvature Estimation*

A common way for determining the normal to a point $\boldsymbol{p}_i$ on a surface is to approximate the problem by fitting a plane to the point's local neighborhood $\mathcal{P}_i$ in the least squares error sense. This neighborhood is formed either by the k nearest neighbors or by all points within a radius r from $\boldsymbol{p}_i$.

Searching for nearest neighbors is computationally expensive. Even specialized algorithms like approximate search in *kd*-trees (Mount and Arya, 1997) can cause longer runtimes when building the search structure for a larger point set. Instead of really searching for nearest neighbors, we approximate the problem and exploit the order of the measurements in the point cloud (176×144 distance measurements). We build a lookup table storing, for every point index, the ring-neighborhood being formed by the k closest indices in index space. That is, starting from $\boldsymbol{p}_i$ we circle around the image index $(x = i \mod 176, y = i/176)$ in anti-clockwise order and store the point index for the traversed pixels in the lookup table. When processing a new point cloud we only update the squared distances from every point $\boldsymbol{p}_i$ to its k neighbors as provided by the lookup table.

The approximated nearest neighbors do not resemble the true nearest neighbors in the vicinity of transitions between different surfaces

or partial occlusions, and if the point cloud is highly affected by noise and erroneous measurements. To take this into account, we check the computed squared distances and mark those being larger than some threshold r^2 as being invalid. That is, our local neighborhood $\mathcal{P}_i$ is bounded by both a maximum number of neighbors k and a maximum distance r.

Given the local neighborhood $\mathcal{P}_i$, the local surface normal $\boldsymbol{n}_i$ can be computed by fitting a plane through the points in $\mathcal{P}_i$ (Shakarji, 1998; Rusu, 2009) . The computed tangent plane goes through the centroid $\bar{\boldsymbol{p}}_i$ of $\mathcal{P}_i$ and its normal is the local surface normal $\boldsymbol{n}_i$. With the centroid

$$\bar{\boldsymbol{p}}_i = \frac{1}{\|\mathcal{P}_i\|} \sum_{j=1}^{\|\mathcal{P}_i\|} \boldsymbol{p}_j, \tag{2.3}$$

where $\|\mathcal{P}_i\|$ is the number of points in the local neighborhood $\mathcal{P}_i$, the surface normal can be estimated by analyzing the eigenvectors of the covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$ of $\mathcal{P}_i$ through Principal Component Analysis (PCA):

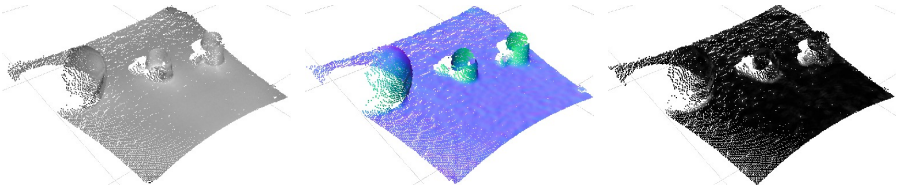
$$\boldsymbol{\Sigma}_i = \frac{1}{\|\mathcal{P}_i\|} \sum_{j=1}^{\|\mathcal{P}_i\|} \left(\boldsymbol{p}_j - \bar{\boldsymbol{p}}_i\right) \left(\boldsymbol{p}_j - \bar{\boldsymbol{p}}_i\right)^T, \text{ and} \tag{2.4}$$

$$\boldsymbol{\Sigma}_i \boldsymbol{v}_{i,j} = \lambda_{i,j} \boldsymbol{v}_{i,j}, j = \{0,1,2\}. \tag{2.5}$$

An estimate of $\boldsymbol{n}_i$ can be obtained from the eigenvector $\boldsymbol{v}_{i,0}$ corresponding to the smallest eigenvalue $\lambda_{i,0}$. The ratio between the smallest eigenvalue and the sum of eigenvalues provides an estimate of the local curvature κ_i or local variance:

$$\kappa_i = \frac{\lambda_{i,0}}{\lambda_{i,0} + \lambda_{i,1} + \lambda_{i,2}}. \tag{2.6}$$

The eigenvector corresponding to the smallest eigenvalue is only an approximation of the local surface normal and its orientation is ambiguous since it depends on the order and positions of the points in the local neighborhood. In order to obtain consistent normal orientations, we flip normals pointing away from the sensor towards the sensor. Also, at surface boundaries the approximated normals can be inaccurate since the local neighborhoods may contain points from more than one surface. Rusu (2009) proposed to compute the centroids and the normals in two stages where the best fitting plane through the points is first computed in a RANSAC-based approach in order to determine inliers and outliers. The distances of the points in the neighborhood to the best fitting plane are then used to derive weights (per point) for the computation of the centroid $\bar{\boldsymbol{p}}_i$ and the covariance matrix $\boldsymbol{\Sigma}_i$ in Equations (2.3) and (2.4). In order to avoid sweeping through the data twice, we accept the inaccuracies at boundaries and



(a) Intensities (b) Normals (c) Curvature

Figure 2.5: Computing local surface normals and curvature. Shown are the input point cloud with intensity information (a) as well as the computed surface normals (b) and local curvature changes (c). The used parameters are $k = 40$ and $r = 5\,\text{cm}$.

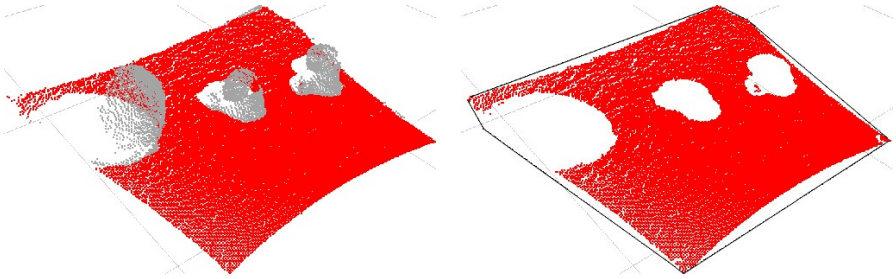
compute the local surface normal approximations in a single sweep with constant weights (the weights are neglected in the equations).

Figure 2.5 shows a typical example of local surface normals and curvature changes as computed for a pre-processed point cloud with normal and curvature estimates. The aforementioned neighborhood approximation by using fixed lookup tables drastically decreases the computational complexity of the surface normal and curvature estimation. However, more important is that the involved inaccuracies did not considerably degrade the results in our experiments.

2.3.4 *Table Plane Detection*

In order to detect the table plane, we exploit the assumption that objects are usually standing on horizontal surfaces. For detecting tables in the vicinity of the robot, we extract those points $\boldsymbol{p}_i$ from the point cloud that are close to the robot, e.g., not more than 2 m away and that lie in a height in which we expect planes, e.g., above the ground and no higher than 1.5 m. In order to obtain an efficient representation of tables and to segment individual objects, we fit a planar model to the extracted point set using the M-estimator sample consensus (MSAC) framework, an extension to the well-known RANSAC paradigm where inliers receive a certain score depending on how well they fit the data (Torr and Zisserman, 2000). This M-Estimator is particularly robust against noise and outliers. For other variants of the RANSAC paradigm we refer to the extensive overview and evaluation of Choi *et al.* (2009). In the detection, we restrict the MSAC to only use horizontal planes, i.e., planes whose plane normals are not roughly parallel to the $\hat{\boldsymbol{Z}}$-axis are neglected. The planes are expressed with normal $\boldsymbol{n}$ and distance d to the origin. In order to identify the inliers, we use the following constraints:

1. the point is close to the computed plane, i.e., $\|\boldsymbol{n} \cdot \boldsymbol{p}_i + d\| < \epsilon_d$
2. the surface normal $\boldsymbol{n}_i$ points along the plane normal, i.e., $\boldsymbol{n}_i \parallel \boldsymbol{n}$,
3. the surface around $\boldsymbol{p}_i$ is smooth (i.e., $\kappa_i \approx 0$).



(a) Inliers supporting the model (b) Polygonal approximation

Figure 2.6: Detecting the table plane. Shown are the inliers (red) supporting the planar model (a) as well as the 3D polygonal approximation (b) formed by the two-dimensional convex hull of the inliers (projected onto the table plane).

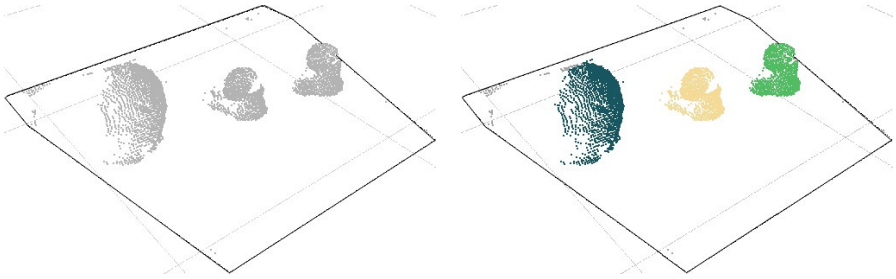
For the point cloud from Figure 2.5, all extracted points belong to the same table. The result of fitting a planar model to the points is shown in Figure 2.6. The planar model that best fits the data is almost parallel to the xy-plane and is supported by 20 731 inliers. It can already be seen that, despite some points on the table's boundaries, the outliers correspond to the objects on the table.

Once the set of inliers is determined, we re-compute the planar model as in Equation (2.3) and Equation (2.4) where $\mathcal{P}_i$ is replaced by the set of all plane inliers. Once the planar model has been found, we project all inliers onto the detected plane and compute the 2D convex hull by means of Graham's Scan Algorithm (Graham, 1972). The convex hull for the 20 731 points from Figure 2.6 consists of 9 points and accurately represents the surface of the table top.

2.3.5 *Object Detection*

All outliers from fitting the planar model as well as the points that have not been considered for the table point set T are potential object points. That is, they could have been measured on the surface of an object. Since we are only interested in objects on top of the table, we first sort out all points lying below the table plane as well as those points that do not lie within the bounding polygon. In order to obtain point sets that represent a common object, we apply a simple clustering based on the Euclidean distance between neighboring points. Neighboring points whose point-to-point distance is below a threshold $d_{\max}$ are recursively merged into clusters. Clusters whose cardinality exceed a minimum number $n_{\min}$ of support points are considered as object candidates.

The resulting segmentation of the ongoing examples from Figure 2.5 and Figure 2.6 is shown in Figure 2.7. In order to use the segmented object clusters for grasp and motion planning, we compute the centroid as well as the oriented bounding box for all points in each cluster.



(a) Outliers from planar fit (b) Object candidates

Figure 2.7: Detecting object candidates. Shown are the unclustered outliers (a) and the object candidates (b) obtained from Euclidean clustering. Object candidates are colored. The remaining points are gray. Here, the parameters are $d_{\max} = 2.5\,\text{cm}$ and $n_{\min} = 250$.

2.3.6 *Object Shape Detection*

After obtaining the object candidates in the form of point clusters, we now present a possible post-processing step to determine geometric primitives, e.g., to ease grasp planning and obtaining a parameterized scene representation. In order to robustly detect different kinds of geometric primitives like planes, spheres, and cylinders, we employ an efficient RANSAC algorithm that directly operates on the extracted clustered point clouds and the associated surface normal information. In our setting we can closely follow a simplified version of the approach proposed by Schnabel *et al.* (2007). While the original method focuses on achieving efficiency even on huge point clouds, the point clouds in the considered application are comparatively small and thus not all optimizations worthwhile.

Given a point cloud $P = \{\boldsymbol{p}_i \in \mathbb{R}^3 | i = 1, \ldots, N_p\}$ with associated normals $\{\boldsymbol{n}_i, \ldots, \boldsymbol{n}_{N_p}\}$, the output of the algorithm is a set of primitive shapes $\Psi = \{\psi_1, \ldots, \psi_n\}$ with corresponding disjoint sets of points $P_\Psi = \{P_{\psi_1} \subset P, \ldots, P_{\psi_n} \subset P\}$ and a set of remaining points $R = P \setminus \bigcup_\psi P_\psi$.

Similar to Roth and Levine (1993) and Décoret *et al.* (2003), the shape extraction problem is framed as an optimization problem defined by a score function σ_P. In each iteration of the algorithm, the primitive with maximal score is searched using the RANSAC paradigm. New shape candidates are generated by randomly sampling minimal subsets of the point cloud P. Candidates of all considered shape types are generated for every minimal set and all candidates are collected in the set $\mathcal{C}$. Thus, no special ordering has to be imposed on the detection of different types of shapes. After new candidates have been generated, the candidate m with the highest score is computed employing an efficient lazy score evaluation scheme. The best candidate is only accepted if, given the number of inliers $|m|$ of the candidate and the number of drawn candidates $|\mathcal{C}|$, the probability that no better

candidate was overlooked during sampling is high enough (Fischler and Bolles, 1981). If a candidate is accepted, the corresponding points P_m are removed from P and the candidates $\mathcal{C}_m$ generated with points in P_m are deleted from $\mathcal{C}$. The algorithm terminates as soon as the probability of detection for a shape with a user defined minimal size τ is large enough.

2.3.6.1 *Shape Estimation*

The shapes we consider in this work are planes, spheres, and cylinders with different numbers of parameters. Every 3D point $\boldsymbol{p}_i$ can fix only one parameter of the shape. In order to reduce the number of points in a minimal set, we also use the approximate surface normal $\boldsymbol{n}_i$ for each point, so that the direction gives us two more parameters per sample. That way it is possible to estimate each of the considered basic shapes from at most three point samples. However, always using one additional sample is advantageous because the surplus parameters can be used to immediately verify a candidate and thus eliminate the need of evaluating many relatively low scored shapes (Matas and Chum, 2002).

For a plane, $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\}$ constitutes a minimal set when not taking into account the normals in the points. To confirm the plausibility of the generated plane, the deviation of the plane's normal from $\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3$ is determined and the candidate plane is accepted only if all deviations are less than the predefined angle α.

A sphere is fully defined by two points with corresponding normal vectors. We use the midpoint of the shortest line segment between the two lines given by the points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and their normals $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ to define the center of the sphere $\boldsymbol{c}$. The sphere radius is then approximated by

$$r = \frac{\|\boldsymbol{p}_1 - \boldsymbol{c}\| + \|\boldsymbol{p}_2 - \boldsymbol{c}\|}{2}. \tag{2.7}$$

The sphere is accepted as a shape candidate only if all three points are within a distance of ϵ_d of the sphere and their normals do not deviate by more than ϵ_θ degrees.

In order to generate a cylinder from two points with normals, we first establish the direction of the axis with $\boldsymbol{a} = \boldsymbol{n}_1 \times \boldsymbol{n}_2$. Then we project the two parametric lines $\boldsymbol{p}_1 + t\boldsymbol{n}_1$ and $\boldsymbol{p}_2 + t\boldsymbol{n}_2$ along the axis onto the plane $\boldsymbol{a} \cdot \boldsymbol{x} = 0$. We then take their intersection as the center $\boldsymbol{c}$. The radius of the candidate is the distance between $\boldsymbol{c}$ and $\boldsymbol{p}_1$ in that plane. Again the cylinder is verified by applying the thresholds ϵ_d and ϵ_θ to distance and normal deviation of the samples.

2.3.6.2 *Score Function*

The score function σ_P is responsible for measuring the quality of a given shape candidate. We use the following aspects in our scoring function:

1. To measure the support of a candidate, we use the number of points that fall within an ϵ-band around the shape.
2. To ensure that the points inside the band roughly follow the curvature pattern of the given primitive, we only count those points inside the band whose normals do not deviate from the normal of the shape more than a given angle ϵ_θ.
3. In addition, we incorporate a connectivity measure: Among the points that fulfill the previous two conditions, only those are considered that constitute the largest connected component on the shape.

More formally, given a shape ψ whose fidelity is to be evaluated, σ_P is defined as follows:

$$\sigma_P(\psi) = |P_\psi|. \tag{2.8}$$

That is, we count the number of points in P_ψ. P_ψ is defined in the following two steps:

$$\hat{P_\psi} = \{\boldsymbol{p} | \boldsymbol{p} \in P \wedge |d(\psi, \boldsymbol{p})| < \epsilon_d \wedge \arccos(|\boldsymbol{n}(\boldsymbol{p}) \cdot \boldsymbol{n}(\psi, \boldsymbol{p})|) < \epsilon_\theta\} \tag{2.9}$$

$$\mathcal{P}_\psi = maxcomponent(\psi, \hat{P_\psi}), \tag{2.10}$$

where $d(\psi, \boldsymbol{p})$ is the signed distance of point $\boldsymbol{p}$ to the shape primitive ψ, $\boldsymbol{n}(\boldsymbol{p})$ is the normal in $\boldsymbol{p}$ and $\boldsymbol{n}(\psi, \boldsymbol{p})$ is the normal of ψ in $\boldsymbol{p}$'s projection on ψ. $maxcomponent(\psi, \hat{P_\psi})$ extracts the group of points in $\hat{P_\psi}$ whose projections onto ψ belong to the largest connected component on ψ.

We find connected components in a bitmap in the parameter domain of the shape. A pixel in the bitmap is set if a point is projected into it. Clustering the set pixels in the bitmap yields the largest connected component. For details on the score function and the bitmap used for clustering, we refer to the original paper of Schnabel *et al.* (2007).

2.3.6.3 *Score Evaluation*

Obviously the cost of evaluation would be prohibitive without any optimizations because in a naïve implementation, the distance to all points in P would have to be computed together with a normal at a corresponding position on the shape for each candidate. But since in each run we are only interested in the candidate that achieves the highest score, using the entire point cloud P when computing $\sigma_P(\psi)$ is not necessary for every shape candidate. We significantly reduce the

number of points that have to be considered in the evaluation of $\sigma_P(\psi)$ by splitting the point cloud P into a set of disjoint random subsets: $P = S_1 \bigcup \ldots \bigcup S_r$.

After a shape candidate was generated and successfully verified, the candidate is only scored against the first subset S_1 and no connected component is extracted yet. From the score $\sigma_S(\psi)$ on a subset $S \subset P$ an estimate $\hat{\sigma}_P(\psi)$ for the score $\sigma_P(\psi)$ on all points can be extrapolated using the well known induction from inferential statistics:

$$\hat{\sigma}_P(\psi, S) = -1 - f(-2 - |S|, -2 - |P|, -1 - |S_\psi|) \;, \tag{2.11}$$

$$\text{where } f(N, x, n) = \frac{xn \pm \sqrt{\frac{xn(N-x)(N-n)}{N-1}}}{N} \tag{2.12}$$

is the mean plus/minus the standard deviation of the hypergeometric distribution. $\hat{\sigma}_P(\psi)$ is a confidence interval $[l_\psi, u_\psi]$ that describes a range of likely values for the true score $\sigma_P(\psi)$. The expected value $E(\sigma_P(\psi))$ is given by $\frac{l_\psi + u_\psi}{2}$. With this extrapolation the potentially best candidate ψ_m can be quickly identified by choosing the one with the highest expected value. Since the uncertainty of the estimation is captured in the confidence intervals, the truly maximal candidate can be found by comparing the confidence intervals of the candidates.

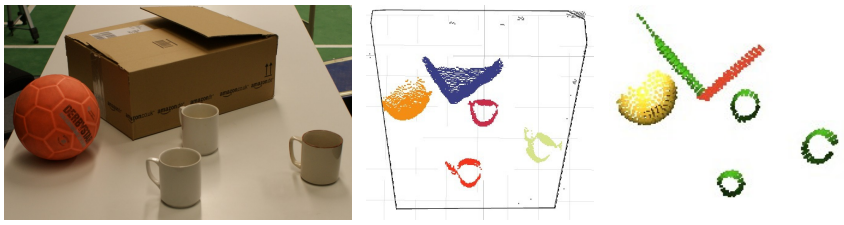
If the confidence intervals of ψ_m and another candidate ψ_i overlap, the score on an additional subset is evaluated for both candidates and new extrapolations are computed, now taking into account the scores on all subsets that have already been computed.

$$\hat{\sigma}_P(\psi) = \hat{\sigma}(\psi, \bigcup_i S_i) \tag{2.13}$$

The more subsets have been considered, the smaller becomes the range of the confidence intervals, since the uncertainty in the estimation decreases. Further subsets are included until the confidence intervals of ψ_i and ψ_m no longer overlap and it can be decided if either ψ_i or ψ_m is better.

To include the effect of the connectedness condition in the extrapolation, every time an additional subset has been evaluated, the maximal connected component is found among all the compatible points that have been discovered so far. The advantage of this priority-based candidate evaluation is that it is less dependent on the random order in which candidates are generated.

We present a typical example of the results of the complete pipeline in Figure 2.8. After determining the parameterized model of the support plane (plane model and contour) and clustering the potential object points, the different primitive shapes are detected. The detected shape primitives also show a shortcoming of our approach: in noisy regions planar surfaces may be better described by a cylinder with a



(a) Scene (b) Clustered objects (c) Detected shapes

Figure 2.8: Typical shape detection results for a scene with planes (red), cylinders (green), and spheres (yellow). Shown are the table scene (a), the extracted object clusters (b), and the detected shape primitives (c). There is one false detection (the left side of the box is detected as a cylinder due to the highly inaccurate data in this region).

large radius. If it can be assumed that cylinders with large radii are not present in the scene, they can easily be ignored thus allowing to correctly detect even noisy planar surfaces.

2.3.7 *Preliminary Experiments and Results*

As a proof-of-concept for this initial segmentation pipeline, we have conducted experiments with the autonomous mobile manipulator *Dynamaid* (Stückler and Behnke, 2011). It consists of an omnidirectional mobile base with four individually steerable differential drives and a size of 60 cm by 42 cm. For the purpose of navigation, it is equipped with two 2D safety laser scanners, one in roughly 20 cm height and one shortly above the ground. It also features an anthropomorphic upper body with two arms and grippers, and a movable head hosting two standard visual cameras in a stereo pair, a directed microphone, and a MESA SwissRanger SR4000 ToF camera. The whole platform is designed to be of low weight and low cost.

For the experiments, the robot was driven in front of a table with different sets of objects. We put a special emphasis on having objects of different shapes on the table, e.g., cylindrical mugs, a spherical ball, and a box with planar walls. For every setting, the robot acquired one point cloud with the ToF camera. Results of applying the object and shape detection pipeline to one of these clouds is shown in Figure 2.9. In all experiments, the robot could reliably detect all objects on the tables and the estimated shape parameters were close to the ground truth parameters of the objects (see Table 2.1). A particular shortcoming of the approach is that noisy planes can be misinterpreted as cylinders or spheres with large radii. Visually inspecting the input point clouds where these false detection happen shows that the sampled surface in fact shows a cylindrical shape rather than a planar surface. Limiting the shape parameter space, e.g., using a maximum

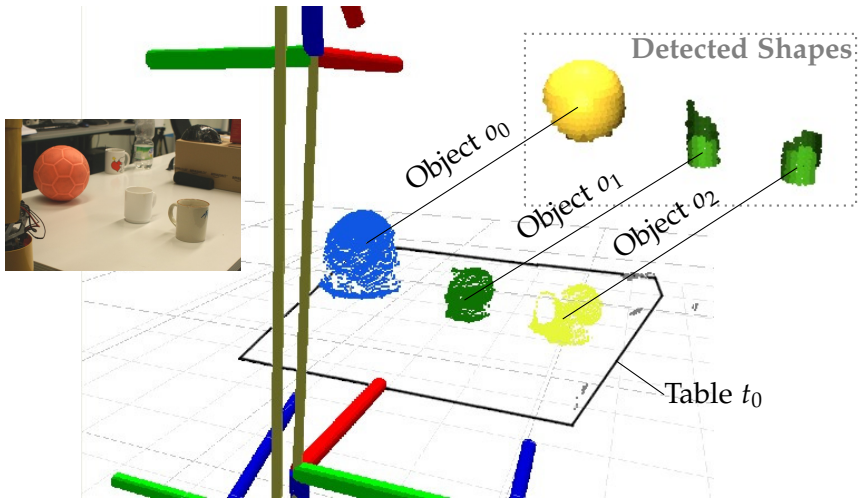


Figure 2.9: Typical result of table, object, and shape detection. After fitting the planar model, the 20 753 inliers are removed and the table is represented solely by the model parameters and the convex hull of the inliers. The remaining 4591 points are segmented into 3 clusters (random colors) and successfully classified as being a sphere (yellow) and two cylinders (green).

Objects	Detected shapes and parameters (units in m)	
Table	Detected primitive: Plane, 20 753 inliers	
	Normal vector:	$[-0.0232 \ -0.0156 \ 0.9996]$
	Distance to **0**:	0.742 795
Object o_0 (cup)	Detected primitive: Cylinder, 758 inliers	
	Centroid:	$[0.6847 \ -0.0166 \ 0.7885]$
	Radius:	0.0314 (ground truth: 0.034)
	Axis direction:	$[0.0869 \ -0.0120 \ 0.9961]$
Object o_1 (ball)	Detected primitive: Sphere, 1738 inliers	
	Centroid:	$[0.6871 \ 0.2058 \ 0.8598]$
	Radius:	0.0778 (ground truth: 0.082)
Object o_2 (cup)	Detected primitive: Cylinder, 815 inliers	
	Centroid:	$[0.6847 \ -0.0166 \ 0.7885]$
	Radius:	0.0284 (ground truth: 0.032)
	Axis direction:	$[0.1028 \ 0.0080 \ 0.9915]$

Table 2.1: Detected shapes and parameters. Although the input point cloud is quite noisy, the object are reliably detected and classified w.r.t. the dominant shape. The estimated shape parameters differ only slightly from ground truth (determined using a measuring tape).

radius for cylinders and spheres, considerably reduces these false detections.

In its current state, the presented approach can process complete point clouds of 25 344 points (i.e., without working on a sub-sampled copy of the cloud) with 2.5 Hz to 5 Hz. Although very simple, the pipeline has been found to be an efficient and robust tool chain for extracting (semantic) information from ToF camera data for mobile manipulation tasks. It comprises techniques for correcting erroneous measurements caused by the ambiguity in distance measurements as well as for filtering points on jump edges. These pre-processing steps are optional and only necessary for ToF cameras that exhibit these special characteristics. That is, for cameras not showing these error characteristics, the pre-processing steps can be skipped to safe computation time.

Furthermore, the pipeline features an approach for detecting primitive shapes in the detected objects. Although the detected shape primitives are particularly useful for object recognition as well as grasp and motion planning (Holz *et al.*, 2014a), they form only one of the many applications of the clustered objects and an exemplary post-processing step in the pipeline. In case, the object clusters are used as masks for further recognition like for instance using the recognition infrastructure REIN (Muja *et al.*, 2011), this post-processing step can also be skipped.

In the remainder of this section, an improved version of the pipeline is presented where we do not use the pre-processing and post-processing steps, but replace some processing steps by faster approximations and focus processing on relevant portions of the data even more. Overall, this improved pipeline allows segmenting complete 640×480 RGB-D point clouds in near real-time.

2.4 FAST RGB-D TABLE TOP SEGMENTATION

Real-time 3D perception of the surrounding environment is a crucial precondition for the reliable and safe application of mobile service robots in domestic environments. Both the perception of the environment and the planning of grasps and motions are computationally expensive and can cause longer interruptions in the workflow of a robot. Color and depth (RGB-D) cameras, such as the Microsoft Kinect™ camera, acquire both visual information (RGB) like regular camera systems and depth information (D) at high frame rates. In terms of measurement accuracy in low ranges (up to a few meters), the acquired depth information does not considerably rank behind the accuracy achieved with 3D laser scanners. However, for applying and making use of these cameras in typical mobile manipulation problems like detecting objects and avoiding collisions, the acquired RGB-D

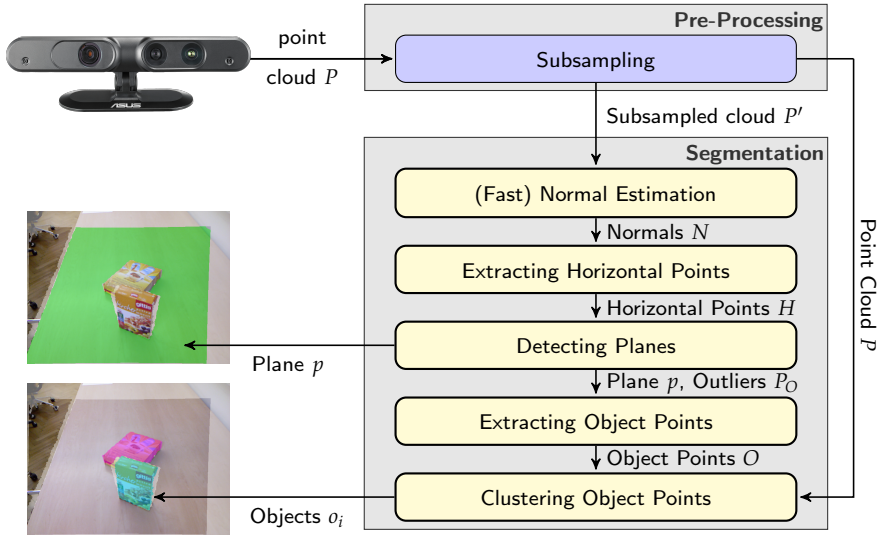


Figure 2.10: Overview of the fast object detection pipeline: input point clouds are first subsampled (160×120 or less) to speed up internal computations. The next difference to the former pipeline is a method for fast approximation of local surface normals and extracting points with horizontal normals. In the extract points, plane are detected and the outliers above the plain are clustered. Both plane and object points are provided in both the subsampled and the original resolution.

camera data needs to be processed in real-time (possibly with limited computing power).

In this section, we present an improved pipeline specifically designed for RGB-D cameras which explicitly makes use of the organized structure of RGB-D camera data. It acquires and processes 3D information at frame rates of up to 30 Hz, and allows for

1. reliably detecting obstacles (e.g., on the ground plane),
2. detecting graspable objects as well as the planes supporting them, and
3. segmenting and classifying all planes in the acquired 3D data.

An important characteristic of the proposed system is that all the above outcomes can be obtained at frame rates of up to 30 Hz.

2.4.1 *Pipeline Overview*

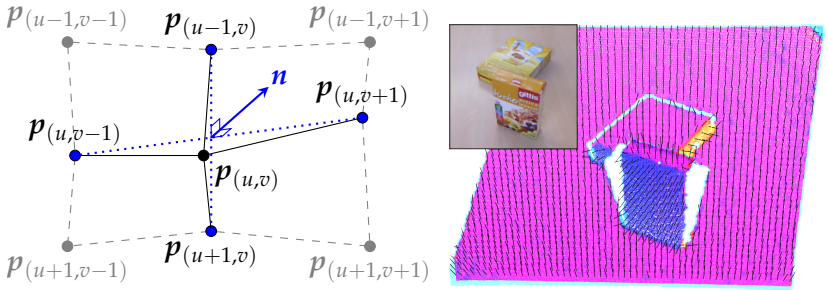
The pipeline allows a mobile robot to reliably detect obstacles and segment graspable objects and supporting surfaces as well as the overall scene geometry. An overview of the extended pipeline is shown in Figure 2.10. In essence, we replace several components of the table top segmentation from Section 2.3 with fast approximations such as using vector cross products and integral images for computing

local surface normals. For segmenting planes, we cluster, segment, and classify points in both normal space and spherical coordinates. For object segmentation, we change the order of components thus focusing computations on relevant portions of the data. Furthermore, we subsample the full resolution input point clouds for many processing steps while still providing the segmented output clusters in full resolutions. Overall, these extensions make it possible to process 640×480 RGB-D point clouds in real-time. The system is tested in different setups in a real household environment. The results show that the system is capable of reliably detecting obstacles at high frame rates, even in case of obstacles that move fast or do not considerably stick out of the ground. We also present a first approach to compute planar segmentations of input point clouds by segmenting all planes in the 3D data for correcting characteristic measurement errors and for reconstructing the original scene geometry in far ranges.

2.4.2 *Fast Computation of Local Surface Normals*

Local geometric features such as surface normal or curvature at a point form a fundamental basis for extracting semantic information from 3D sensor data. A common way for determining the normal to a point $\boldsymbol{p}_i$ on a surface is to approximate the problem by fitting a plane to the point's local neighborhood $\mathcal{P}_i$. This neighborhood is formed either by the k nearest neighbors of $\boldsymbol{p}_i$ or by all points within a radius r from $\boldsymbol{p}_i$. Given the neighborhood $\mathcal{P}_i$, the local surface normal $\boldsymbol{n}_i$ can be estimated by analyzing the eigenvectors of the covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$ of $\mathcal{P}_i$. The eigenvector $\boldsymbol{v}_{i,0}$ corresponding to the smallest eigenvalue $\lambda_{i,0}$ can be used as an estimate of $\boldsymbol{n}_i$ (see Section 2.3.3). The ratio between $\lambda_{i,0}$ and the sum of eigenvalues provides an estimate of the local curvature.

Both k and r highly influence how well the estimated normal represents the local surface at $\boldsymbol{p}_i$. Chosen too large, environmental structures are considerably smoothed so that local extrema such as corners completely vanish. If the neighborhood is too small, the estimated normals are highly affected by the depth measurement noise. A common way to compensate these effects that is also used by Rusu *et al.* (2009b) is to compute the distances of all points in $\mathcal{P}_i$ to the local plane through $\boldsymbol{p}_i$. These distances are then used in a second run to weight the points in $\mathcal{P}_i$ in the covariance computation. By this means, corners and edges are less smoothed and the estimated normals better approximate the local surface structure. However, with or without this second run, estimating the point's local neighborhood is computationally expensive, even when using approximate search in kD-trees which is $O(n \log n)$ for n randomly distributed data points (plus the construction of the tree). Another possibility to compensate for the aforementioned effects is to compute the normals in different neighborhood ranges or



(a) Basic principle (b) Example result

Figure 2.11: Fast normal computation using integral images (a). Two vectors tangential to the surface (dotted blue lines) at the desired position are computed using the blue points. The local surface normal is computed using the cross product of the tangential vectors. A typical result of an acquired point cloud with surface normals is shown in (b).

different scales of the input data and to select the most likely surface normal for each point (Holzer *et al.*, 2012).

Less accurate, but considerably faster than approximate neighborhood search using *kd*-trees is to consider pixel neighborhoods instead of spatial neighborhoods (Holz *et al.*, 2010). That is, the organized structure of the point cloud as acquired by ToF or RGB-D cameras is used instead of searching through the 3D space spanned by the points in the cloud. Compared to a fixed radius r or a fixed number of neighbors k, using a fixed pixel neighborhood has the advantage of having smaller neighborhoods in close range (causing more accurate normals) and larger neighborhoods smoothing the data in far ranges that are more affected by noise and other error sources.

By using a fixed pixel neighborhood and, in addition, neglecting precomputed neighbors outside of some maximum range r as in (Holz *et al.*, 2010), one can avoid the computationally expensive neighbor search, but still needs to compute and analyze the local covariance matrix. Here, we use an approach that directly computes the normal vector over the neighboring pixels in x and y image space.

The basic principle of our approach is to compute two vectors which are tangential to the local surface at the point $\boldsymbol{p}_i$. From these two tangential vectors we can easily compute the normal using the cross product. The simplest approach for computing the normals is to compute them between the left and right neighboring pixel and between the upper and lower neighboring pixel, as illustrated in Figure 2.11.a. However, since we expect noisy data and regions where no depth information is available (a special characteristic of the used cameras), the resulting normals would also be highly affected. For this reason, we apply a smoothing on the tangential vectors by computing the average vectors within a certain neighborhood. To perform this

smoothing efficiently we use integral images. We first create two maps of tangential vectors, one for the x- and one for the y-direction (again in image space). The vectors for these maps are computed between the corresponding 3D points in the point cloud. That is, each element of these images is a 3D vector. For each of the channels (Cartesian x, y, and z) of each of the maps we compute an integral image, which leads to a total number of six integral images. Using these integral images, we can compute the average tangential vectors with only $2 \times 4 \times 3$ memory accesses, independent of the size of the smoothing area. The overall runtime complexity is linear in the number of points for which normals are computed (see the experiments on runtime and accuracy of computing surface normals in Section 2.5.1).

The computation of normals is conducted in the local image coordinate frame ($\hat{\boldsymbol{Z}}$-axis pointing forwards in measurement direction). For further processing, we transform both Cartesian coordinates of the points as well as the local surface normals into the base coordinate frame of the robot (right-handed coordinate frame with the $\hat{\boldsymbol{X}}$-axis pointing in measurement and driving direction and the $\hat{\boldsymbol{Z}}$-axis pointing upwards thereby representing the height of points). In case this transformation is not known, we only apply the corresponding reflection matrix and a translation by 2cm along the $\hat{\boldsymbol{Y}}$-axis that accounts for the difference in position between the regular camera and the infrared camera that senses the emitted pattern for depth reconstruction. It should be noted that this transformation (or the knowledge of the camera's position and orientation in space) is not necessary for the fast plane segmentation in Section 2.4.4, but only for task-specific applications like the extraction of horizontal surfaces.

In addition to the fast normal estimation, we compute spherical coordinates (r, ϕ, θ) of the local surface normals that ease the classification of measured points and the processing steps presented in the following. We define ϕ as the angle between the local surface normal (projected onto the $\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}$-plane) and the $\hat{\boldsymbol{X}}$-axis, r the distance to the origin in normal space, and θ the angle between the normal and the $\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}$-plane. (r, ϕ) is of special interest in obstacle detection, as it represents direction and distance of an obstacle to the robot (in the $\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}$-plane). For plane segmentation, r is abused in our implementation to hold the plane's distance from the origin (in Cartesian space).

2.4.3 *Extracting Horizontal Points*

For finding planes in 3D point clouds common procedures are least squares fitting (if the cloud does not contain plane outliers) and RANSAC-based approaches to detect a plane and its inliers as well as to identify the outliers. Least squares fitting is computationally complex and can only be applied if the data is already pre-segmented to contain only points belonging to the plane to be fit. RANSAC-based

approaches fit planes only to subsets of data and use the rest of the data to determine the amount of points supporting the plane model fit to the subset. Since many iterations are necessary to find the dominant plane in a complete RGB-D point cloud our main idea is to not process points which are not likely to lie on this plane. That is, we focus processing on a subset of points which is likely to constitute the plain we are looking for.

For detecting tables in the vicinity of the robot, we extract those points $\boldsymbol{p}_i$ from the point cloud that have a surface normal $\boldsymbol{n}_i$ that points upwards, i.e., $\boldsymbol{n}_i$ is nearly parallel to the z-axis ($\boldsymbol{n}_i \parallel \hat{\boldsymbol{Z}}$). The extracted points have likely been measured on the surface of a table and form an initial set of table points $T \subseteq P$. In order to distinguish multiple tables, we examine the distribution in the measured heights $\boldsymbol{p}_i^z, \boldsymbol{p}_i \in T$ and split T into multiple sets $T_1, \ldots, T_n$ in case of larger fluctuations. The same is done for larger variations in the position $(\boldsymbol{p}_i^x \ \boldsymbol{p}_i^y)^T$ of the points. The obtained clusters can directly be used to estimate support planes, either by directly fitting planes to the sets or by using a RANSAC-based approach to filter out potential outliers in the set. In our implementation, we first cluster the table point set in Euclidean space and select the largest cluster for RANSAC-based plane fitting.

The segmentation output of the pipeline suits two purposes: obstacle detection during navigation, and object and obstacle detection for manipulation. For navigation purposes, only the ground floor plane (z-component of normal $\boldsymbol{n}_i^z \approx 1$ and distance to origin or height above ground $z \approx 0$) is considered safe. All other points and planes including other horizontal surfaces such as tables are considered as obstacles. For object detection, we limit the search space by the height range in which the robot can manipulate ($0 < z \leq 1.5\,\mathrm{m}$). Clusters of horizontal points outside of these ranges are ignored.

2.4.4 *Fast Plane Segmentation*

In the object detection pipeline only horizontal support planes need to be found. However, the same approximation that is used to find sets of horizontal points can be extended to compute complete planar segmentations of input point clouds, i.e., detecting all larger 3D planes. In order to compute complete planar segmentations, we extend our pipeline to cluster all planar regions based on clustering of the local surface normals. We segment local surface normals in two steps: we 1. cluster (and merge) the points in normal space $(\boldsymbol{n}^x, \boldsymbol{n}^y, \boldsymbol{n}^z)^T$ to obtain clusters of plane candidates and 2. cluster (and merge) planes of similar local surface normal orientation in distance space (distance between plane and origin). After clustering and merging the initial plane segments, we obtain a complete planar segmentation of the input point cloud. Although the clustering-based segmentation is very naïve,

it can already achieve reliable segmentations of larger environmental structures in real-time.

2.4.4.1 *Initial segmentation in normal space*

For the initial clustering step in which we want to find clusters of points with similar local surface normal orientations, we construct a voxel grid either in normal space or using the spherical coordinates. Using the spherical coordinates allows for clustering in the two-dimensional (ϕ, θ)-space, but requires a larger neighborhood in the subsequent processing step in which we merge clusters. Both results and processing times do not differ, however.

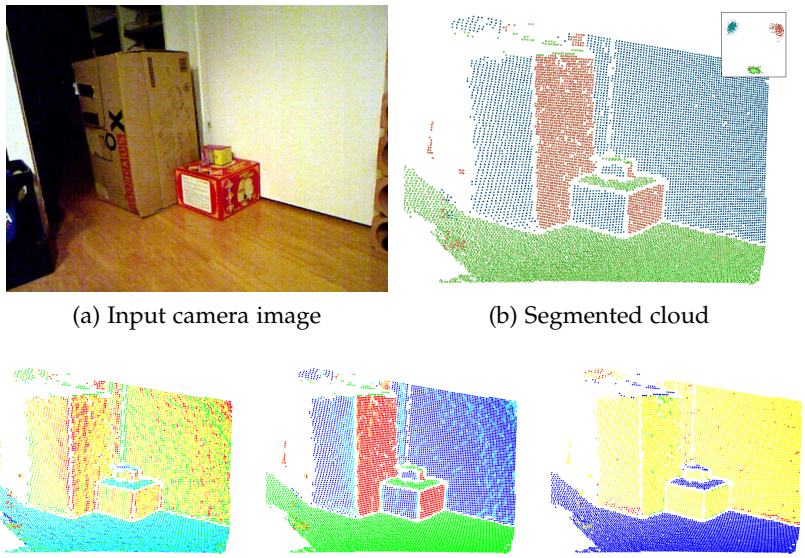
For clustering in normal space, we compute a three-dimensional voxel grid and map local surface normals to the corresponding grid cell w.r.t. the cell's size. Points for which the surface normals fall into the same cell, form the initial cluster and potential set of planes with the same normal orientation. Either all non-empty cells or only those with a minimum number of points are considered as initial clusters.

In order to compensate for the involved discretization effects, we examine the cell's neighbors in the three-dimensional grid structure. If the deviation of the average surface normal orientations in two neighboring grid cells falls below the cluster size (and the desired accuracy), the corresponding clusters are merged. For being able to merge multiple clusters, we keep track of the conducted merges. In case cluster a should be merged with cluster b that was already merged with cluster c, we check if we can merge a and c, or if $a + b$ is a better merge than $b + c$. Although this procedure is less adaptive (and complex) as sophisticated clustering algorithms like *k-Means*, *mean-shift*-clustering or, e.g., ISODATA (Memarsadeghi *et al.*, 2007), this simple approach allows for reliably detecting larger planes in 3D point clouds at high frame rates. In all modes and resolutions of the camera, plane segmentation is only a matter of milliseconds. In contrast to region growing algorithms, we find a single cluster for planes that are not geometrically connected, e.g., parts of the same wall.

An example segmentation is shown in Figure 2.12. Planes with similar (or equal) local surface normal orientations are contained in the same cluster and visualized with the same color.

2.4.4.2 *Segmentation refinement in distance space*

Up to now the found clusters do not represent single planes but sets of planes with similar or equal surface normal orientation. For some applications like extracting all horizontal surfaces, this information can directly be used. For other applications, we split these normal clusters into plane clusters such that each cluster resembles a single plane in the environment.



(a) Input camera image (b) Segmented cloud

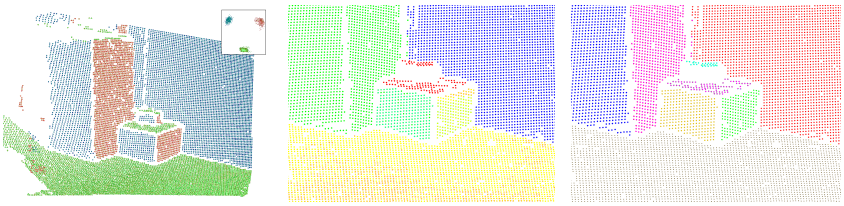
(c) n^x space (d) n^y space (e) n^z space

Figure 2.12: Typical result of the first segmentation step: points with similar surface normal orientation in the input data (a) are merged into clusters (b, shown in both Cartesian and normal space). The components of the normals (n^x, n^y, and n^z) are visualized in (c-e) using a color coding from -1 (red) to 1 (blue). This simple clustering allows for a fast segmentation of planes with similar surface normal orientation., e.g., extracting all horizontal planes.

Under the assumption that all points in a cluster are lying on the same plane, we use the corresponding averaged and normalized surface normal to compute the distance from the origin to the plane through the point under consideration. Naturally these distances differ for points on different parallel planes and we can split clusters in distance space. For compensating the fact that measurements farther away from the sensor are stronger affected by the different error and noise sources, we compute a logarithmic histogram. Again, points whose distances fall into the same bin form initial clusters. These clusters are then refined by examining the neighboring bins just like in the refinement of the normal segmentation. An example of the resulting plane clusters is shown in Figure 2.13. We include this clustering-based complete plane segmentation as "Holz *et al.* (2011)" in the comparative evaluation of plane segmentation algorithms in Section 3.7.3.

2.4.5 *Table Plane and Object Detection*

In order to obtain an efficient representation of tables and to segment individual objects, we extract the horizontal points (Section 2.4.3) and cluster them into planes (Section 2.4.4). The found plane models consisting of the averaged normals and plane-origin distances in



(a) 3 Normal clusters (b) Distance space (c) 8 Plane clusters

Figure 2.13: Typical result of the second segmentation step: Clusters with similar surface normal orientations (a) are clustered in distance space (b). Clusters with similar normal orientations but varying distances (of the respective planes to the origin) are split. For compensating discretization effects, neighboring clusters are again merged to form the segmented planes (c). Color coding in (a+c) is random per cluster, and in distance space from 0.4m (red) to 1.3m (blue) in (b).

each plane cluster are then optimized again using MSAC (Torr and Zisserman, 2000) to filter out residual outliers. Since plane fitting is only conducted in the largest and closest sets of horizontal points the detection is considerably faster than finding dominant planes in the complete point cloud. Moreover, since the majority of points in a cluster (if not all points) belong to the same plane, few iterations are sufficient to find a good plane estimate.

The rest of the pipeline does not considerably differ from the one presented in Section 2.3. We project the points of a cluster onto the found plane and compute the convex hull. These steps are repeated for all horizontal planes that have been found in the given height range. For all points from non-horizontal plane clusters, we then check if they lie above a supporting plane (within a range of e.g. 30 cm) and within the corresponding convex hull (again with a tolerance of a few centimeters). Points meeting both requirements are then clustered to obtain object candidates. For each of the candidates we compute the centroid and the oriented bounding box in order to distinguish graspable from non-graspable objects. Here we simply assume that the minimum side length of graspable objects needs to lie between 1 cm and 10 cm. Furthermore, we neglect clusters where the number of contained points falls below a threshold (e.g. 50 points).

Here, the main difference to the pipeline in Section 2.3 lies in the fact that so far all processing steps have used a subsampled version of the original point cloud. For a full-resolution segmentation output, we project both the object points and the table points back into the original full resolution image. If previously not processed points are directly connected to an object or a table point they get the same label if and only if they support the same plane model and fit into the same object cluster, respectively.

Figure 2.14: Examples of detected tables and objects in the OSD dataset by Richtsfeld *et al.* (2012). Shown are the original RGB input images with a color overlay encoding pixels with no depth information (white), table plane (green), and objects (randomly colored per object). Despite objects being clustered together (bottom right), all objects are reliably detected and clustered.

An example segmentation of a table and the objects thereon can be seen in Figure 2.14. The object clustering assumes well separated parts and, consequently, objects touching each other may get clustered together. In order to reliably cluster touching objects in cluttered scenes, more sophisticated (but also computationally more demanding) approaches need to be used such as the works of Marton *et al.* (2014) and Richtsfeld *et al.* (2014). However, since our pipeline is only designed for efficiently segmenting the input scene into horizontal support surfaces and potential object points we do not further separate such multi-object clusters. That is, reliably detecting points belonging to objects is sufficient in our case as the segmentation output is used as a set of regions of interest for further processing.

2.5 EXPERIMENTS AND RESULTS

In order to evaluate the performance of the presented fast object detection pipeline, we have conducted a series of experiments focusing on the accuracy of approximated surface normals, the object detection accuracy, and the runtimes of both the overall pipeline and the involved components.

2.5.1 *Accuracy and Runtime for Computing Normals*

Both the planar segmentation and the detection of graspable objects and obstacles highly depend on the quality of the estimated surface normals. In order to evaluate the accuracy of the estimated normals, we conducted a sequence of experiments comparing the estimated

	Runtime	Normal deviation	
Resolution	mean ($\pm$ std)	mean ($\pm$ std)	considerable
640×480 (VGA)	36.73 ± 6.12 ms	4.15 ± 2.31 deg	roughly 2%
320×240 (QVGA)	11.08 ± 2.29 ms	5.23 ± 2.35 deg	roughly 1.2%
160×120 (QQVGA)	4.32 ± 0.58 ms	3.76 ± 1.84 deg	roughly 1%

Table 2.2: Normal estimation: runtime and accuracy.

surface normal at each point with the one computed over the real neighbors using the two-run RANSAC and PCA approach as described earlier (Rusu, 2009). For the neighbor search a radius r has been chosen that linearly depends on the measured range to the point under consideration, i.e., a smaller radius for the more accurate close range measurements and a larger radius for measurements being farther away from the sensor. This radius function has been manually adapted for each of the point clouds used in the experiments in order to guarantee correct (and ground truth-like) normals.

The presented results have been measured over 40 points clouds taken in 4 different scenes in a real house-hold environment: 1) a table top scene with only one object, 2) a cluttered table top scene with > 20 objects, 3) a room with a cluttered table top and distant walls, 4) a longer corridor with several cabinets where measurements of up to 6m have been taken. In average, a deviation of roughly 5° has been measured (see Table 2.2). This is primarily caused by the fact that the current implementation does not specifically handle edges and corners as is done with the second PCA run on the weighted covariance. Furthermore, using nearest neighbor search better compensates for missing measurements in regions where no depth information is available. However, especially in close range (e.g. up to 2 m), the estimated normals are quite accurate and do not deviate from the *true* local surface normals. Only one to two percent of the estimated normals considerably deviated from the true normals (deviations larger than 25°).

2.5.2 *Object Detection Accuracy*

In order to assess the detection accuracy of the object detection pipeline, we have used the object segmentation database (OSD)[2] by Richtsfeld *et al.* (2012). It contains a total of 66 RGB-D point clouds with 640×480 points and per-pixel labeling. The labels are 0 for pixels not belonging to any object and positive (label > 0) for pixels belonging to objects. In the latter case, the label encodes the object scene id—all objects in the scene are numbered starting with id 1. The dataset con-

2 OSD is available at: `http://users.acin.tuwien.ac.at/arichtsfeld/?site=4`

	Objects		Object Pixels	
Scene	Detected*	Rate	Detected	Rate
Boxes	36 / 36	100%	449 786 / 438 101	97.4%
Stacked Boxes	21 / 21	100%	233 882 / 228 926	97.8%
Occluded	14 / 14	100%	130 135 / 121 520	93.3%
Cylindric	42 / 42	100%	339 014 / 320 576	94.5%
Mixed	81 / 81	100%	628 524 / 608 185	96.7%
Complex	166 / 166	100%	1 138 116 / 1 064 854	93.5%
Total	360 / 360	100%	2 919 457 / 2 782 162	95.2%

*Objects are detected if more than 90% of its pixel are detected as object pixels.

Table 2.3: Detection results for OSD v0.2, the semantic object detection dataset (Richtsfeld *et al.*, 2012).

tains six different scene types: scenes with boxes, stacked boxes, boxes occluding each other, cylindrical objects, mixed scenes with boxes and cylindrical objects, and complex scenes where objects are both stacked and occluded. For measuring the detection accuracy, we use two metrics: the success rate of detecting object pixels and the success rate for detecting objects. We consider an object pixel as detected if a pixel with a positive label in the ground truth data is detected as belonging to an object by the pipeline. If more 90% of the pixels of an object in the ground truth labeling are detected by the pipeline, we consider that object as being detected. We report the detailed results in Table 2.3.

False positives are not shown in the table since the pipeline did not incorrectly label any object points. Only few detected object points were not contained in the ground truth labeling but were found to lie on the objects by visual inspection of the segmentation results. These differences are caused by inaccuracies in the ground truth labeling. Furthermore, the ground truth labeling incorrectly classifies some points on the support surface occluded by an object as belonging to the object. Consequently, these are (correctly) missed by the pipeline. Both types of inaccuracies in the ground truth labeling have been the motivation for using a 90% pixel overlap as the threshold for detecting an object.

Overall, the object detection pipeline correctly found all 360 objects in the dataset and correctly labeled more than 95% of the object pixels. Missed object points were either caused by inaccuracies in the ground truth labeling or at object boundaries coinciding with the support surface. In the latter case, the points are labeled as belonging to the support surface and are not included in the object clustering. Typical examples of the different scene types and the segmentation results of

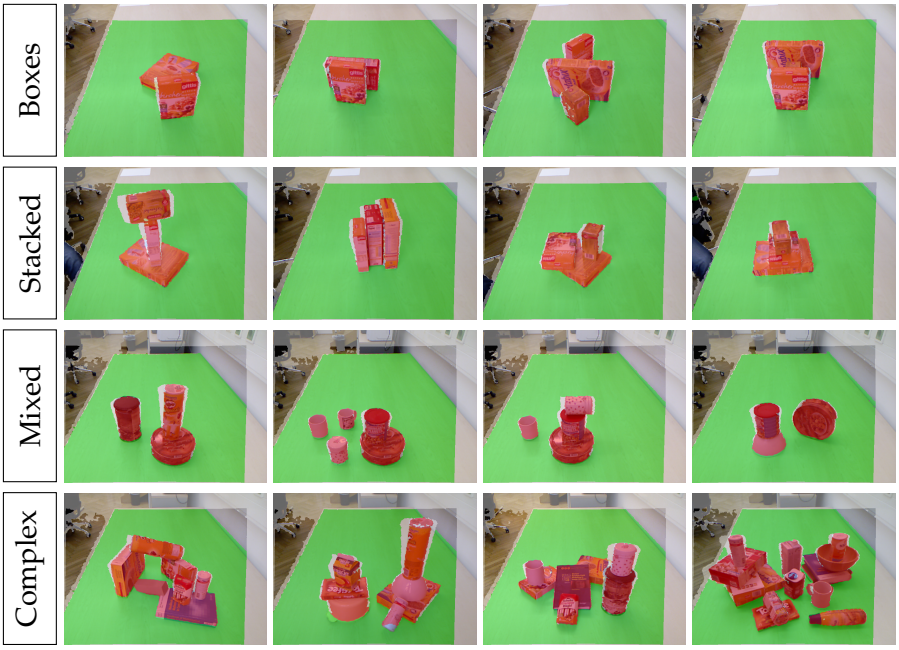


Figure 2.15: Example segmentations for OSD v0.2, the semantic object detection dataset (Richtsfeld *et al.*, 2012). The detection results are shown as semi-transparent overlays over the original RGB images with invalid depth measurements (white), support surface (green), and object candidates (red). Despite some minor deviations at object boundaries, all pixels belonging to an object in the ground truth data are correctly detected as object candidate points.

our pipeline are shown in Figure 2.15. In all 66 point clouds of the data both the support plane and the objects thereon are accurately found.

2.5.3 *Runtime Evaluation*

In order to obtain runtime estimates for the complete object detection pipeline and the involved components, we have recorded RGB-D point clouds in ten different settings. The scenes range from an empty room over close empty tables to cluttered table scenes (up to 10 objects per scene) in different distances from the camera. For each scene, 10 000 range images have been acquired and segmented. We measured both the overall execution time of the pipeline to segment one point cloud and the runtimes of the individual processing steps such as computing surface normals, detecting the support plane, computing the convex hull and clustering the points over the support plane. In all cases, the original 640×480 point clouds were subsampled to 160×120 in the pipeline while the final object point labeling and clustering was done in the original resolution in order to obtain full-resolution masks for further processing steps. Figure 2.16 shows the measured execution

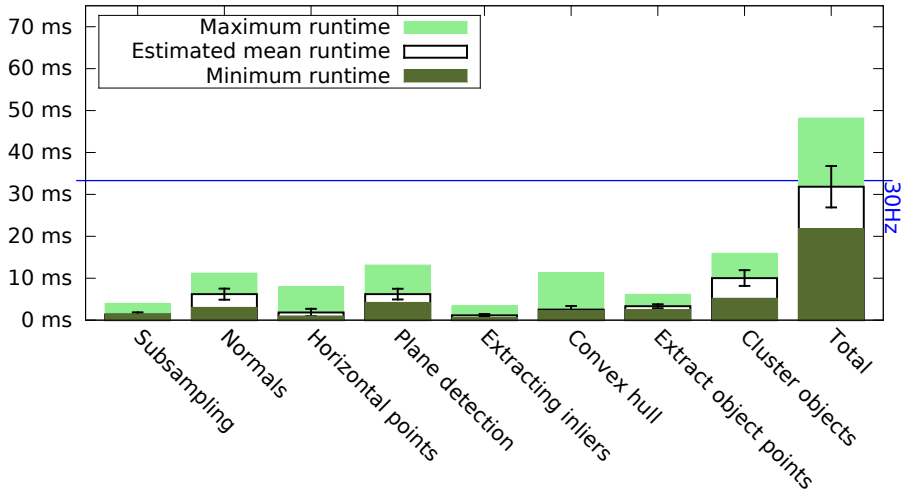


Figure 2.16: Execution times for object detection in ten different scenes. Shown are the average execution time as well as the minimum and maximum execution time. Despite few outliers, the overall execution time stays under 33.3 ms, i.e., segments input point clouds at 30 Hz.

times of the pipeline and its components. On average, the overall execution time stays under 33.3 ms, i.e., segments input point clouds at 30 Hz. Naturally, the maximum execution time shows few outliers. However, the approach never needed more than 50 ms to obtain a segmentation into background, support surface, and objects.

The experiments have been carried out on an Intel(R) Core(TM) i7-3740QM Central Processing Unit (CPU) at 2.70 GHz without parallelization, i.e., all processing was conducted in a single thread on a single CPU. Using the integral image approach, normals can be estimated rapidly for the 19 200 pixels in the subsampled image within approximately 6.5 ms on average. The extraction of horizontal points takes on average 1 ms. Limiting the search space for the support plane does not consume significant runtime (i.e., $\ll$ 1 ms). A more costly step is the application of RANSAC to find the support plane. It amounts to about 6 ms on average. Extracting the points in the support plane, constructing the convex hull of the plane, and extracting the points on objects above the support polygon again require low runtimes of about 2 ms each. The clustering of the points into objects takes about 10 ms. The computation time in this step depends on the number of objects in the scene and the number of points measured on the surface of the objects. The measured runtimes demonstrate that our approach is very performant and that it yields robust detection results in short computation times.

2.6 APPLICATION TO FAST OBJECT PERCEPTION AND GRASPING

Mobile manipulation tasks require a wide variety of perception and action capabilities ranging from navigation over object detection and recognition to planning and executing grasps and arm motions. In addition, robots are expected to not just to achieve an assigned task, but also to perform it in a reasonable amount of time. While much research has been invested into the general solution of complex perception and motion planning problems, little work has been focused on methods that solve such tasks efficiently in order to allow for continuous task execution without interruptions. In a joint work with Ricarda Steffens and Jörg Stückler, the proposed real-time object detection pipeline was integrated with efficient object tracking and grasp planning into a complete system for (near real-time) object perception and grasping of previously unknown parts (Stückler *et al.*, 2013b). The resulting system is successfully applied on the mobile service robots *Dynamaid* (Stückler and Behnke, 2011) and *Cosero* (Stückler *et al.*, 2013a).

2.6.1 *Introduction*

The context of this work is the RoboCup@Home league—an annual international competition for mobile service robots interacting with human users (Holz *et al.*, 2014b; Iocchi *et al.*, 2015). The league addresses service robot applications and focuses on navigation (and SLAM) in dynamic environments, mobile manipulation and human-robot-interaction. A typical task in the competition is the manipulation of objects like, for instance, retrieving objects for a human user. Such tasks pose two specific challenges: 1. the robot needs to accomplish the task quickly and without interruptions (there is a maximum amount of time for each task), and 2. the environment is both cluttered and highly dynamic, i.e., humans and other robots may be present in the robot's workspace, and objects (including furniture) may move since the robot has last observed them.

To cope with the dynamics, we follow a coarse-to-fine strategy for aligning the robot to the objects involved in the task. When told to retrieve an object from a table, for example, the robot first approaches the last known position of the table within the reference frame of a static map. To compensate for deviations in the position, the robot then adjusts in distance (in height) to the table. Both robots feature an anthropomorphic upper body which can be lifted and rotated using two additional degrees of freedom. Finally, it aligns itself to bring the object into the workspace of its arms.

In order to achieve fast performance without (longer) interruptions, we integrate the real-time object perception pipeline with efficient grasp planning and motion control. The object segmentation pipeline processes depth images in real-time. From the extracted object point

(a) Example table scene (b) Colored point cloud (c) Detected objects

Figure 2.17: Real-time segmentation results for an example setting (a) with the raw point cloud from the Microsoft Kinect™ camera (b) and the detected table and objects (c).

clusters, the efficient grasp planning method derives feasible, collision-free grasps. We consider two types of grasps on the objects: from the side and from above, depending on the orientation of the computed principal axes. For example, a longer object lying on the table is grasped from the top while it is grasped from the side if it is standing upright.

2.6.2 *Perception and Grasping Pipeline*

The perception and grasping pipeline consists of four processing steps: 1. segmentation of the support surface of the table or shelf and the objects thereon, 2. tracking the detected objects over several frames to get better estimates of their positions and dimensions, 3. computing feasible and collision-free grasps on the detected object to grasp, and 4. executing the grasping motion.

2.6.2.1 *Real-Time Object Segmentation*

The segmentation of the objects and the support surface does not deviate from the pipeline presented in Section 2.4. For all points, we compute local surface normals using the fast approximation of Section 2.4.2. Referring to Figure 2.17, we then extract all points with vertical normals (normals pointing upwards along the $\hat{\boldsymbol{Z}}$-axis). Using the fast clustering-based segmentation approach (Section 2.4.4) and MSAC (Torr and Zisserman, 2000), we detect the support surface and compute the planar model. For all points that do not belong to the most dominant horizontal support plane, we extract those that are above the plane and whose projections lie within the plane's convex hull. The extracted points are then clustered to obtain individual sets of points and object candidates, respectively. In case of grasping objects in shelves, we slightly shrink the convex hull in order to neglect points at a side or back wall of the shelf.

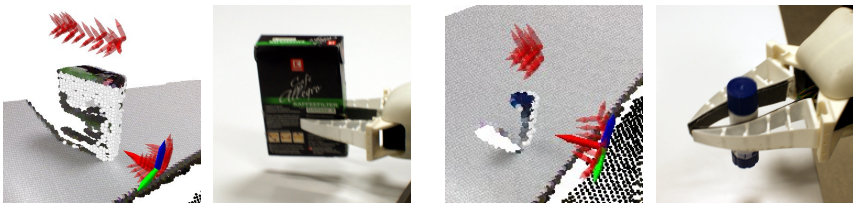
2.6.2.2 *Tracking Detected Objects*

RGB-D cameras a prone to various error and noise characteristics. Depending on the objects on the table, e.g., their visibility and reflection properties, only parts of the object may be visible in an image. Consequently, the object position (centroid of the segmented cluster) and its dimensions (extents along the principal axes) may be inaccurate when computed in a single frame. In order to improve position and dimension estimates over several frames, we track the segmented the object clusters using a multi-hypotheses object tracker. For each hypothesis, we estimate 3D position and velocity in the reference frame of the mobile base through Kalman Filters (KFs). In the KF prediction step, we use odometry information to compensate for the motion of the robot. The tracks are corrected with the observations of their 3D position and extents. In order to associate the object detections in a new frame uniquely with existing hypotheses, we use the Hungarian method (Kuhn, 1955). Especially for objects with diffuse reflection properties and smaller objects not considerably sticking out of the support plane, tracking the estimated parameters considerably improves the accuracy of the computed parameters.

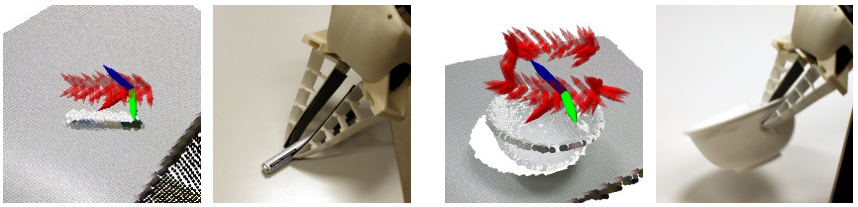
2.6.2.3 *Grasp Planning*

We distinguish two kinds of grasps for which we apply parameterizable motion primitives. Side-grasps are designed to approach the object along its vertical axis by keeping the parallel grippers aligned horizontally. To grasp objects from the top, we pitch the end-effector by 45° downwards to grasp objects with the finger tips. In addition we distinguish two poses in the grasping process: the grasping pose in which the object is inside the gripper and a *pre-grasp* pose above the object from which the grasping pose can be reached by following a straight line in Cartesian space.

The grasp planner generates candidate grasping poses (and pre-grasp poses) along the principal axes of the object for both grasping from the side and grasping from the top. Since the initial sampling does not consider any feasibility constraints or collisions, we then filter the grasp candidates taking into account four different criteria: we reject grasps 1. if the object's width orthogonal to the grasp direction does not fit into the gripper, 2. if the object is too low (for side-grasps), 3. if the grasp is outside the reachable workspace, and 4. if there are collisions during the motions to the pre-grasp pose and the grasping pose. The collision-free and feasible grasps are then ranked, again, using different criteria such as preferring a smaller distance to the object, a smaller angle between the line towards the shoulder and the grasping direction, and a smaller distance to the shoulder. After ranking, we extract the best top- and side-grasps and select the most appropriate one depending on the object and its largest extent in the



(a) Examples for sides-grasps: grasping a box (left) and a glue stick (right)



(b) Examples for top-grasps: grasping a pen (left) and a bowl (right)

Figure 2.18: Sampled and selected pre-grasp poses for both types of grasps. The visualizations show the segmented object, the sampled pre-grasp poses (red), and the select grasp (red-green-blue coordinate frame). The photos show the end-effector while grasping the object. For more examples, we refer to (Stückler *et al.*, 2013b).

	Side-grasp		Top-grasp		
	Table	Shelf	Table	Shelf	Runtime
Object detection	41/41	37/37	55/55	49/49	≈33 ms
Grasp planning	41/41	37/37	50/55	47/49	≈100 ms

Table 2.4: Detection and grasping results. In all experiments, side-grasps have been successful whereas some top-grasps on more complex objects failed. The pipeline allows for fast object detection and grasp planning (detection at 30 Hz, planning at 10 Hz).

horizontal plane. Examples of the sampled and selected grasp poses for different objects and grasps can be seen in Figure 2.18.

2.6.3 *Experiments and Results*

In order to evaluate the robustness and efficiency of the approach, we have conducted different series of experiments with different objects in different orientations and positions. In the first series, the objects were put on a table, in the second series they have been put into a shelf. Referring to the detailed results in Table 2.4, the objects have been well detected both on the table and in the shelf. The majority of the planned grasps was successfully executed and the part was grasped: 100% for side-grasps, 91% and 96% for top-grasps. Only few top-grasps failed especially for more complex objects like the small pen or the bowl (see Figure 2.18).

Both the detection of objects and the planning of grasps are particularly efficient. Objects are detected and tracked with the frame rate of the camera, i.e., 30 Hz. Collision-free and feasible grasps for the detected objects are obtained within 100 ms. In conclusion, the integrated pipeline allows for grasping previously unseen objects without prior knowledge and is efficient enough to avoid interruptions in the robot's workflow. After reaching a table and looking down to the object, it is a matter of not more than a single second until the robot starts grasping the part. This approach has also been successfully demonstrated at RoboCup@Home competitions.

2.7 APPLICATION TO MOBILE ROBOT DEPALLETIZING

In a joint work with Angeliki Topalidou-Kyniazopoulou and Jörg Stückler, another object perception and grasping pipeline was developed that, again, makes use of the proposed object detection pipeline (Holz *et al.*, 2015b; Holz *et al.*, 2015a). We present this work here as a second application for the approach.

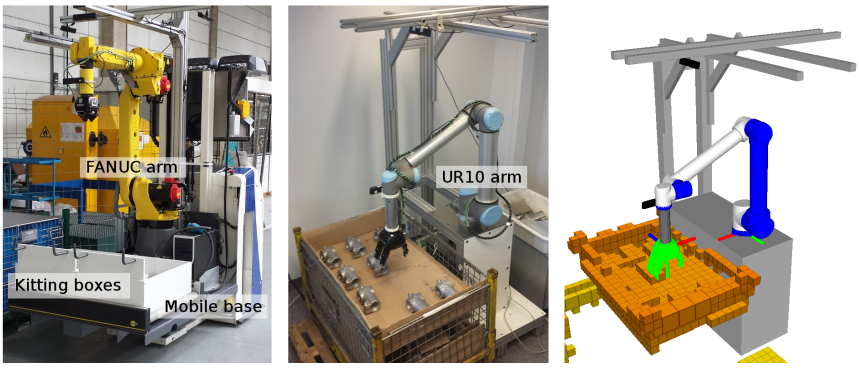
2.7.1 *Introduction*

Another application where the developed segmentation pipeline was used is mobile robot depalletizing, i.e., picking parts from pallets. Picking parts from pallets is a fundamental task in so-called *kitting type distribution*. Kitting became popular in the automotive industry due to a paradigm shift from mass production to increased customization of products (build-to-order). More customized products with increased assembly combinations implicitly means more components to store, transport and feed to the production line. Most often, human operators called *pickers* collect parts as needed from the respective containers, i.e., bins and pallets, and place them in kitting boxes with several compartments. Once complete, the kits are delivered to the production line and synchronized with the cars being produced. In the course of a larger project on kitting using mobile manipulators (Figure 2.19), we have developed a system for automated grasping of parts from pallets.

2.7.2 *Object Perception Pipeline*

Referring to Figure 2.20, the object perception and grasping pipeline comprises the following steps.

1. Using the respective workspace camera (to the left or right side of the robot), we detect the pallet and object candidates using the real-time object detection pipeline. If no object is found (e.g., when the pallet is cleared) the robot stops and reports to the operator.



(a) Final demonstrator (b) Lab setup (c) Simulation

Figure 2.19: Robot platforms and pallets: shown are the demonstrator at the industrial end-user site (a), the lab setup used for development (b) and a visualization of the lab setup (c). The platforms are equipped, respectively, with a Universal Robots UR10 and a FANUC M-20iA/35M arm, a Robotiq 3-finger gripper, and four RGB-D cameras for perceiving the workspace and objects in front of the gripper.

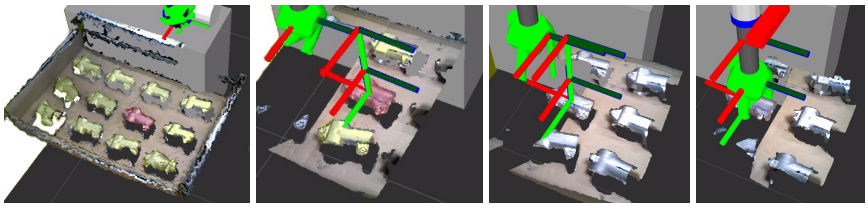


Figure 2.20: Object detection and localization. From left to right: workspace camera point cloud with extracted object candidates (yellow) and selected object (red), and wrist camera point clouds during localization, approach and grasping.

2. The wrist camera is positioned on top of the object candidate being closest to the pallet center.

3. Using the wrist camera, we recognize and localize the part, again after an initial segmentation using the real-time object detection pipeline. The quality of the found matching is used for object verification. Poor matching quality indicates that a wrong object was found. In case of a wrong object, the robot stops, reports the error and waits for an operator instruction to continue its operation.

4. A grasp is selected from a set of predefined grasps and the robot plans a motion to reach it.

5. The robot grasps the object and plans a motion to an intermediate pose for the subsequent placement in the kitting box.

2.7.2.1 *Initial Part Detection*

The task of picking an object from the pallet starts, respectively, when navigation has already taken place and the robot is positioned in the vicinity of the pallet. In order to compensate for potential misalignments or inaccuracies in the estimated poses of robot and pallet, we first use the workspace camera to find the pallet and to get a first estimate of where to find potential object candidates. Assuming that we know on which side of the robot the pallet is located, we acquire images of the corresponding workspace camera and search for horizontal support surfaces above the ground plane. In order to achieve real-time performance, we use the efficient object detection pipeline presented in this chapter.

Referring to Figure 2.20, we restrict the extracted (horizontal) planes to lie in the region where we expect to find the pallet, e.g., not outside the robot's reachable workspace, and neglect others such as the ground plane. In order to find potential object candidates, we then select the most dominant support plane, compute both convex hull and minimum area bounding box, and select all RGB-D measurements lying within these polygons and above the extracted support plane. We slightly shrink the limiting polygons in order to neglect measurements caused by the exterior walls of the pallet. The selected points are clustered (to obtain object candidates), and the cluster being closest to the center of the pallet is selected to be approached first.

After approaching the selected object candidate with the end effector, the same procedure is repeated with the wrist camera in order to separate potential objects from the support surface. Using the centroid of the extracted cluster as well as the main axes (as derived from principal component analysis), we obtain a rough initial guess of the object pose. With the subsequent registration stage, it does not matter when objects are not well segmented (connected in a single cluster) or when the initial pose estimate is inaccurate.

2.7.2.2 *Object Pose Refinement*

The initial part detection only provides a rough estimate of the position of the object candidate. In order to accurately determine both position and orientation of the part, we apply a dense registration of the extracted object cluster against a pre-trained model of the part. We use muli-resolution surfel maps (MRSMAPs) as a concise dense representation of the RGB-D measurements on an object (Stückler and Behnke, 2014). In a training phase, we collect one to several views on the object whose view poses can be optimized using pose graph optimization techniques. Our pose refinement approach is closely related to the soft-assignment surfel registration approach of (Droeschel *et al.*, 2014b) for registering sparse 3D point clouds.

Instead of considering each point individually, we map the RGB-D image acquired by the wrist camera into an MRSMAP and match surfels. This needs several orders of magnitudes less map elements for registration. Optimization of the surfel matches (and the underlying joint data-likelihood) yields the rigid 6 degree-of-freedom (DoF) transformation from scene to model, i.e., the pose of the object in the coordinate frame of the camera. The actual optimization is done through expectation-maximization (Bishop, 2006).

2.7.2.3 *Object Verification*

After pose refinement, we verify that the observed segment fits to the object model for the estimated pose. We can thus find wrong registration results, e.g., if the observed object and the known object model do not match or if a wrong object has been placed on the pallet. In such cases the robot stops immediately and reports to the operator (a special requirement of the end-user).

For the actual verification, we establish surfel associations between segment and object model map, and determine the observation likelihood similar as in the object pose refinement. In addition to the surfel observation likelihood, we also consider occlusions by model surfels of the observed RGB-D image as highly unlikely. Such occlusions can be efficiently determined by projecting model surfels into the RGB-D image given the estimated alignment pose and determining the difference in depth at the projected pixel position. The resulting segment observation likelihood is compared with a baseline likelihood of observing the model MRSMAP by itself. We determine a detection confidence from the re-scaled ratio of both log likelihoods thresholded between 0 and 1.

2.7.2.4 *Grasping Found Parts*

In case, no part was found or if the found part could not be successfully verified, the robot moves back to its initial pose and reports to the operator. In case an object was found, we select a collision-free grasp from a set of predefined grasps. Grasp planning is not desired in this domain, since many of the parts are fragile and there are only few allowed grasps to manipulate them. We then plan a collision-free motion to reach the grasping pose, pick up the part, and move the part to the kitting box. In order to save motion planning time, we split the necessary motions into chunks and pre-compute as many of them as possible. For this purpose, we use grids of poses to the left and right side of the robot and, for every pose in the grid, pre-compute the shortest path to and from the initial pose of the arm.

2.7.3 *Experiments and Results*

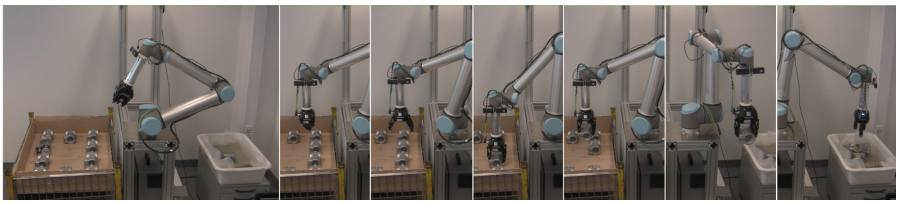
In order to assess robustness and performance of our approach, we conducted a series of experiments of both individual components and the integrated platform. As evaluation criteria, we focus on the success rates and the execution times of the individual components and the overall cycle times of the integrated system for picking an object from the pallet.

The purpose of the integrated depalletizing test is to test the complete object perception and grasping pipeline. In the experiment, the pallet is equipped with a total of 12 objects: 10 being the right part to pick, and two wrong objects (a very similar one and a very different one). A typical experiment setup is depicted in Figure 2.21a. Instead of waiting for a particular pick order, the robot is repeatedly asked to pick an object from the pallet. The robot is expected to clear the pallet by grasping all (correct) objects, stop and report to the operator when it has found a wrong object, and to report when the pallet is empty. In case of failure (wrong object, empty pallet, etc.), the robot waits for commands by an operator, e.g., to continue operation. The latter is a special requirement by the industrial partner. It is intended that robots and human workers not only share a workspace but work hand-in-hand.

In all ten runs, the pallet was cleared without major failures. Only for a single out of the 100 grasps, a grasp failure occurred: during approach, the robot avoided a phantom obstacle, collided with the object and failed grasping. The robot stopped operation due to the detected collision and reported the error. After inspection of the scene, the operator commanded the robot to continue. When the same object was approached again later, it was successfully grasped. In another case, the robot stopped execution due to a phantom object: when approaching and grasping the last object on the pallet, a phantom obstacle (erroneous measurements caused by the surrounding packaging) appeared right on top of the object. The problem was reported to the operator who commanded the robot to continue after inspecting the scene. The robot then successfully continued grasping the object. Both failures were caused by incorrectly updating the obstacle map used for motion planning. The problem was resolved and did not occur in later runs.

Regarding object detection, localization and verification, no errors occurred. The robot correctly localized all objects, correctly identified wrong objects, and correctly detected that the pallet had been cleared in 100 % of the cases. None of the components had false positives (or false negatives).

As for the execution times, the initial object (candidate) detection and pallet detection runs roughly with the framerate of the workspace camera (30 Hz). Object localization and verification using the wrist



(a) Photo sequence of one run (detecting, localizing, grasping and releasing the part)

	Execution Times	Success Rate	
Component	Mean	Successful / Total	
Initial detection	26.3 ms ± 10.3 ms	120 / 120	(100 %)
Detecting empty pallets		10 / 10	(100 %)
Localization & verification	532.7 ms ± 98.2 ms	100 / 100	(100 %)
Identifying wrong objects		20 / 20	(100 %)
Grasping a found object	7.80 s ± 0.56 s	99 / 100	(99 %)
Detection and grasping	13.84 s ± 1.89 s	99 / 100	(99 %)

(b) Execution times and success rates per component (measured over 10 complete runs). Cycle times include releasing and moving to initial pose.

Figure 2.21: Depalletizing experiments. In a total of 10 runs, the robot clears a pallet containing 10 correct objects (a). It correctly detects the objects on the pallet and detects, localizes, and verifies parts with high success rates and low execution times (b). Only a single grasp fails due to a collision. The robot reports the error and successfully grasps the object after being commanded to continue operation. Overall, we achieve cycle times for detecting and grasping objects of approximately 13 s (c). Note that we focus on object perception and neglect further optimization of motion execution.

camera takes roughly 0.5 s. Overall, none of the object perception components considerably interrupts the operation of the robot and increases cycle time. That is, almost 100 % of the reported cycle times is spent on motion planning and execution.

In recent and ongoing work, the pipeline has been integrated into a skill-based robot control architecture (Holz *et al.*, 2015a) and was successfully tested on the final demonstrator platform at the industrial end-user site (Holz *et al.*, 2015b). We refer to these papers for more details and more experimental results.

2.8 CONCLUSION

In this chapter, we have presented a complete pipeline for segmenting organized point clouds into horizontal support surfaces and objects thereon. For ToF cameras, two optional pre-processing steps are presented that unwrap distance measurements outside the measurement

interval and filter out spurious measurements between occluding edges and occluded surfaces. Using the filtered data a first pipeline was presented that detects dominant support planes, extracts points above these planes, and clusters the extracted points as potential object candidates for further processing. Examining the position, the principal axes and the dimensions of the object clusters allows for determining whether or not an object is graspable by a robot, i.e., if the object is within the reachable workspace, if the object fits into the gripper, and if the object can be reached using a collision-free motion. As one possible post-processing step, we detected geometric shape primitives such as planes, cylinders and spheres. Such shape primitives are particularly useful for tasks like grasp planning.

In order to speed up processing and considerably reduce the runtime of the approach, we have extended the pipeline by replacing several components by fast approximations. Instead of computing local covariance matrices and normals, we use an efficient approach for approximating local tangential vectors in integral images and computing the normals using the cross product of the tangential vectors. Experiments have shown that the approximated normals are accurate and only slightly deviate from normals obtained using a state-of-the-art computation in two runs. Instead of trying to find dominant planes in input point clouds, we segment the point cloud into planes by clustering points in both normal and distance spaces. In order to further speed up processing, we limit the search space for the planar segmentation using only points with normals pointing upwards and lying in a given height range. Furthermore, internal processing steps use subsampled point clouds where possible. The resulting pipeline can reliably detect horizontal support surfaces and objects thereon at 30 Hz for 640×480 RGB-D point clouds. If objects are well separated on the support plane (e.g., having an inter-object distance of at least 2 cm), they are also reliably clustered into individual objects.

Not restricting the planar segmentation to horizontal planes allows for segmenting complete point clouds and extracting all planes. Even though individual points may be missed in the detected planes (e.g., due to noise in the coordinates and/or the normal), this approach achieves reliable segmentations of all larger planes in the tested scenes. We include this approach as a baseline in the experimental evaluation of Section 3.7.3.

Finally, we have presented two applications of the real-time object detection: a particularly fast pipeline for perceiving and grasping objects without a priori knowledge, and an efficient pipeline for detecting, localizing, verifying, and grasping known objects. In both applications, the real-time object detection allows for reliable object perception at particularly low cycle times.

3 EFFICIENT SEGMENTATION OF RGB-D IMAGES

Decomposing sensory measurements into coherent parts is a fundamental prerequisite for scene understanding that is required for solving complex tasks, e.g., in the field of mobile manipulation. In this chapter, we describe methods for efficient segmentation of range images and organized point clouds acquired by consumer color and depth (RGB-D) cameras. In order to achieve real-time performance in complex environments, we focus our approach on simple but robust solutions. We present a fast approach to surface reconstruction in range images and organized point clouds by means of approximate polygonal meshing. The obtained local surface information and neighborhoods are then used to 1) smooth the underlying measurements, and 2) segment the image into planar regions and other geometric primitives. A comparative evaluation using publicly available datasets shows that our approach achieves state-of-the-art performance while being significantly faster than other methods.

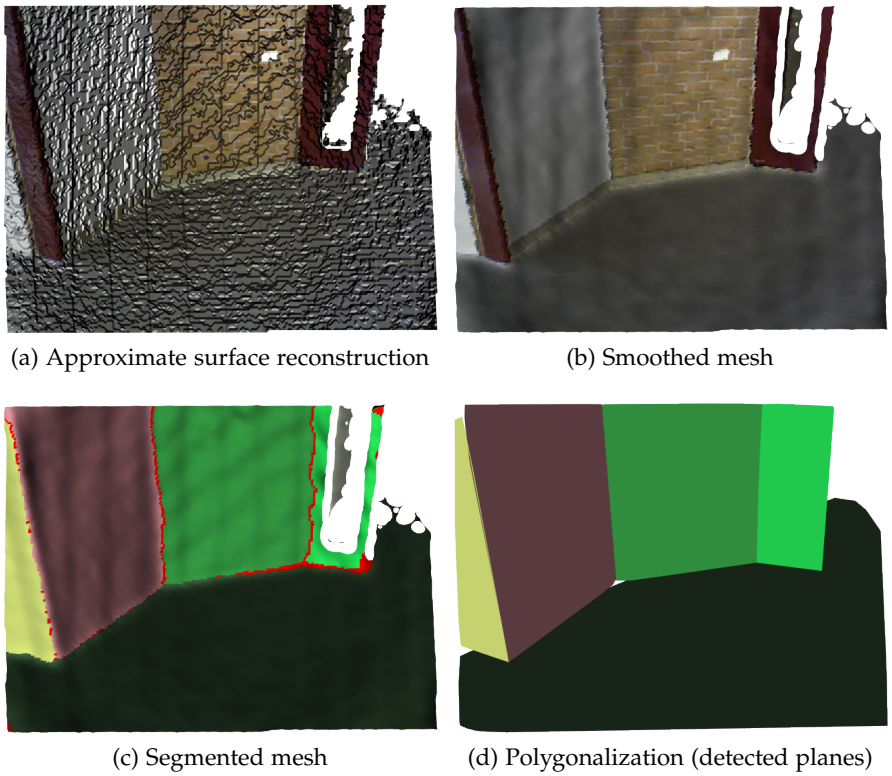
3.1 INTRODUCTION

As robots and autonomous systems move away from laboratory setups towards complex real-world scenarios, both the perception capabilities of these systems and their abilities to acquire and model semantic information must become more powerful. A key issue for this is the decomposition of sensory measurements into homogeneous parts that are relevant for the tasks of the robot. For mobile manipulation in complex environments, for example, the perception of objects and their surroundings is a key prerequisite. A common approach (Rusu *et al.*, 2009b) in three-dimensional (3D) object and environment perception is to exploit typical characteristics of man-made environments and to apply the following processing pipeline:

1. detect horizontal support planes,
2. extract and cluster points on top of these planes, and
3. perform further processing in the found clusters, e.g., recognizing, classifying or tracking objects.

One of the fundamental challenges in this pipeline is to segment the 3D data into planes and other geometric primitives—or regions of local surface continuity in general.

In this chapter, we address the problem of segmenting range images and organized point clouds in real-time using only a single CPU.



(a) Approximate surface reconstruction (b) Smoothed mesh

(c) Segmented mesh (d) Polygonalization (detected planes)

Figure 3.1: Surface reconstruction and plane segmentation of an RGB-D point cloud: (a) initial approximate triangulation; (b) the smoothed mesh after multilateral filtering; (c) the result of segmenting planes in the mesh (random colors per plane, red triangles are assigned to multiple planes and lie along borders); (d) a complete polygonalization of the scene as a collection of alpha shapes.

The central idea of our approach is to approximately reconstruct the surface and segment the range image by growing regions using the resulting local mesh neighborhoods. Through the use of easily exchangeable components, our generalized region growing approach allows for different region models (e.g., planes) to be segmented in the data. We present models for segmenting planes, regions of local surface continuity, and simple geometric primitives at high frame rates. Figure 3.1 shows an example of plane segmentation in an indoor environment.

We further use the same mesh neighborhoods to efficiently compute local surface normals and curvature estimates, as well as to smooth both the 3D measurements and the computed normals using a multilateral filter.

This chapter organized as follows: After a discussion of related work on range image and 3D plane segmentation methods in Section 3.2, we give an overview of our segmentation pipeline in Section 3.3. Our methods for approximate surface reconstruction, efficient computation of local surface normals and curvature, and filtering of the constructed

mesh are presented in Section 3.4. In Section 3.5, we describe our generalized region growing algorithm as well as different models for plane segmentation and the detection of other geometric primitives. We investigate different camera noise models that assist both initial mesh construction and segmentation in Section 3.6. We evaluate efficiency and robustness of our approach on multiple publicly available datasets and summarize the results in Section 3.7. We show that our approach achieves state-of-the-art performance while being faster than other methods.

The approximate surface reconstruction approach and the region growing based segmentation method were first published and presented at the 12th International Conference on Intelligent Autonomous Systems (Holz and Behnke, 2012). An extended version of this paper (and an early version of this chapter, respectively) was published in *Robotics and Autonomous Systems* (Holz and Behnke, 2014a).

3.2 RELATED WORK

Research on computer and robot vision has yielded a wide variety of approaches to range image segmentation—and plane segmentation in particular. Hoover *et al.* (1996) compiled a survey and performed an evaluation of early work. For an overview of more recent work, we refer to the survey of Vosselman *et al.* (2004). In principle, four different types of approaches can be distinguished according to the underlying working principle: methods using variants of the random sample consensus (RANSAC) algorithm (Fischler and Bolles, 1981), 3D Hough transforms, scan line grouping, and region growing.

3.2.1 *Segmentation based on Sample Consensus*

RANSAC-based approaches try to find models for geometric primitives that best explain a set of points and the set of inliers supporting it. For segmenting a complete range image, Lee *et al.* (1998) sequentially find a model using RANSAC, remove inliers from the original dataset, and continue the segmentation with the residual points. Silva *et al.* (2002) first identify connected regions and apply RANSAC region-wise. Gotardo *et al.* (2003) compute an edge map for pre-segmentation and fit model parameters using a robust estimator based on the M-estimator sample consensus (MSAC) by Torr and Zisserman (2000). Gotardo *et al.* (2004) extend the approach to segmenting planar and quadric surfaces using both a robust estimator—again based on MSAC—and a genetic algorithm for accelerating the optimization process and avoiding premature convergence. Bab-Hadiashar and Gheissari (2006) propose an approach that simultaneously identifies the type of geometric primitive for the underlying surface region while performing a complete segmentation of the input image. Segmentation and model selection

are based on defining and minimizing the strain energy of fitted surfaces. The resulting approach is able to segment various types of planar and curved objects.

Another efficient solution to the problem of segmenting even unorganized point clouds and detecting simple geometric primitives such as planes, cylinders, and spheres has been proposed by Schnabel *et al.* (2007). They decompose unorganized point clouds using octree subdivision and apply RANSAC only to subsets of the original point cloud.

In previous work (Section 2.3; Holz *et al.*, 2010), we adapted the perception scheme from Section 3.1 as well as the techniques of Rusu *et al.* (2009b) and Schnabel *et al.* (2007), and made them applicable to the measurements of time-of-flight (ToF) cameras. We presented techniques to cope with the specific error sources of the cameras, and to speed up processing by exploiting the image-like data organization. After detecting the most dominant plane, we applied the octree-based primitive detection of Schnabel *et al.* (2007) only to already extracted and segmented points above that plane. In another previous work (Section 2.4; Holz *et al.*, 2011), we further sped up the segmentation process by using integral images for computing local surface normals more efficiently, and using the index neighborhood underlying the 3D data to extract and track segments of points and object candidates, respectively. The overall approach is applicable in real time on a Microsoft Kinect™ RGB-D camera and has been used for real-time object tracking and grasp planning (Stückler *et al.*, 2011).

3.2.2 *Hough-based Plane Segmentation and Clustering*

The Hough transform is the de-facto standard for finding lines and circles in two-dimensional (2D) images. Various extensions to 3D exist that try to find, respectively, planes and maxima in histograms over the possible space of plane orientations and distances. For an overview of the different variants and an evaluation of Hough-based segmentation approaches, we refer to the works of Vosselman *et al.* (2004) and Borrmann *et al.* (2011).

RANSAC- and Hough-based segmentation share a common disadvantage. Points belonging to the same segment do not necessarily lie on connected components. Both approaches will merge plane segments if they share a common orientation and distance to the origin. For example, in a shelf with vertical separators, boards on the same level are merged, although they do not physically lie on the same surface. In addition, Hough-based segmentation may suffer from discretization effects.

Frigui and Krishnapuram (1999) present a robust clustering algorithm for finding various shapes in noisy datasets. It starts with a large number of clusters and iteratively reduces the number of clusters

(converging to the actual number of clusters) by competitive agglomeration. Despite other clustering applications, they also present an objective function for segmenting range images into planar regions. The approach is included in our comparative experimental evaluation in Section 3.7.3.

In (Holz *et al.*, 2011), we present a fast plane segmentation approach that uses a similar parameter space as the Hough transform. We pre-cluster points and segment planes first in normal space and then, for each cluster, in distance space to obtain individual planes. We compensate for discretization effects by conducting a post-processing step in which neighboring segments are merged if their parameters do not considerably deviate. Still, unconnected planar patches may get merged into the same cluster (in contrast to scan line grouping and the region growing-based approaches following).

3.2.3 *Scan Line Grouping*

In the context of segmenting 3D laser range scans, another popular approach is *scan line grouping*. It aims for computational efficiency by first detecting lines in planar cuts (and 2D range scans forming a 3D scan), and by merging neighboring line segments to regions in a second step. The basic idea behind this principle is the observation that planes in a 3D scan form straight lines in two-dimensional scan lines and that points on the same line segment belong to the same 3D plane. Jiang and Bunke (1994) store detected line segments in a link-based data structure that eases region growing for extracting planar patches from detected lines. The approach is extended and further evaluated by Jiang and Bunke (1996). Jiang (2000) then combines this edge detection with an adaptive edge grouping algorithm for efficiently solving the contour closure problem. The resulting approach successfully detects closed contours and achieves better segmentation results. It is included in our comparative experimental evaluation. Nüchter *et al.* (2003) follow a similar approach but use RANSAC and the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992) for plane detection, and label detected planes as belonging to floor, ceiling and walls. Gutmann *et al.* (2008) improve various aspects of the original formulation by computing and analyzing statistics about points on a scan line for splitting line segments and growing regions. An *et al.* (2012) first cluster the scan lines and only consider the end points of line segments to speed up line and plane detection. Recently, Georgiev *et al.* (2011) applied scan line grouping on data acquired from RGB-D cameras. We include the approaches of Jiang (2000) and Georgiev *et al.* (2011) in our experimental evaluation (Section 3.7.3).

3.2.4 *Segmentation using Region Growing*

The underlying idea of region growing-based segmentation is to exploit the image-like data structure of organized point clouds. Koster and Spann (2000) estimate the scale of local Gaussian distributions around points and then detect identical distributions. Segmentation is performed by iteratively growing regions of points that have identical Gaussian distributions. Checchin *et al.* (1997) also propose a two-stage approach making use of both edges and regions. They first compute an over-segmentation of the range images primarily based on local depth discontinuities. This over-segmentation is presented in a connectivity graph. Neighboring segments in the graph are then merged using a surface-based description and comparison. Hähnel *et al.* (2003) connect neighboring points in 3D laser range scans to a mesh-like structure. The scans are then segmented recursively by merging connected patches that are likely to lie on the same planar surface. Poppinga *et al.* (2008) apply the same approach to Time-of-Flight cameras and re-formulate the algorithm in an incremental fashion. They grow planar regions by adding neighboring points whose distances to the currently grown plane lie below a threshold. The centroid and covariance matrix for estimating the plane's parameters are thereby updated incrementally. Here, we follow a similar approach for segmenting planes. Instead of incrementally computing the covariance matrix however, we compute the normals for all points beforehand and simply average local surface normals to obtain an estimate of the plane normal. That is, we only store and incrementally update the centroids in both Cartesian and normal space. We include the approaches of Koster and Spann (2000) and Checchin *et al.* (1997) as well as our probabilistic plane segmentation based on the work of Poppinga *et al.* (2008) and our approximate variant in our experimental evaluation (Section 3.7.3).

Other popular region growing approaches to range image segmentation make use of local surface curvature. Regions are grown until points with a considerably larger curvature are reached. Just like Gotardo *et al.* (2003) for RANSAC-based segmentation, Harati *et al.* (2006), first compute an edge map to find connected regions of local surface continuity. Rabbani *et al.* (2006) approximate local surface curvature by first fitting planar segments to local point neighborhoods and then computing, for each point, the distance to that plane. Recently, Cupec *et al.* (2011) followed a similar approach. They first apply two-and-a-half-dimensional (2.5D) Delaunay triangulation on a range image to obtain an initial triangular mesh and then use the maximum distance of an examined point to all triangles in a region to determine whether or not the point is added.

Similar to our approach for segmenting geometric primitives such as cylinders and spheres is the work of Attene *et al.* (2006) who hierarchically cluster and decompose a triangular 3D mesh into geometric

primitives. They first initialize every single triangle to be an individual cluster and then form new clusters by connecting triangles (and clusters respectively) to neighboring triangles. When forming a new cluster the primitive best explaining the connected triangles is chosen to represent the new cluster. That is, clusters and detected primitives are iteratively grown and optimized. In contrast, we first pre-segment the mesh and estimate the best fitting primitive for each segment.

Here, we deduce a surface reconstruction directly from the image-like data structure, and use the local ring neighborhood around vertices to 1) efficiently compute local surface normals and curvature estimates, and 2) efficiently smooth the depth measurements using a multilateral filter. Our framework allows for using different types of models for region growing, including the approaches of Rabbani *et al.* (2006), Cupec *et al.* (2011) and Poppinga *et al.* (2008), as well as our fast approximation (see Section 3.5.2).

3.3 OVERVIEW OF THE SEGMENTATION PIPELINE

The proposed approach to range image segmentation consists of a complete pipeline of several methods ranging from data acquisition over pre-processing and surface reconstruction to segmenting the resulting mesh representation and detecting geometric primitives such as planes, cylinders and spheres. The overall processing pipeline of our approach is composed of the following components:

1. Deducing an approximate mesh from the image neighborhoods.
2. Using the mesh neighborhoods to compute approximate local surface normals and curvature estimates.
3. Multilateral filtering to smooth both points and normals.
4. Segmentation based on region growing (using different region models depending on the desired segmentation).

As an optional last step, we replace found segments by the geometric primitives best fitting the contained points in order to obtain a compact representation. Planes are replaced by polygons, e.g., the convex hull and the model parameters for the plane. An overview of our system is shown in Figure 3.2. Not shown in the overview figure are basic input and output operations such as retrieving range images, loading 3D point clouds from datasets or storing the final and intermediate results. Furthermore, depending on the configuration of the pipeline, some components can run repeatedly and alternating, e.g., computing the normals, filtering both points and normals, and then re-computing the normals for the filtered points rather than using the filtered normals.

Compared to related work, our approach is particularly efficient since local point neighborhoods as well as distances and changes in

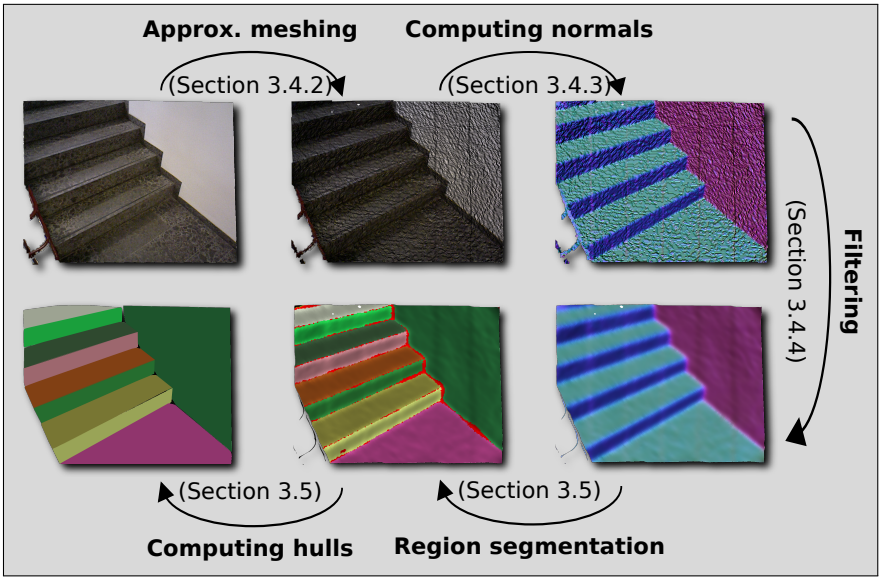


Figure 3.2: Overview of the processing pipeline (following the black arrows from top left): for an input organized point cloud or range image, we first deduce an approximate triangle or quad mesh. We efficiently compute local surface normals and curvature directly on the mesh and apply a multilateral filter to smooth both the points and their normals. The smoothed mesh is then segmented into planar regions. In a last (optional) processing step, detected planes are replaced by polygons.

surface orientations between neighboring points are not only computed efficiently using the approximate mesh but also cached in the mesh structure for further processing. All components described in the above list use the same cached neighborhoods. Moreover, there is a wide range of mesh processing algorithms available that makes use of the contained topology. Both the possibility to cache information in the edges and the availability of a large corpus of mesh processing algorithms constitute the two reasons why polygonal meshes are used as the underlying data representation in this pipeline.

3.4 FAST APPROXIMATE SURFACE RECONSTRUCTION

Whereas range images form a dense 2.5D representation, point clouds are ordered or unordered collections of points in 3D space. Naturally, these points provide an intuition of the shape of the surface they have been sampled from but cannot explicitly represent this surface. Surface reconstruction algorithms aim at building a surface representation from the loose collection of 3D points, e.g., in the form of polygonal meshes (explicit representation) or in the form of a signed distance field (implicit form). Polygonal meshes approximate the surface using a collection of polygons with arbitrary topology. Thereby, they represent the underlying surface in the form of piece-wise smooth

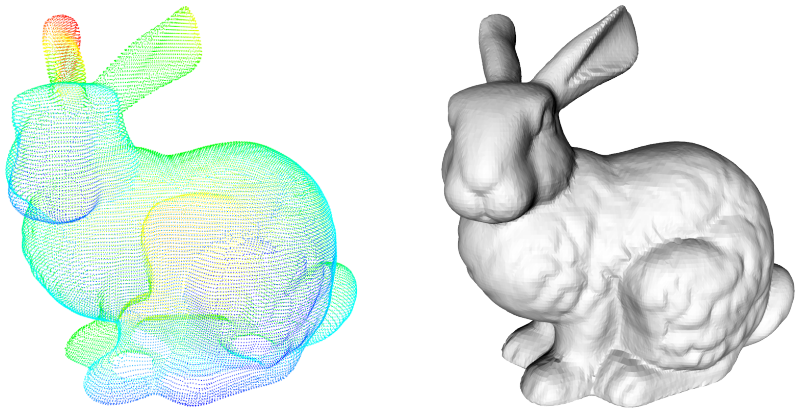
surfaces. A commonly used form of polygonal meshes is the triangle mesh. It can be represented as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$ with vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, edges $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, and triangles $\mathcal{F} = \{f_1, f_2, \ldots, f_k\}$. Edges form straight line segments between two vertices and encode adjacency, i.e., $e_i \in \mathcal{V} \times \mathcal{V}$. Triangles include three edges and connect three vertices in the topology, i.e., $f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$. Every vertex has a position in 3D space and, consequently, the vertex positions themselves form a 3D point cloud $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$.

3.4.1 *Surface Reconstruction Algorithms*

Surface reconstruction is a well understood problem and a wide variety of algorithms has been proposed in the last decades. In general, surface reconstruction has two goals: 1. the reconstructed surface should be geometry close to the sampled surface, and 2. the reconstructed surface should be topologically equivalent to the sampled surface. That is, neighboring sample points on the object should also be close to each other in the topology of the reconstructed mesh and the reconstructed mesh should not (considerably) extend into the interior or exterior of the sampled object. Several distance error metrics for evaluating the quality of surface reconstructions and especially the local smoothness of the resulting meshes have been investigated by Berger *et al.* (2013).

One of the classic early algorithms for surface reconstruction is *Marching Cubes* (Lorensen and Cline, 1987). It first computes a voxel-based subdivision of the input point clouds, and then iterates over the neighboring voxels of a voxel to compute the local orientation of the surface. The efficiency of the approach comes from explicitly formulating all connectivity cases and looking up the corresponding polygon configuration for the local isosurface in the voxel. The exact positions of the vertices on the edges of the voxels is computed by linear interpolation. Several extensions and improvements especially regarding the interpolation have been proposed in the last years including the work by Hoppe *et al.* (1992) and the approach of Carr *et al.* (2001) that uses radial basis functions. Another approach for both uniform and adaptive grid structures has been proposed by Li *et al.* (2010) for reconstructing extremal surfaces.

Many surface reconstruction algorithms start with Delaunay triangulations or Voronoi diagrams like, for instance, the *Cocone* family of algorithms (Amenta *et al.*, 2000; Dey *et al.*, 2001; Dey and Goswami, 2003). A particularly popular algorithm is *Poisson surface reconstruction* (Kazhdan *et al.*, 2006). Given a set of points with local surface normals, Poisson surface reconstruction aims at fitting an implicit function to the surface whose values are zero at the points and whose gradient at the point equals to the normal vector. The function and the optimization are based on *Poisson's equation*—a partial differential equation of elliptic type. Kazhdan and Hoppe (2013) extend the ap-



(a) 3D point cloud (b) Reconstructed mesh

Figure 3.3: Example of 3D surface reconstruction. Shown are a point cloud of the well-known *"Stanford Bunny"* as scanned by Turk and Levoy (1994) and a polygonal reconstruction obtained using Poisson Surface Reconstruction (Kazhdan *et al.*, 2006).

proach to explicitly incorporate the sampling points as interpolation constraints and also reduce the runtime complexity of the optimization to be linear in the number of points, i.e., $\mathcal{O}(n)$. An example of a reconstructed surface from a sample of 3D points using Poisson reconstruction can be seen in Figure 3.3.

Another popular approach is the Smooth Signed Distance (SSD) Surface Reconstruction of Calakli and Taubin (2011). It computes signed distance functions in an octree representing the input point cloud. Instead of forcing the implicit function to approximate the implicit surface, SSD reconstruction forces the implicit function to be a smooth approximation of the signed distance function to the surface. This reconstruction algorithm and its variants became particularly popular with the advent of 3D mapping algorithms that inherently make use of signed distance functions such as KinectFusion (Newcombe *et al.*, 2011). After building the grid-based map, the reconstructed surface can be easily computed from the signed distance functions. Wiemann *et al.* (2016), for example, use an open source implementation of a large-scale KinectFusion variant, extract the reconstructed surface from the truncated signed distance functions, and apply an efficient Marching cubes implementation to obtain topologically correct surface reconstructions. For building textured reconstructions from sequences of RGB-D images, Steinbruecker *et al.* (2014) compute signed distance functions in an octree which is updated when new data arrives. The surface reconstruction is derived from the signed distance functions in the octree cells, and updated octree cells change by keeping track of the dependencies between mesh regions and octree cells.

All the methods mentioned above do not assume a particular ordering of the points and do not exploit a certain structure. Hence,

algorithms exploiting a certain structure can be considerably faster in case the input has the assumed structure. Schmitt and Chen (1991) proposed a planar segmentation approach which includes an approximate surface reconstruction specifically designed for range images. In essence, it computes a 2.5D Delaunay triangulation in the range image. Since it is computationally less complex than the full 3D approaches for unorganized point clouds, the triangulation has been used in several other methods, including the range image segmentation algorithm by Cupec *et al.* (2011).

A hybrid approach for unorganized 3D points is to extract local neighborhoods around points, fit a plane to the neighborhood, project the points onto the plane, and then compute a local 2.5D triangulation on this plane (Gopi and Krishnan, 2002). A particularly efficient and robust variant of this approach has been presented by Marton *et al.* (2009).

3.4.2 *Exploiting Structure for Approximate Meshing*

The central idea of our surface reconstruction approximation is to deduce the desired mesh structure directly from the image-like organization of measurements. In fact, the following algorithms could easily be applied on local index neighborhoods in range images. However, an approximate mesh allows for 1) the application of a wide variety of sophisticated algorithms for processing meshes, and 2) the storage of edge weights for caching point attributes or relations between points, e.g., differences in local surface normal orientations or the difference vectors for integral image-based normal computation as in our previous work (Holz *et al.*, 2011).

We traverse a given range image or organized point cloud P once and check for every point $\mathbf{p}_i = P(u, v)$

- if $P(u, v)$ and its neighbors $P(u, v+1)$, $P(u+1, v+1)$, and $P(u+1, v)$ in the next row and the next column are valid depth measurements, and
- if all edges between $P(u, v)$ and these three neighbors are not occluded, i.e., the edge connects points lying on the same physical surface.

The first check is necessary because of the structure in the sensory data that we are exploiting. If the sensor cannot acquire a valid depth measurement for a certain pixel, it stores an invalid one, in order to keep the structure organized. The invalid measurements are then ignored in the reconstruction.

The latter occlusion checks are necessary to avoid that points lying on different physical surfaces are connected. When one surface occludes another spurious measurements may be caused along these so-called *jump edges* (May *et al.*, 2009). Obviously, these phantom points

should not be included in the final surface reconstruction. Furthermore, even if no jump edges cause such spurious measurements, a point on the occluding surface should not be connected to a point on the occluded surface. Both can be efficiently checked for by examining the difference vectors between $\mathbf{p}_i$ and its three neighbors. If one of the difference vectors falls into a common line of sight with the viewpoint from where the measurements were taken (the focal point $\mathbf{f} = \mathbf{0}$), then one of the underlying surfaces occludes the other. The condition for having an occluded edge between point $\mathbf{p}_i$ and its neighbor $\mathbf{p}_j$ can be formulated as

$$valid = \left(\left|\cos\theta_{i,j}\right| \leq \cos\epsilon_\theta\right) \wedge \left(d_{i,j} \leq \epsilon_d^2\right), \tag{3.1}$$

$$\text{with} \quad \theta_{i,j} = \frac{(\boldsymbol{p}_i - \boldsymbol{f}) \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j)}{\|\boldsymbol{p}_i - \boldsymbol{f}\| \, \|\boldsymbol{p}_i - \boldsymbol{p}_j\|}, \tag{3.2}$$

$$\text{and} \quad d_{i,j} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2, \tag{3.3}$$

where ϵ_θ and ϵ_d denote maximum angular and length tolerances, respectively. In the same way, we can check for jump edges (May *et al.*, 2009), i.e., erroneous measurements often induced by range sensors in the vicinity of occlusions and depth discontinuities. Note that in addition to checking whether or not an edge falls into a common line of sight with the focal point, we also use a distance threshold in Equation (3.1). Whenever points are too far apart from each other a connecting edge can be avoided even if the connecting edge passes the angle check in Equation (3.2).

If all checks pass, $P(u, v)$ and its neighbors are used to extend the so far built mesh. Otherwise, holes arise. Referring to Figure 3.4, we distinguish four types of meshes:

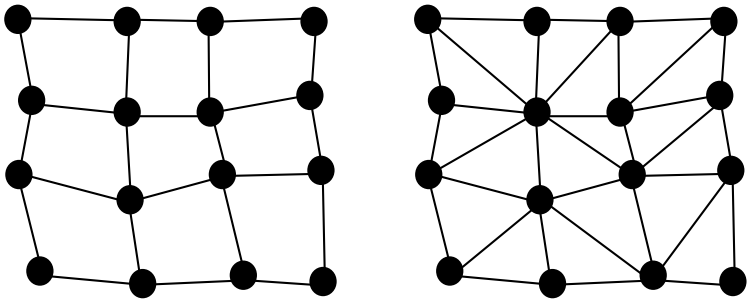
1. Quad meshes are formed by connecting the point $P(u, v)$ to $P(u, v+1)$, $P(u+1, v+1)$ and $P(u+1, v)$. After going over all points, this forms quads, i.e., polygons with four edges.

2. Fixed left cut and right cut triangular meshes are formed by cutting quads either from top right to bottom left (left cut) or from top left to bottom right (right cut).

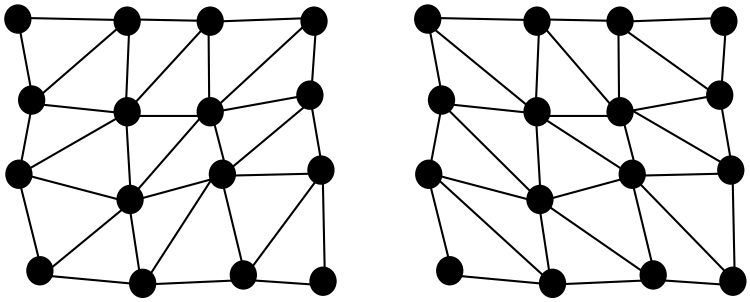
3. Adaptive triangulation cuts the quad along the diagonal that has a smaller length. Compared to the fixed triangulations, it achieves a higher accuracy in the vicinity of edges.

For triangulations, a single invalid neighbor causes that only one triangle is added. After construction, we simplify the resulting mesh by removing all vertices that are not used in any polygon. Example triangulations are shown in Figures 3.1 and 3.2.

Recently, Orts-Escolano *et al.* (2015) us this approximate surface reconstruction in their pipeline for computing local feature descriptors for object recognition on a graphics processing unit (GPU). In addition



(a) Quad mesh (b) Adaptive triangle mesh



(c) Triangle mesh: left cuts only (d) Triangle mesh: right cuts only

Figure 3.4: Visualization of mesh types: fast approximate meshing using a quad mesh (a) and different triangulations (b-d). Compared to the adaptive approach (b), triangulations using only left cuts (c) or only right cuts (d) can be obtained slightly faster.

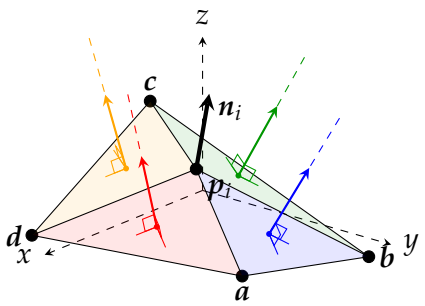
to the two validity checks, they introduced a third criterion checking for the compatibility of surface normal orientations, i.e., whether a point and its neighbors have similarly oriented surface normals. Consequently, the mesh is not only disconnected at depth discontinuities between surfaces but also at differently oriented surfaces, e.g., convex and concave edges between walls. We do not check for similar surface normal orientations in order to have all physically connected surfaces connected in the mesh.

3.4.3 *Fast Computation of Local Surface Normals*

We compute the local surface normal $\mathbf{n}_i$ for a point $\mathbf{p}_i$ as the weighted average of the plane normals of the N_T faces surrounding $\mathbf{p}_i$.

$$\mathbf{n}_i = \frac{\sum_{j=0}^{N_T} w_j \boldsymbol{n}_j}{\| \sum_{j=0}^{N_T} w_j \boldsymbol{n}_j \|}, \tag{3.4}$$

There are different ways of including and weighting neighboring triangles in Equation (3.4). For an extensive overview of these different



with

	$\triangle_1$	$\triangle_2$	$\triangle_3$	$\triangle_4$
p_i	$= p_{1,a}$	$= p_{2,a}$	$= p_{3,a}$	$= p_{4,a}$
a	$= p_{1,b}$			$= p_{4,c}$
b	$= p_{2,c}$	$= p_{3,b}$		
c		$= p_{3,c}$	$= p_{4,b}$	
d			$= p_{4,c}$	$= p_{1,b}$

Figure 3.5: Computing local surface normals on the mesh. After computing the four face normals, the local surface normal $\boldsymbol{n}_i$ of the point $\boldsymbol{p}_i$ is obtained by averaging over the face normals and normalizing the resulting vector.

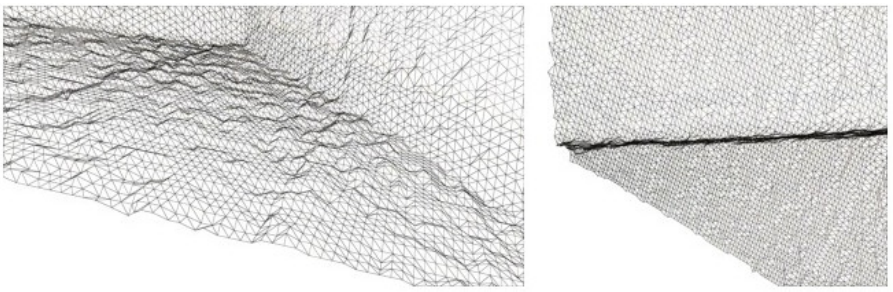
formulations we refer to the work of Jin *et al.* (2005). Using the cross product between the difference vectors of the bounding vertices to compute the face normals $\boldsymbol{n}_j$ and not normalizing the resulting vectors automatically gives a higher influence to larger faces, i.e., faces where a higher stability in surface normal orientation can be assumed. That is, by not normalizing the face normals and choosing weights of $w_j = 1$, we implicitly choose the weights to be proportional to the area of the surrounding triangles. We compute the normal $\mathbf{n}_i$ as:

$$\mathbf{n}_i = \frac{\sum_{j=0}^{N_T} (\mathbf{p}_{j,a} - \mathbf{p}_{j,b}) \times (\mathbf{p}_{j,a} - \mathbf{p}_{j,c})}{\left\| \sum_{j=0}^{N_T} (\mathbf{p}_{j,a} - \mathbf{p}_{j,b}) \times (\mathbf{p}_{j,a} - \mathbf{p}_{j,c}) \right\|}, \tag{3.5}$$

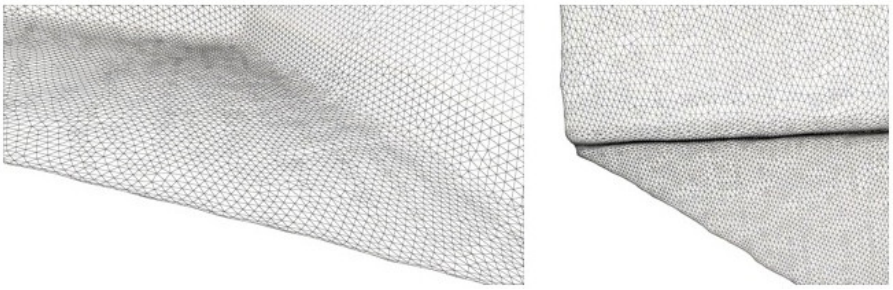
where $\mathbf{p}_{j,a}$, $\mathbf{p}_{j,b}$ and $\mathbf{p}_{j,c}$ form triangle j. Our formulation of the approximation resembles the seminal work of Gouraud (1971).

An example of computing the local surface normal of a point surrounded by four faces in a triangle mesh can be seen in Figure 3.5. In the actual implementation, we simply iterate over the faces, compute the difference vectors and their cross products, and add them to the normals of the involved points. Finally, we normalize all point normals at once. The local surface curvature of a point is then approximated by the standard deviation of the normal directions of its neighbors (Magid *et al.*, 2007). An example of computed local surface normals (color coded) can be seen in Figure 3.2.

Since we compute local surface normals on the mesh deduced from the range image and not on the range image itself (as in the case of integral image-based normal estimation (Holz *et al.*, 2011; Holzer *et al.*, 2012) we get proper normal estimates even in the vicinity of depth discontinuities whereas image-based methods tend to smooth over edges and depth discontinuities.



(a) Views of an unfiltered mesh. Noise produces high curvature corners on planes.



(b) Views of the filtered mesh. Planar regions appear smooth in the mesh.

Figure 3.6: Typical results of multi-lateral filtering: two views on an adaptive triangular mesh (a) before and (b) after filtering. The surface is considerably smoothed while preserving edges and corners.

3.4.4 *Multilateral Filtering*

Naturally, sensor measurements are affected by noise. Since this noise may hinder further processing, e.g. segmentation, we apply a filter for smoothing both the points and their normals while preserving edges in the sensed geometric structures. The formulation of our filter is motivated by the concept of multilateral filtering (Butt and Rajpoot, 2009) and measures the similarity of points w.r.t. their position, surface orientation, and appearance. As for the other components in our pipeline, we directly extract the neighborhood of a point from the mesh instead of searching for nearest neighbors. We filter both a point $\mathbf{p}_i$ and its normal $\mathbf{n}_i$ over its 1-ring-neighborhood N_i, i.e., all points that are directly connected to $\mathbf{p}_i$ by an edge in the mesh:

$$\mathbf{p}_i = \sum_{j \in N_i} w_{ij} \mathbf{p}_j \Big/ \sum_{j \in N_i} w_{ij}, \quad \text{and} \tag{3.6}$$

$$\mathbf{n}_i = \sum_{j \in N_i} w_{ij} \mathbf{n}_j \Big/ \sum_{j \in N_i} w_{ij}, \tag{3.7}$$

$$\text{with} \quad w_{ij} = \underbrace{e^{\alpha \|\mathbf{p}_i - \mathbf{p}_j\|}}_{\text{distance term}} \underbrace{e^{\beta \|\mathbf{n}_i - \mathbf{n}_j\|_1}}_{\text{normal term}} \underbrace{e^{\gamma (I_i - I_j)/c_I}}_{\text{intensity term}}, \tag{3.8}$$

where the optional intensity term is only evaluated for colored point clouds and range images where also an intensity image is available. The normalization constant c_I is used to scale the intensity differences to lie in the interval $[0, 1]$. Weights α, β, and γ can be used to adjust

the behavior of the filter. Equally weighting distance and surface normal deviation term already achieves considerable smoothing while preserving edges and corners. Depending on the desired smoothing level, we can extend a point's neighborhood to include the neighbors of neighbors and ring neighborhoods farther away from the point. An example of filtering an input mesh (weights $\alpha = 1$, $\beta = 1$, $\gamma = 0$) can be seen in Figure 3.6.

A similar filter has been used by Fleishman *et al.* (2003) to smooth the surface of 3D object meshes. However, the only incorporate the Euclidean distance between points and their distances to a plane fit through the local neighborhood, whereas our multilateral filter incorporates three components: the Euclidean distance, the deviation of local surface normals and the difference in intensity.

3.5 REGION MODELS AND SEGMENTATION

For being able to efficiently segment 3D data of various types, e.g., range images as well as both organized and unorganized point clouds, we have implemented a generalized segmentation framework. It allows the involved components such as the underlying region model (e.g., planes or non-planar locally smooth surfaces), the module for estimating the sensor noise, and the method for obtaining the local neighborhoods of query points (and the corresponding distances and surface normal deviations) to be easily switched. Whereas for organized data, we only require a look-up in the initially constructed and smoothed mesh, for unorganized data we need to either apply the greedy projection-based surface reconstruction algorithm by Marton *et al.* (2009) or search for the neighbors for all points using fast approximate neighbor search (Muja and Lowe, 2009) and cache the neighbors for later use.

3.5.1 *Region Growing-based Segmentation*

Despite the generalization over different neighborhood searches and region models, the implementation of our segmentation algorithm does not considerably deviate from other region growing algorithms in literature. Given is a set of seed points (and a priority queue of seeds) or simply the array of all points.

Outer loop, until all points are processed:

1) Select the next seed point,

2) initialize the region model of interest, and

3) put the seed point onto the empty processing queue.

Inner loop, while the processing queue is not empty:

4) Take the next point from the processing queue (and go back to 1) if it is empty),
5) check the compatibility of the point with the region model, and
6) add it in case of compatibility. If necessary, update the region model w.r.t. the new point.
7) Add the neighbors of the point to the processing queue if they pass a neighbor compatibility check.

3.5.2 *Different Region Models for Segmentation*

We have encapsulated the processing steps 2) initialization, 5) point compatibility, 6) model update, and 7) neighbor compatibility in exchangeable region models allowing the behavior of the segmentation to be configured and controlled. Note that we distinguish two types of compatibility checks—one for points determining whether or not they belong to the currently grown region and one for a point's neighbors determining whether or not they are added to the processing queue at all.

We have implemented several region models for plane segmentation: a probabilistic incremental formulation based on the approach of Poppinga *et al.* (2008) and an approximate variant using local surface normals, as well as for segmenting regions of local surface continuity as a pre-segmentation for further processing. In addition, we added models that reproduce the behavior of other segmentation algorithms found in the literature, amongst others, the approaches of Rabbani *et al.* (2006) and Cupec *et al.* (2011).

3.5.3 *Probabilistic Plane Segmentation*

In order to reliably detect planes even in noisy data, we use a probabilistic region model that is based on the incremental plane fitting algorithm of Poppinga *et al.* (2008). Aimed at time-of-flight cameras, their approach exploits the sequential structure of the data and uses incremental updates of both the centroid and covariance matrix of the inliers. The centroid and covariance matrix are not only used to determine the plane parameters, but also the uncertainty of the estimated plane parameters. A point is added to the probabilistic region model if the incrementally updated mean square error of the plane fit and the point's distance to the estimated plane model do not exceed a threshold. Despite the efficient incremental updates to centroid, covariance and mean square error, the probabilistic model is computationally more demanding than other region models in our framework, but achieves reliable plane detection results (Section 3.7).

3.5.4 *Approximate Plane Segmentation*

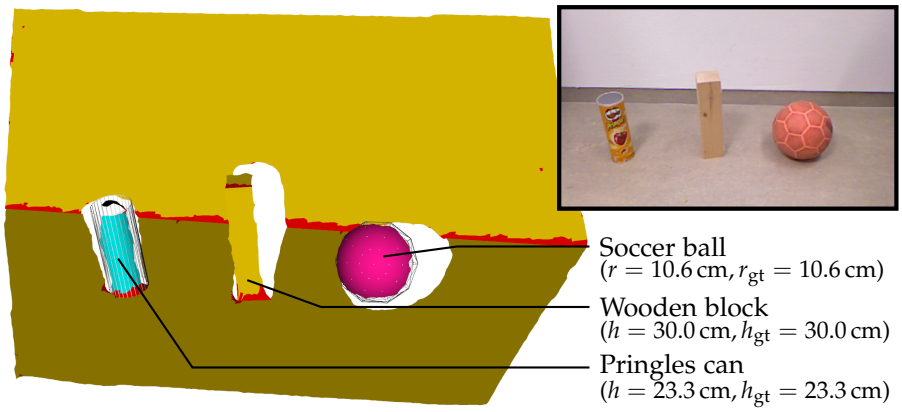
In order to further speed up plane segmentation while preserving reliable plane detections, we have developed an approximate variant of the probabilistic plane segmentation model. For approximate plane segmentation, we initialize the centroid and the normal of the region model using the seed point and its normal. Both can be taken directly from the smoothed approximate mesh instead of computing an initial centroid and covariance matrix using an initial set of points as in the probabilistic approach. In order to determine the compatibility of a point $\mathbf{p}_i$ with the model, we simply check the angle between its normal $\mathbf{n_i}$ and the normal of the plane model, as well as $\mathbf{p}_i$'s distance to the plane. To incorporate $\mathbf{p}_i$ in the model, we incrementally update the plane's centroid, but instead of incrementally updating a covariance matrix to derive a plane normal from it, we incrementally update its centroid in normal space. That is, by pre-computing the surface normals on the mesh neighborhood, and approximating the plane normal by averaging over point normals, we can reduce the number of required computations considerably. Note, for assessing the uncertainty in the determined plane parameters, e.g., for the registration of planar segments as done by Pathak *et al.* (2009), we can compute the necessary Hessian and covariance matrices after region growing using the final sets of inliers.

In order to obtain full polygonalizations such as the one shown in Figure 3.2, we compute the convex or concave hulls (using alpha shapes) for all planar patches. In case a triangulation is required for further processing, we decompose resulting polygons again using ear clipping (which is fast and produces satisfactory results in most cases). This post-processing step allows for the creation of highly efficient scene representations—thousands of triangles or quads are replaced by a fraction as many polygons, while still representing all dominant planes.

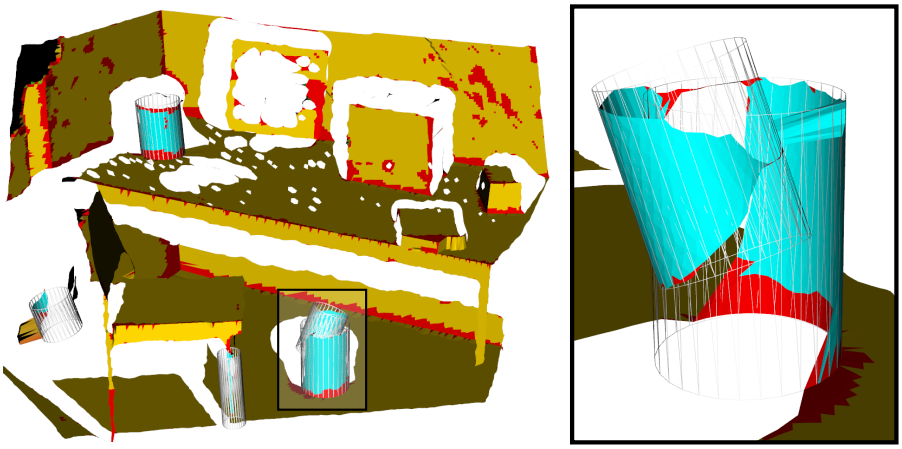
3.5.5 *Smooth Surfaces and Geometric Primitives*

For detecting geometric primitives, we need a rough pre-segmentation of the scene. This can easily be accomplished within the neighbor compatibility check of the models by, e.g., examining changes in the local surface curvature or comparing a point's surface normal with either the region's mean surface normal or the surface normal of the seed point. A typical result of applying a segmentation that uses the latter model is shown in Figure 3.7. Points on the same physical (locally smooth) surface end up in the same segment.

For every locally smooth segment, we try to find the geometric primitive that best explains the underlying point set. Whereas we can directly compute a least squares plane fit to all the points in a segment,



(a) Examples of all primitives (cylinder, planes, and sphere) in a simple scene



(b) Segmented example scene with detail view of the segmentation error

Figure 3.7: Detecting planes (yellow), cylinders (cyan), and spheres (magenta). Points and polygons belonging to multiple segments are colored red. All points are projected onto the found models. (a) The estimated model parameters are accurate to the millimeter, e.g., measuring $h = 23.302\,\text{cm}$ for the Pringles can instead of the ground truth object height $h_{\text{gt}} = 23.368\,\text{cm}$. (b) In case of cylinders where outer and inner part form two distinct clusters, two cylinders are detected. The model parameters for the inner part tend to be prone to errors.

we use RANSAC to find the best sphere and cylinder model. Here, the computational efficiency of our approach comes from applying RANSAC only if the computed planar model does not fully explain the segment. Typical results of applying the rough pre-segmentation and primitive detection are shown in Figure 3.7.

Since we still stick to the RANSAC-based primitive detection for spheres and cylinders, we focus the experimental evaluation to plane segments directly obtainable from region segmentation and extracted using our approximate model.

3.6 CAMERA NOISE MODELS

The key to both the initial construction of the approximate mesh and its segmentation is to know whether or not neighboring points in the range image are close to each other and lie on the same physical surface.

For all of the aforementioned region models, parameters such as, for instance, the distance to the model and the deviation between surface normals play an important role. Whereas the latter can (almost) be neglected after applying the multilateral filter, the distance to the model is a parameter that is crucial for the quality of the segmentation. It resembles the amount of noise hindering a measurement from lying on the ideal model.

In order to obtain a rough estimate of the amount of noise at a given point, we use a simple isotropic noise model. As suggested by Anderson *et al.* (2005), we assume Gaussian noise $\mathcal{N}(0, \sigma^2)$ and use a simple quadratic polynomial as a function of distance to determine σ, since noise in range sensors usually increases quadratically with the measured distance. Since the primary sensor used in our work is a Microsoft Kinect™ RGB-D camera, we have computed a simple error model for this sensor. In 10 different scenes (ranging from scenes with only close range measurements to views of wide open space), we have collected 100 range images each. For each of the locations, we compute the mean and standard deviation per pixel and perform a least squares fit to find appropriate coefficients for the quadratic model; resulting in:

$$\sigma_{\text{FIT}}(z) = 0.00263z^2 - 0.00519z + 0.00755. \tag{3.9}$$

This simple model considers only the expected measurement noise but already provides a good estimate (see Section 3.7.2) although it neglects both the characteristic camera errors induced by the quantization of measurements, and the angle to the surface that measurements are acquired on. Measurements taken at extreme angles (e.g., on walls while traversing a corridor) are considerably more affected by noise. Nguyen *et al.* (2012) measure both lateral and axial noise distributions as a function of both distance and angle to the sensed surfaces. They

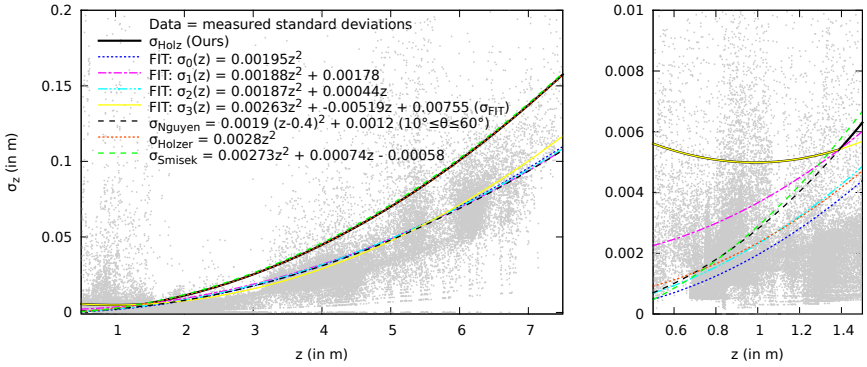


Figure 3.8: Isotropic noise models for Microsoft Kinect™ cameras (left). The detail view (right) shows that most models underestimate the noise in close ranges, compared to our fitted σ_3 model.

report an (almost) angle independent error when neglecting spurious data measured under extreme angles (smaller than 10° or larger than 60°). For this regular case their approximation is:

$$\sigma_{\text{Nguyen}}(z) = 0.0019\,(z - 0.4)^2 + 0.0012. \tag{3.10}$$

Holzer *et al.* (2012) neglect the angle-dependent error and provide a noise model solely based on the quantization effect induced by the measurement principle:

$$\sigma_{\text{Holzer}}(z) = 0.0028z^2. \tag{3.11}$$

Smisek *et al.* (2011) conduct a similar set of experiments and assess the quantization effect of a Microsoft Kinect camera as being

$$\sigma_{\text{Smisek}}(z) = 0.00273z^2 + 0.00074z - 0.00058. \tag{3.12}$$

We have conducted a set of experiments (see Section 3.7.2) to evaluate the influence of the noise model used for both approximate meshing and region segmentation. Although the final segmentation performance does not considerably deviate for the different noise models, the best results can be achieved with a combination of the quantization-based models of Holzer *et al.* (2012) and Smisek *et al.* (2011), and the fitting-based models of Nguyen *et al.* (2012) as well as our σ_{FIT}). In the remainder of this chapter we will use this combination which we simply coin σ_{Holz}.

$$\sigma_{\text{Holz}}(z) = \begin{cases} \sigma_{\text{FIT}}(z) & \text{if } z \leq 0.85\,\text{m, and} \\ \sigma_{\text{Holzer}}(z) & \text{if } z > 0.85\,\text{m.} \end{cases} \tag{3.13}$$

The different natures of the two model types are reflected by the two curve clusters in Figure 3.8.

3.7 EXPERIMENTS AND RESULTS

In order to assess the performance of our approach and the influence of the individual components, we perform a set of experiments. We evaluate the correctness and efficiency of our approach using two publicly available datasets for which ground truth plane (and cylinder) segmentations are available: the SegComp dataset collection[1] by Hoover *et al.* (1996), and the recently published Kinect datasets[2] by Oehler *et al.* (2011). The latter follows the same file formats and conventions as the SegComp datasets. From the SegComp dataset, we have used the ABW and the PERCEPTRON parts. For the evaluation, we follow (and refer to) the scheme of Hoover *et al.* (1996). The two SegComp parts consist of 10 training and 30 test images. For all images ground truth segmentations and plane parameters are available. The evaluation tool averages over all test images correctly segmented planes, oversegmented planes, undersegmented planes and false detections (labeled as "noise"). The Kinect dataset by Oehler *et al.* (2011) consists of two parts—one for planes and one for cylinders—with 30 organized RGB-D point clouds each. Training and test data is not distinguished and the evaluation includes all point clouds. Ground truth plane orientations are not available for the Kinect dataset. However, Oehler *et al.* provide a modified evaluation tool[2] that neglects plane orientations. The following comparative evaluations include segmentation results gathered by Gotardo *et al.* (2003) and Oehler *et al.* (2011), achieved results using publicly available implementations of Georgiev *et al.* (2011) and Trevor *et al.* (2013), our previous work (Holz *et al.*, 2011), and the plane and cylinder segmentation models used in this work.

In all experiments, we follow the original metrics as used by Hoover *et al.* (1996), i.e., we inspect the numbers of correctly detected planes, oversegmented planes, undersegmented planes, not detected planes and erroneously detected planes all averaged over all images in the respective testing dataset (see the legend in Table 3.1). Furthermore, in order to allow a direct comparison with the results reported by Hoover *et al.* (1996), Gotardo *et al.* (2003), and Oehler *et al.* (2011), we use a pixel overlap of 80 %.

3.7.1 *Runtime Evaluation*

In order to assess the runtime of the overall approach and of the involved components, we have defined a baseline system consisting of the following components: adaptive triangular meshing, our noise model σ_{Holz}, the multilateral filter with weights $\alpha = 1$, $\beta = 1$, $\gamma = 0$, and the region model for approximate plane segmentation. The

1 The SegComp datasets are available at: `http://marathon.csee.usf.edu/seg-comp`.

2 The segmentation datasets and the modified evaluation tool of Oehler *et al.* are available at: `http://www.ais.uni-bonn.de/download/segmentation/kinect.html`.

Result	Description
correct	Average number of correctly segmented planes.
over	Average number of over-segmented planes, i.e., correctly found planes but detected as more than one plane segment.
under	Average number of under-segmented panes, i.e., detected planes that span over more than one groud truth plane.
miss	Average number of undetected planes, e.g., smaller planes in the ground truth annotation that are not detected.
noise	Average number of planes that are not existing in the ground truth annotation, e.g., detecting a plane in a region labeled as noise, or detecting plane segments on the surface of a cylinder.

Table 3.1: Legend for the comparative evaluations using the SegComp datasets and evaluation toolkit of Hoover *et al.* (1996) and Oehler *et al.* (2011). All values are averaged over all images of the respective testing dataset.

baseline system is applied to all training and test images of both the SegComp and the Kinect datasets. The Kinect dataset has been downsampled to assess the runtimes for all resolutions offered by the camera (see Table 3.2). We can segment all dominant planes with roughly 7.7 Hz at VGA resolution, and more than two times faster than the camera measurement frequency at the downsampled 160×120 resolution.

Compared to the approach of Oehler *et al.* (2011), we obtain better segmentation results while being faster by a factor of eight (they report runtimes of $>$100 ms for 160×120, and > 2 s for 640×480). The approach of Trevor *et al.* (2013) achieves roughly 10 Hz (VGA) at the cost of inferior segmentation results, just as with the approach of Georgiev *et al.* (2011) with roughly 8 Hz (VGA). For both implementations parameters yielding the best results have been chosen. Slightly faster than the approach of Trevor *et al.* (2013) is our clustering method from previous work (Holz *et al.*, 2011) with roughly 15 Hz (VGA), again at the cost of segmentation performance. Using the probabilistic plane segmentation model (instead of our approximate one), the runtime for the segmentation step in Table 3.2 increases to approximately 95 ms leading to an overall computation frequency of roughly 5 Hz at VGA resolution.

	Resolution			
Component	160×120	320×240	512×512	640×480
Mesh Reconstruction	6.1 ms	17.2 ms	40.4 ms	48.4 ms
Computing normals	1.5 ms	3.6 ms	12.2 ms	14.1 ms
Filtering	4.5 ms	11.6 ms	27.4 ms	33.5 ms
(Plane) segmentation	3.2 ms	11.3 ms	26.3 ms	32.9 ms
Overall frequency	≈65 Hz	≈23 Hz	≈10 Hz	≈7.7 Hz

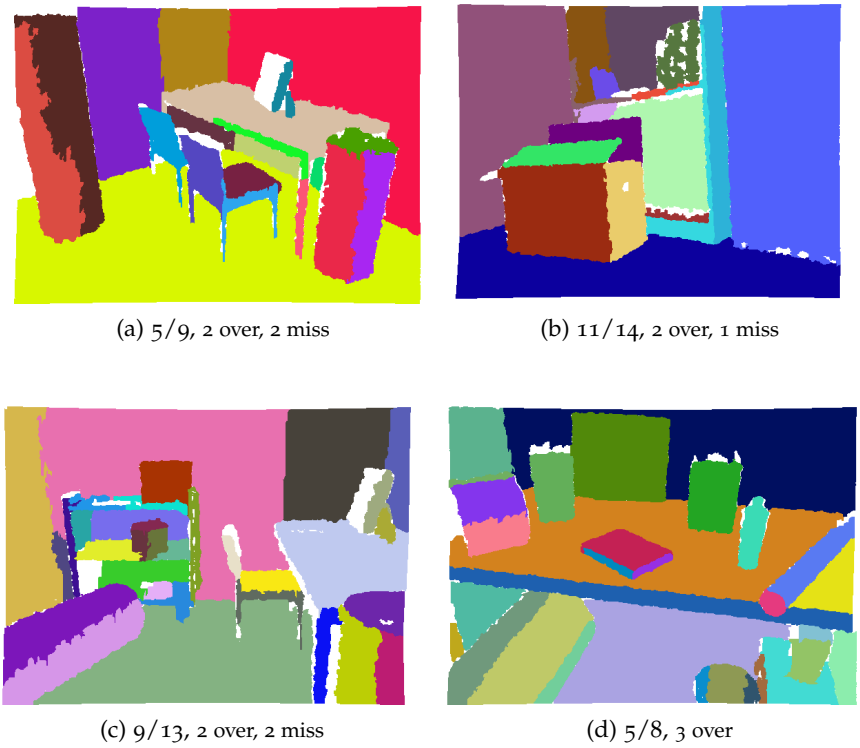
Table 3.2: Runtimes of the individual processing steps for the images from the SegComp ABW and PERCEPTRON datasets (512×512) and Kinect frames with different resolutions: 160×120 (QQVGA), 320×240 (QVGA), and 640×480 (VGA). Runtimes are measured over 10 000 runs on a single core of an Intel Core i7-3740QM CPU @ 2.7 GHz (no parallelization), using adaptive triangulation (Section 3.4.2) and approximate plane segmentation (Section 3.5.4).

3.7.2 *Influence of the Camera Noise Model*

We have conducted a set of experiments to assess the influence of the applied noise model on both approximate meshing and segmentation. Using the Kinect planes dataset by Oehler *et al.* (2011), we evaluated the average plane detection results for the same baseline system as in the runtime evaluation (see Section 3.7.1), but with different noise models.

The dataset contains two different parts together with the corresponding ground truth segmentations—one for plane segmentation and one for cylinder segmentation. The ground truth segmentations only contain the respective primitive type (e.g., plane *or* cylinder) while instances of other primitives in the data is labeled as noise. Naturally, we obtain a larger amount of oversegmentations here, since larger cylinders in the planes part are labeled as noise, while unconnected regions belonging to a single plane are labeled as one segment. However, visually inspecting the segmentation results reveals that all dominant planes are reliably segmented.

Typical plane segmentation results (for the baseline system with model σ_{Holz}) as well as the detailed segmentation results are shown in Figure 3.9 and Table 3.3. Although the final segmentation performance does not considerably deviate for the different noise models under consideration, visual inspection of the triangulation results showed that the quantization-based models (Holzer *et al.* (2012) and Smisek *et al.* (2011)) underestimate the noise in close ranges (leading to oversegmentations) while the fitting-based models (Nguyen *et al.* (2012) and our σ_{FIT}) underestimate at larger distances (causing missed detections). In the dataset this does not further affect the overall detection



(a) 5/9, 2 over, 2 miss

(b) 11/14, 2 over, 1 miss

(c) 9/13, 2 over, 2 miss

(d) 5/8, 3 over

Figure 3.9: Examples for the Kinect planes dataset. The segmented planes are randomly colored. Overall, we obtain clean segmentation results without major failures, e.g., missing larger planes. Despite minor differences, all noise models achieve very similar segmentations and results.

		Segmentation Result				
	Approach	correct	over	under	miss	noise
Lit.	Oehler *et al.* (2011)	4.50 (36.3%)	0.60	0.40	6.80	16.2
Noise model	σ_{FIT}	7.10 (57.2%)	3.63	0.10	1.46	13.2
	σ_{Nguyen}	7.10 (57.2%)	3.63	0.10	1.46	13.2
	σ_{Holzer}	7.20 (58.0%)	3.67	0.10	1.33	13.0
	σ_{Smisek}	7.20 (58.0%)	3.67	0.10	1.33	13.0
	σ_{Holz}	**7.23 (58.3%)**	3.70	0.10	1.27	12.9

Table 3.3: Results for the Kinect Planes dataset as reported by the author of the dataset (Oehler *et al.*, 2011) and obtained using the baseline system and the different noise models. Larger cylinders are segmented into multiple planes. Overall, about 58 % of the 12.4 planes are correctly segmented using the baseline system (assuming 80 % pixel overlap). The performance does not considerably deviate with the different noise models.

results and is only reflected by missing or not missing few very small planes and oversegmenting or not oversegmenting a few very close planes. Combining the two classes in the new noise model with σ_{Holz} yields the best results.

3.7.3 *Plane Segmentation*

In the following, we present our plane detection results for the publicly available SegComp datasets PERCEPTRON and ABW by Hoover *et al.* (1996). Since our noise model σ_{Holz} has been developed for RGB-D cameras, and the Microsoft Kinect in particular, it can only be applied to the datasets recorded by Oehler *et al.*, 2011. In a first evaluation (Holz and Behnke, 2015a), an ad hoc error model was used for the SegComp datasets. It was computed by fitting a quadratic polynomial to the mean square distance of the measurements in the PERCEPTRON and ABW training images to the underlying ground truth planes. The resulting noise model was approximated by:

$$\sigma_{\text{SegComp}}(z) = 0.0036z^2. \tag{3.14}$$

The same noise model was used in the approximate surface reconstruction as the maximum edge length

$$\epsilon_d = 2.5\sigma_z \quad \text{with } \sigma_z = \sigma_{\text{SegComp}}(z) \tag{3.15}$$

for both PERCEPTRON and ABW. In the following comparative evaluation it is referred to the plane segmentation variants using this simple noise model as PPS (Probabilistic Plane Segmentation) and APS (Approximate Plane Segmentation).

For the evaluation in this section, different configurations and parameter setups have been computed and tested per dataset. We will refer to these updated variants as PPS+σ_{ABW} and APS+σ_{ABW} for the ABW dataset, and PPS+σ_{PERC} and APS+σ_{PERC} for the PERCEPTRON dataset. Since the two sensors used for recording the datasets have different measurement and noise characteristics, using different noise models for PERCEPTRON and ABW can yield slightly better results than those reported earlier (Holz and Behnke, 2015a). For completeness, we include both configurations and parameterizations in the following. As a reminder, the noise models are used for both approximate surface reconstruction and inlier detection in the segmentation, i.e., deciding whether or not a point belongs to the currently grown planar region.

In the earlier evaluation (Holz and Behnke, 2015a), we used ϵ_d in Equation (3.15) for both approximate surface reconstruction and planar segmentation. Using the training images, we computed different noise models for ABW and PERCEPTRON as well as different models for the reconstruction and the segmentation. For the reconstruction, we computed a distance-dependent maximum edge length by examining the average distance of points to neighboring points on the same planar

	Reconstruction	Segmentation	
	Edge length	Distance	Normal deviation
ABW	$\epsilon_d(z) = \infty$	$d_{\max}(z) = 0.0042z^2$	$\theta_{\max} = 20°$
PERCEPTRON	$\epsilon_d(z) = 0.008z^2$	$d_{\max}(z) = 0.008z^2$	$\theta_{\max} = 20°$

Table 3.4: Noise models and parameters (SegComp)

segment. For the segmentation, we computed a distance-dependent maximum distance to planar segments by examining the distances of points to their ground truth plane segments. Quadratic functions have been fit to both resulting in the parameter configurations reported in Table 3.4. The ABW dataset, in particular, contains numerous planes whose orientations are almost perpendicular to the line of sight, i.e., the planes are barely visible and points belonging to the same planar segment may have larger distances in between. Consequently, it turned out that using a fully connected mesh, i.e., using edges between all valid measurements, yields the best results.

In addition to the edge length and distance thresholds, we used a maximum angle during region growing to discard points with local surface normals that considerably deviate from the orientation of the grown planar region. The latter, however, is only relevant for the approximate variant (APS) where it forms the primary comparison function to decide whether or not a point is an inlier. The probabilistic region model (PPS) incrementally computes the mean square error of the planar fit and uses this value together with the maximum distance threshold for comparison. For this reason, in the updated configurations we use a maximum normal deviation of $\theta_{\max} = 20°$ only for the approximate variant (APS).

Typical results of applying the presented plane segmentation approaches on range images of the two datasets can be seen in Figures 3.10 and 3.11. Considering our goal of obtaining a fast decomposition into dominant planes and other objects of interest, the obtained results are more than satisfying. Moreover, as it can be seen in the detailed comparisons of Tables 3.5 and 3.6, the proposed approximate plane segmentation method and the probabilistic model achieve state-of-the-art range image segmentation performance, while providing very efficient means to compute—within milliseconds—rough scene segmentations.

On the ABW dataset (Hoover *et al.*, 1996), our approximate plane segmentation approach tends to oversegment the range image. This is caused by a special characteristic of the used camera that is not explicitly handled here. It results in inconsistent normal orientations. Besides oversegmented planar patches, both plane detection models correctly detect more than 80% of the planes. The approach of Georgiev *et al.* (2011) considerably undersegments planes meeting at

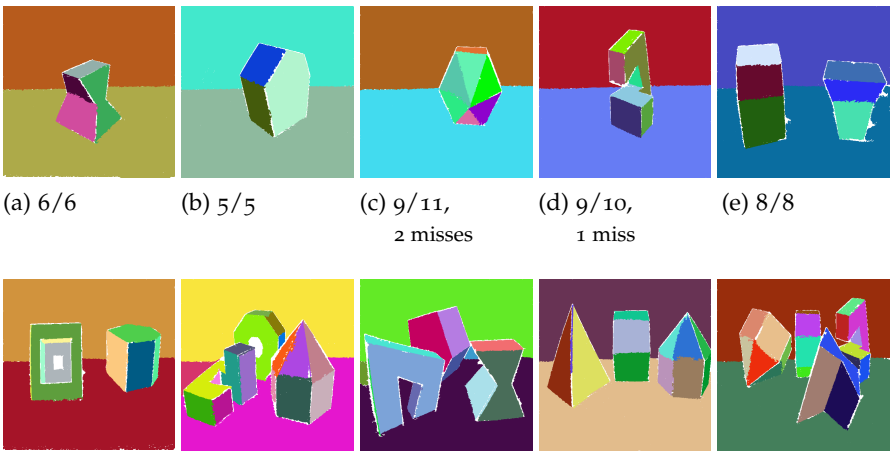
(a) 10/11, 2 over (b) 9/10, 2 over (c) 9/9 (d) 8/8 (e) 8/10, 1 over, 1 miss

Figure 3.10: Examples for the ABW dataset. The segmented planes are randomly colored. Overall, we obtain clean segmentation results without major failures, e.g., missing larger planes.

		Segmentation Result				
	Approach	correct	over	under	miss	noise
SegComp	USF*	12.7 (83.5 %)	0.2	0.1	2.1	1.2
	WSU*	9.7 (63.8 %)	0.5	0.2	4.5	2.2
	UB*	12.8 (84.2 %)	0.5	0.1	1.7	2.1
	UE*	13.4 (88.1%)	0.4	0.2	1.1	0.8
	OU**	9.8 (64.4 %)	0.2	0.4	4.4	3.2
	PPU**	6.8 (44.7 %)	0.1	2.1	3.4	2.0
	UA**	4.9 (32.2 %)	0.3	2.2	3.6	3.2
Literature	Checchin *et al.* (1997)	13.0 (85.5 %)	0.6	0.3	1.0	1.3
	Frigui *et al.* (1999)	13.0 (85.5 %)	0.8	0.1	1.3	2.1
	Jiang (2000)	**13.5 (88.8%)**	0.2	0.0	1.5	0.6
	Koster *et al.* (2000)	13.4 (88.1%)	0.4	0.3	0.8	1.1
	Gotardo *et al.* (2003)	13.0 (85.5 %)	0.5	0.1	1.6	1.4
	Gotardo *et al.* (2004)	13.4 (88.1 %)	0.4	0.1	1.2	1.7
	Oehler *et al.* (2011)	11.1 (73.0 %)	0.2	0.7	2.2	0.8
	Enjarini *et al.* (2012)	13.2 (86.8 %)	0.3	0.2	1.1	1.8
	Feng *et al.* (2014)	12.8 (84.2 %)	0.1	0.0	2.4	0.7
	Husain *et al.* (2015)	10.0 (66.0 %)	0.3	0.8	3.0	1.9
Others	Georgiev *et al.* (2011)	6.9 (45.4 %)	0.6	1.9	3.6	2.1
	Trevor *et al.* (2013)	9.7 (63.8 %)	0.8	0.4	3.9	2.8
	Holz *et al.* (2011)	8.4 (55.1 %)	1.2	0.5	4.2	2.3
Ours	PPS (3.5.3)	12.8 (84.2 %)	0.5	0.1	1.7	2.1
	APS (3.5.4)	12.2 (80.1 %)	1.8	0.1	0.9	1.3
	PPS+σ_{ABW} (3.5.3)	13.2 (86.8 %)	0.3	0.1	1.1	1.8
	APS+σ_{ABW} (3.5.4)	13.0 (85.5 %)	0.4	0.1	1.3	1.2

* As reported by Hoover *et al.* (1996). ** As reported by Jiang *et al.* (2000)

Table 3.5: Results for the ABW dataset as given in literature (top), reproduced using available implementations (middle) and our approach (bottom). Our approach (using approximate plane segmentation tends to oversegment the planes in this dataset. On average, 13.2 and 13.0 planes out of 15.2 planes are correctly segmented.



(a) 6/6 (b) 5/5 (c) 9/11, 2 misses (d) 9/10, 1 miss (e) 8/8

(f) 10/14, 4 misses (g) 18/24, 6 misses (h) 10/13, 2 over, 1 miss (i) 12/13, 1 over (j) 22/22

Figure 3.11: Examples for the PERCEPTRON dataset. The segmented planes are randomly colored. Overall, we obtain clean segmentation results without major failures, e.g., missing larger planes.

		Segmentation Result				
	Approach	correct	over	under	miss	noise
SegComp	USF*	8.9 (60.9 %)	0.4	0.0	5.3	3.6
	WSU*	5.9 (40.4 %)	0.5	0.6	6.7	4.8
	UB*	9.6 (65.7 %)	0.6	0.1	4.2	2.8
	UE*	10.0 (68.4 %)	0.2	0.3	3.8	2.1
Literature	Checchin *et al.* (1997)	10.6 (72.6 %)	0.2	0.6	2.6	2.0
	Frigui *et al.* (1999)	9.6 (65.8 %)	0.7	0.2	3.7	3.6
	Jiang (2000)	10.5 (71.9%)	0.0	0.2	3.6	1.6
	Koster *et al.* (2000)	**11.2 (76.7%)**	0.1	0.2	2.9	5.2
	Gotardo *et al.* (2003)	11.0 (75.3%)	0.3	0.1	3.0	2.5
	Gotardo *et al.* (2004)	10.8 (73.9 %)	0.1	0.1	3.4	2.0
	Oehler *et al.* (2011)	7.4 (50.1 %)	0.3	0.4	6.2	3.9
	Enjarini *et al.* (2012)	10.7 (73.3 %)	0.4	0.1	3.6	4.4
	Feng *et al.* (2014)	8.9 (60.9 %)	0.2	0.2	5.1	2.1
Others	Georgiev *et al.* (2011)	6.5 (44.2 %)	2.3	0.3	5.8	5.0
	Trevor *et al.* (2013)	8.3 (57.0 %)	0.6	0.2	5.2	2.1
	Holz *et al.* (2011)	7.9 (54.1 %)	1.4	0.8	5.9	3.5
Ours	PPS (3.5.3)	11.0 (75.3%)	0.4	0.2	2.7	0.3
	APS (3.5.4)	11.0 (75.3%)	0.4	0.2	2.7	0.3
	PPS+σ_{PERC} (3.5.3)	**11.2 (76.7%)**	0.2	0.2	2.6	0.3
	PPS+σ_{PERC} (3.5.4)	**11.2 (76.7%)**	0.2	0.2	2.6	0.3

* As reported by Hoover *et al.* (1996).

Table 3.6: Results for the PERCEPTRON dataset as given in literature (top), reproduced using available implementations (middle) and our two approaches (bottom). Our approaches achieve state-of-the-art performance by correctly segmenting 76.7 % of the 14.6 planes.

obtuse angles, while the approach of Trevor *et al.* (2013) misses smaller planar patches. We present typical results for the ABW testing dataset using our approximate plane segmentation in Figure 3.10. Detailed results as measured with the SegComp evaluation tool are reported in Table 3.5. Using ABW-specific noise models and parameters considerably improves the segmentation performance. The probabilistic segmentation (PPS) is only slightly better than the considerably faster approximate variant (APS). With the former, we achieve results comparable to Enjarini and Graser (2012). The faster APS can still keep up with the performance of Checchin *et al.* (1997), Frigui and Krishnapuram (1999), and Gotardo *et al.* (2003). Only Jiang (2000), Koster and Spann (2000), and Gotardo *et al.* (2004), achieve better results. From the original SegComp competitors, only the University of Edinburgh (UE) achieves better results than the presented pipeline.

In the PERCEPTRON dataset, no special sensor characteristics cause errors and our approach yields similar results as the work by Koster and Spann (2000) and Gotardo *et al.* (2003). Again, the noise model computed specifically for the PERCEPTRON dataset achieves a considerably better performance. Only Koster and Spann (2000) can achieve the same overall rate of correct detections. However, at the same time Koster and Spann also produce a considerably larger amount of inexistent planes, i.e., noise in the form of smaller planes that are not contained in the ground truth annotations. Compared to other related works and the original entries in the SegComp competition as presented by Hoover *et al.* (1996), we achieve considerably better segmentations.

Muller (2013) proposed another planar segmentation pipeline that combines jump edge detection, region growing based segmentation, graph cut optimization and a subsequent connected component labeling. For the PERCEPTRON dataset, Muller reports an average correct detection of 10.3 % (80 % pixel overlap) as opposed to the 11.2 % achieved using both our probabilistic plane segmentation approach and our approximate plane segmentation as well as the best result found in the literature by Koster and Spann (2000). We excluded the results of Muller from the comparison in Table 3.6 since he does not report detailed results on oversegmentations, undersegmentations. missed planes and false detections.

We present typical results for the PERCEPTRON testing dataset using our approximate plane segmentation (APS) in Figure 3.11. Detailed results measured using the SegComp evaluation tool are reported in Table 3.6. Not correctly found by our approaches are very small plane segments, e.g., the inner parts of the objects in Figure 3.11f and Figure 3.11g. In addition, some planes are oversegmented due to noise, e.g., the support plane in Figure 3.11h.

For both ABW and PERCEPTRON, oversegmentations are rare for our approach. Instead, a considerable number of ground truth planes

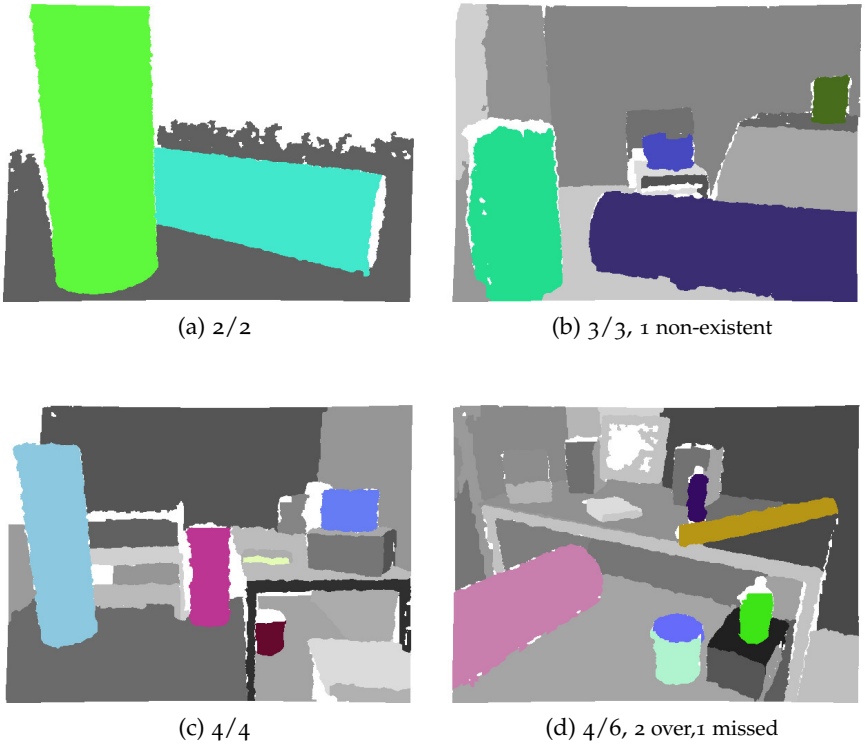
are not perceived. These planes are formed by only a small number of points and are neglected here due to a minimum region cardinality $|R| = 200$ that we use to eliminate outliers and very small planar patches. The approach of Georgiev *et al.* (2011) considerably oversegments in particular smaller planar patches. Insufficient pixel overlap causes a considerable amount of missed planes (especially the dominant support planes) in the approach of Trevor *et al.* (2013).

Our previous approach from Section 2.4.4 (Holz *et al.*, 2011) suffers from the same inconsistent normal orientations in the ABW dataset and tends to oversegment. Furthermore, it does not consider models of the underlying noise, but uses fixed distance-independent thresholds. As a consequence, the algorithm tends to oversegment in both the ABW and the PERCEPTRON dataset. Although it merges neighboring plane clusters to compensate for discretization effects in the histograms, not all oversegmentations are resolved. In addition, since the approach does not consider the neighborhood of a pixel, larger normal deviations in noisy regions can lead to individual pixels not belonging to the same planar segment as its neighbors. Since the evaluation considers an 80% pixel overlap with the ground truth segmentations, the pixels left out cause some planes to be missed. Overall, our previous approach (Holz *et al.*, 2011) achieves similar performance as the algorithms of Georgiev *et al.* (2011) and Trevor *et al.* (2013), but ranks behind state-of-the-art segmentation performance. However, it was the fastest in our experiments (over 15 Hz at VGA resolution), roughly 20 % to 25 % faster than the proposed approach.

3.7.4 *Cylinder Segmentation*

In order to assess the reliability of our approach to detect simple geometric primitives, we have used the Kinect cylinder dataset by Oehler *et al.* (2011). As described in Section 3.5.5, we first segment the recorded RGB-D point clouds into regions of local surface continuity and then fit plane, cylinder, and sphere models to each region and select the model best supporting the underlying point set. Both detected planes and spheres (no spheres were detected in the dataset) are marked as belonging to the background, since the dataset contains only ground truth pixels for cylinders whereas all other pixels are labeled as noise. Typical results of cylinder detection as well as the overall segmentation performance on the Kinect cylinder dataset are shown in Figure 3.12 and Table 3.7.

All larger cylinders in the dataset are reliably detected. Cylinders being sensed from above, i.e., where both outer and inner part are visible, reveal a systematic effect of our approach. In the segmentation of regions with local surface continuity, the two parts are split since the outer part occludes the inner part and is unconnected in the mesh, and the small connected regions to the sides show discontinuities in



(a) 2/2 (b) 3/3, 1 non-existent

(c) 4/4 (d) 4/6, 2 over,1 missed

Figure 3.12: Examples for the Kinect Cylinders dataset. Shown are cylinder detections (random colors) and plane detections (random gray tones). A systematic effect of our segmentation approach can be seen in the oversegmentation in (d) where outer (front) and inner (back) part of the cylinder are separate regions.

	Segmentation Result				
Approach	correct	over	under	miss	noise
Oehler *et al.* (2011)	1.13 (34.2%)	0.007	0	2.100	6.300
Ours (Sec. 3.5.5)	**2.33 (71.4%)**	0.667	0	0.366	0.100

Table 3.7: Results for the Kinect Cylinders dataset. Our approach (first segmenting smooth surfaces and then segmenting planes and other geometric primitives) clearly outperforms the approach of Oehler *et al.* (2011). Oversegmentations are primarily caused by not merging segments belonging to the same cylinder (e.g., inner back and outer front of a single cylinder).

the surface normal orientations. Currently, cylinders found twice are not merged until we replace the regions with cylinder models in the final polygonalization step. This causes, on average, one cylinder to be oversegmented in every second point cloud. In contrast, not a single cylinder was undersegmented. With respect to missed detections, some of the cylinders in the dataset are rather small (radius $< 10\,\text{cm}$) and far away from the sensor (distance $> 2\,\text{m}$). All cylinders missed in the evaluation—on average one in every three point clouds—belong to this class.

A typical problem that arises when not solely segmenting cylinders but all geometric primitives is that larger cylinders are segmented as multiple planes. Especially in larger distances from the sensor, larger distance and normal deviations may be caused by local noise which in turn may cause that a single primitive (e.g., a cylinder) gets segmented into more than one region. As already mentioned, initial segmentations are currently not corrected even if neighboring segments are found to lie on the same primitive, i.e., have the same primitive type and the roughly the same parameters such as radius in case of cylinders. Naturally, parts of larger cylinders can also be well explained by tangential planes which pass through to the centroid of the part. Consequently, especially in larger scenes (with measurements close to the maximally measurable distance) or scenes captured with especially noisy sensors such as ToF cameras, cylinders may get detected as several planes just as with the cylinders in the planar segmentation dataset of Oehler *et al.* (2011). In the experiments in Section 3.7.2, cylinders were not considered and by only detecting planar segments the aforementioned behavior of segmenting cylinders as planes is the default. When segmenting both types of primitives at the same time, this behavior does usually not show up since all relevant parameters (thresholds for distance and normal deviations within a segment, and thresholds for distance and normal deviations within a primitive) are automatically fine-tuned using training data and the noise models of Section 3.6.

3.8 APPLICATION TO STAIR DETECTION

Detecting planes and other geometric primitives has a wide range of applications including, for example, creating two-dimensional floor plans of buildings using planes and edges detected in sequences of RGB-D images (Zhang *et al.*, 2012c) or representing and recognizing objects as compounds of geometric primitives in the context of mobile manipulation (Berner *et al.*, 2013; Holz *et al.*, 2014a). Planar segmentations are especially useful in problems where the object of interest is mainly composed of planes and edges between planes. In the context of the Diploma thesis of Elmasry (2013), we applied the approximate plane segmentation approach to detect the horizontal and vertical

planes which form a staircase. The main idea was to develop a multimodal pipeline using the color (RGB) image, the depth (D) image, and a computed image of surface normals, and to detect both planes and edges. Using the detected planes and edges, the final goal would have been to create a semantic model of the staircase as done by Schmittwilken *et al.* (2007). Whereas the latter could not be achieved during project runtime, the multimodal pipeline was found to reliably detect the planes and edges of sensed staircases. Hence, the pipeline forms a useful preprocessing step in stair detection problems.

In this section, we implemented a simple pipeline for detecting steps of staircases as a possible application of the region-growing-based planar segmentation. It consists of the following steps.

1. Initial Segmentation: computing the approximate surface reconstruction for the input point cloud, smoothing the data using the multilateral filter, and segmenting the filtered mesh using the approximate plane segmentation.

2. Sorting: sorting the detected planes with respect to distance from the sensor and height. For sorting the plane we use IntroSort (Musser, 1997) which has a complexity of $\mathcal{O}(n \log n)$ in the worst and the average case where n is the number of detected planes.

3. Sequencing: splitting the ordered sequence into sequences of ascending and descending neighboring planes and deducing missing planes, e.g., a not detected vertical plane in a sequence of two horizontal planes.

4. Step detection: iterating over the ordered and filled sequences of detected planes and grouping pairs of horizontal and vertical planes into steps.

Since the detected planes are already sorted, the runtime of sequencing and step detection is linear in the number of detected and deduced planes.

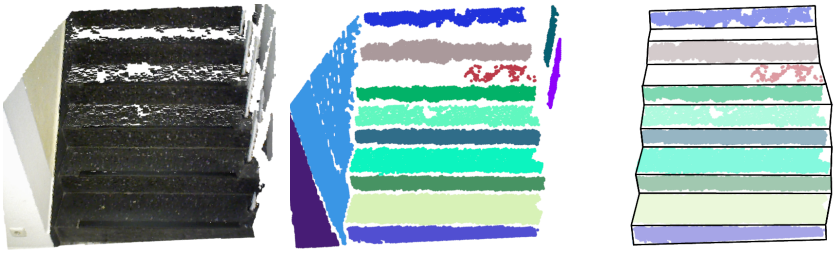
In order to assess the performance of this approach for detecting the planes of staircases, we have recorded a dataset[3] of a staircase in the *LBH* building of the University of Bonn. The dataset was recorded with a hand-held ASUS Xtion PRO RGB-D camera. A total of 165 RGB-D point clouds show parts of ascending stairs, descending stairs or both. We assess the performance in two stages and metrics: 1. success rate of detecting visible planes belonging to the staircase, and 2. success rate of detecting steps of the staircase, i.e., a connected pair of a vertical plane and a horizontal plane. For the latter, we only count steps which are fully visible in the scene, i.e., steps where both the vertical plane and the horizontal plane are contained in the acquired point cloud (or

3 The dataset is available at `http://ais.uni-bonn.de/data/stairs`.

	Planes			Steps	
Type	Detected	Deduced	Total	Detected	Total
Asc.	1569 (91%)	1664 (97%)	1717	785 (97%)	808
Desc.	607 (78%)	607* (78%)	783	611 (78%)	783

*Deduced but not visible vertical planes are excluded.

Table 3.8: Stair detection results: success rates for detecting planes and steps. For ascending stairs, the pipeline detects (and deduces) 97% of the planes and steps. In case of descending stairs the approach fails in regions with only few measurements.

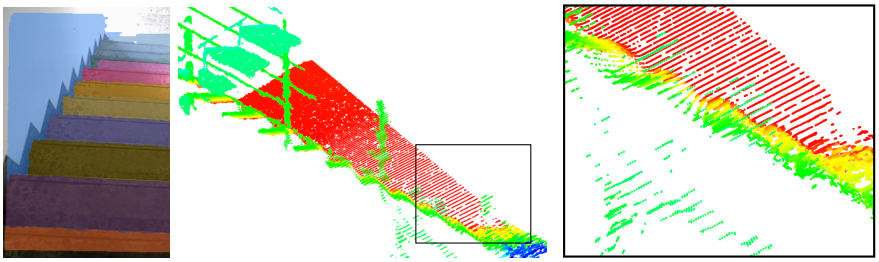


(a) Input point cloud (b) Segmented planes (c) Detected stairs

Figure 3.13: Stair detection results. Shown are a typical input RGB-D point cloud with a staircase, a handrail and an adjacent wall (a), the planes found using approximate plane segmentation (b), and stairs deduced from the detected planes and edges (c). As can be seen, the deduced stairs give a good intuition of the underlying surface and edges but are not consistent over the flight of stairs and need to be corrected, e.g., using a semantic stair model.

can be deduced from the sequence). We report the detailed results in Table 3.8. A typical example of a single scene in the dataset is shown in Figure 3.13 together with the results of detecting planes and steps.

The primary reason for not detecting a plane is sampling not enough points on the plane. In order to avoid false positives in plane detection, we neglect planar regions with less than 50 points. In addition, in larger distances not only the point density decreases but also the probability of not measuring a valid distance. That is, steps and planes farther away from the sensor contain holes of invalid measurements in the data. Moreover, steps may be occluded depending on the slope of the stairs and the orientation of the camera (ascending stairs) or the inclination angle to steps in the height of the sensor (ascending stairs). Consequently, the steps belonging to these planes cannot be adequately detected in case there is no preceding and/or succeeding step in the staircase. In the latter case, the missing part of the step can be deduced from the former and next step (see Figure 3.13). In addition to the number of correctly detected planes, we report the



(a) Planes (b) 3D point cloud (c) Detail view

Figure 3.14: Problem with descending stairs: steps farther away from the sensor are not accurately detected, e.g., the last two steps in (a). These planes do not appear as individual steps and planes in the RGB-D point clouds but as a single curvy surface following the slope of the staircase (b+c).

number of planes deduced, i.e., planes which are visible in the scene but have not been detected. In the case of ascending stairs, horizontal planes are often represented by only few points in the RGB-D point cloud and are rejected in our initial detection. However, two sequential vertical planes allow deducing the missing horizontal plane in between. In case of descending stairs, the vertical planes are not visible. They can be deduced from sequential horizontal planes. Consequently, the steps of descending stairs can be detected by solely detecting the horizontal planes. However, since they are not visible in the scene, the deduced vertical planes in descending stairs have been excluded from our results. Overall, the approach reliably detects the planes and steps closer to the camera and only fails for only partially visible steps farther away from the camera.

Overall, the pipeline of first segmenting the input RGB-D point clouds into planes and then deducing sequences of sequential planes, missing planes, and steps has been found to reliably detect steps for both ascending and descending staircases. The approach only fails in regions farther away from the sensor where the quality of distance measurements does not allow detecting planes in the 3D points. In these regions, especially the quantization effects and the local measurement noise make sequences of planes appear as a single curvy surface instead of individual plane segments. Consequently, the planes in these regions are either detected as a single larger plane (see Figure 3.14) or not detected at all. It remains a matter of future work, to use these initial per-frame segmentations in order to build a global representation of the staircase and to derive a parameterized model, e.g., as proposed by Schmittwilken *et al.* (2007).

3.9 CONCLUSIONS

We have presented a fast yet robust approach for segmenting range images and organized point clouds. Using an approximate polygonal mesh reconstruction directly deduced from the image-like structure, we are able to efficiently compute point features such as local surface normals, smooth the measured data using a multilateral filter, and detect planes and other geometric primitives.

Experimental evaluation has shown that our approach achieves state-of-the-art range image segmentation performance. To achieve real-time processing, we proposed several simplifications and approximations that make the overall segmentation (and primitive detection) algorithm run within milliseconds on a CPU for point clouds acquired by typical RGB-D cameras. Further speeding up the approach by parallelizing individual components is expected to considerably increase efficiency.

Finally, we have presented a potential application where the planar segmentation is used to detected steps and staircases. Experiments have shown reliable detections and that the pipeline is able to deduce planes that have not been detected in the initial segmentation (e.g., not visible vertical planes in descending staircases). Possible applications of such an initial detection are semantic modeling of staircases (Schmittwilken *et al.*, 2007) or planning the motions of an autonomous mobile robot to climb staircases (Osswald *et al.*, 2011).

It remains a matter of future work to exploit the extracted segmented planar patches (and geometric primitives) in further processing steps for purposes such as registration. Potential applications of this segmentation pipeline include extracting dominant planes and environmental structures, e.g., for automatically generating floor plans from sequences of RGB-D images as is done by Zhang *et al.* (2012b). The implementations of most of the components presented in this chapter are publicly available within the open source Point Cloud Library (PCL)[4].

Since the approach is both robust and efficient, it has already been used in many other works. Most recently, Fernández-Moral *et al.* (2016) used the pipeline to extract planar patches from RGB-D images. They represent the scene using graphs of connected planes, and register the images using graph matching.

4 The latest stable release of PCL is available at `http://pointclouds.org`.

Part II

REGISTRATION AND MAPPING

... in which sensor readings acquired at different view points are accurately aligned to build consistent maps of the environemnt.

3D Registration is the problem of consistently aligning two or more point clouds, i.e., sets of three-dimensional points. Often point clouds are acquired using 3D sensors such as 3D laser scanners or 3D cameras. In order to sense all parts of an object or an environment, multiple point clouds need to be acquired from different viewpoints. It is the task of registration to find the relative pose (position and orientation) between the acquired views in a global coordinate frame, such that the overlapping areas between the point clouds match as well as possible. After registration, the aligned point clouds can be fused into to a single point cloud so that subsequent processing steps like object modeling can be applied.

Simultaneous localization and mapping (SLAM) refers to the problem of building a model of an object or an environment while localizing the sensor in the so far built model. It involves the registration of all acquired point clouds and often comprises several processing steps including a rough initial alignment and an accurate refinement in order to build globally consistent models.

This part of the thesis focuses on registration and SLAM using two different types of 3D sensors: custom light-weight 3D laser scanners carried by micro aerial vehicles and consumer color and depth cameras such as the popular Microsoft Kinect™ camera.

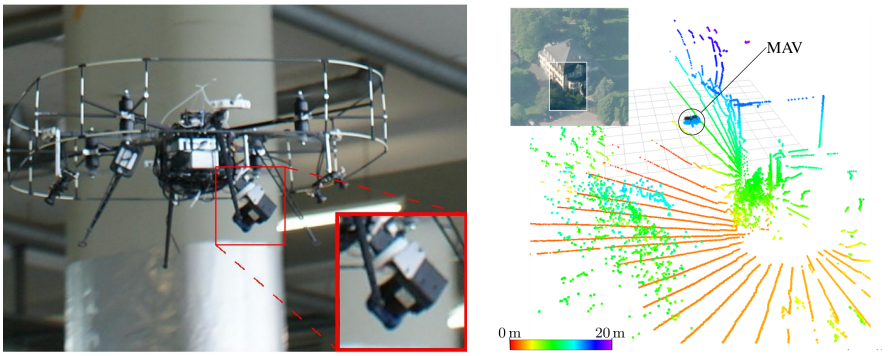
4 REGISTRATION AND MAPPING WITH MICRO AERIAL VEHICLES

Micro aerial vehicles (MAVs) pose specific constraints on onboard sensing, mainly limited payload and limited processing power. For accurate three-dimensional (3D) mapping even in GPS-denied environments, we have designed a lightweight 3D laser scanner specifically for the application on MAVs. Similar to other custom-built 3D laser scanners composed of a rotating two-dimensional (2D) laser range finder, it exhibits different point densities within and between individual scan lines. When rotated fast, such non-uniform point densities influence neighborhood searches which in turn may negatively affect local feature estimation and scan registration. In this chapter a complete pipeline is presented that allows for 3D mapping including pair-wise registration and global alignment of such non-uniform density 3D point clouds acquired in-flight. For registration, a state-of-the-art registration algorithm is extended to include topological information from approximate surface reconstructions. For global alignment, a graph-based approach is applied that makes use of the same error metric and iteratively refines the complete vehicle trajectory. In experiments, the proposed approach can compensate for the effects caused by different point densities up to very low angular resolutions. It allows for building accurate and consistent 3D maps in-flight with a micro aerial vehicle.

4.1 INTRODUCTION

MAVs such as quadrotors have attracted much attention in the field of aerial robotics in recent years. Their size and weight limitations, however, pose a problem in designing sensory systems for environment perception. Most of today's MAVs are equipped with ultrasonic sensors and camera systems due to their minimal size and weight. While these small and lightweight sensors provide valuable information, they suffer from a limited field-of-view. Furthermore, cameras are sensitive to illumination conditions. Only few MAVs (Tomić *et al.*, 2012; Grzonka *et al.*, 2009; Bachrach *et al.*, 2009; Shen *et al.*, 2011) are equipped with 2D laser range finders (LRFs) that are used for navigation. These provide accurate distance measurements to the surroundings but are limited to the two-dimensional scanning plane of the sensor. Objects below or above that plane are not perceived.

3D laser scanners provide robots with the ability to extract spatial information about their surroundings, detect obstacles in all directions, build 3D maps, and localize. In the course of a larger project



(a) Flying micro aerial vehicle with scanner (b) Example of an acquired scan

Figure 4.1: Sensor setup and example scan: the laser scanner is mounted slightly below the MAV facing forwards. Continuously rotating it allows an almost omnidirectional perception of its surroundings. The resulting 3D scans (aggregated over one half rotation using visual odometry) show different point densities within and between individual scan lines.

on mapping inaccessible areas with autonomous micro aerial vehicles, we have developed a lightweight 3D scanner (Droeschel *et al.*, 2014a) specifically suited for the application on MAVs. It consists of a Hokuyo 2D laser range scanner, a rotary actuator and a slip ring to allow continuous rotation. Just as with other rotated scanners, the acquired point clouds (aggregated over a half rotation of the scanner) show the particular characteristic of having non-uniform point densities: usually a high density within each scan line and a larger angle between scan lines (see Figure 4.1). Since we use the laser scanner for omnidirectional obstacle detection and collision avoidance, we rotate it quickly with 1 Hz, resulting in a particularly low angular resolution of roughly 9° to 10°. The resulting non-uniform point densities affect neighborhood searches and cause problems in local feature estimation and registration when keeping track of the MAV movement and building allocentric 3D maps by means of simultaneous localization and mapping (SLAM).

In this paper, we present a complete processing pipeline for building globally consistent 3D maps with this sensor on a flying MAV. To compensate for the non-uniform point densities, we approximate the underlying measured surface and use this information in both initial pairwise registration of consecutive 3D scans to track the MAV movement and graph-based optimization for building a consistent and accurate 3D map. For initial registration, we extend the state-of-the-art registration algorithm Generalized-ICP (GICP) by Segal *et al.* (2009) to include topological surface information instead of a point's 3D neighborhood. We represent the resulting trajectory in a pose graph (Kümmerle *et al.*, 2011) and connect neighboring poses by edges representing point-pair correspondences between scans and encoding

the same error metric using topological surface information. This graph is iteratively refined, re-estimating the point correspondences in each iteration, to build a consistent 3D map.

This chapter is structured as follows. After a discussion of related work in Section 4.2, we present our registration approach including the approximate surface reconstruction and the approximate feature estimation in Section 4.3. In Section 4.4, we discuss the extension to a complete SLAM system: we extend the registration approach to also estimate the pose uncertainty and use this information for graph-based SLAM (single edge between connected nodes) as a baseline system. We then introduce our SLAM approach using multiple edges per connection where every edge encodes a point-to-point correspondence (in terms of the GICP error metric). In Section 4.5, we present the results of a thorough experimental evaluation of both the plain registration approach and the two SLAM variants. Finally, we summarize the main conclusions and discuss future work in Section 4.6.

The registration algorithm has been published and presented at the International Symposium on Robotics (Holz and Behnke, 2014b). The extension to a complete SLAM system including the registration algorithm was published and presented at the International Conference on Intelligent and Autonomous Systems (Holz and Behnke, 2014c). An extended version of these two papers (and an early version of this chapter, respectively) was published in *Robotics and Autonomous Systems* (Holz and Behnke, 2015a).

4.2 RELATED WORK

Particularly important for the autonomous application of MAVs is the ability to perceive and avoid obstacles. Building environment maps is necessary for goal-directed navigation planning and executing the planned trajectories.

In the following, we discuss related works with a focus on 1) perception, 2) registration and 3) mapping. The former two allow sensing environmental structures, keeping track of the motion of the MAV, and aggregating measurements in local egocentric maps in order to be able to reliably avoid collisions. The latter aims for building allocentric 3D environments for being able to plan paths and missions.

4.2.1 *Perception and Mapping with Micro Aerial Vehicles*

Scaramuzza *et al.* (2014) present vision-based perception, control and mapping for a swarm of MAVs. In contrast to our work, 3D mapping is done on a ground station gathering visual keypoints from all MAVs, and dense 3D maps are reconstructed from the final trajectories offline. Moreover, the approach is purely vision-based and restricted to downward-facing cameras whereas our approach aims at omnidirec-

tional perception thereby allowing to map environmental structures that are not below the MAV.

For mobile ground robots, 3D laser scanning sensors are widely used due to their accurate distance measurements even in bad lighting conditions and their large field-of-view. For instance, autonomous cars often perceive obstacles by means of a rotating laser scanner with a 360° horizontal field-of-view, allowing for the detection of obstacles in every direction (Urmson *et al.*, 2008; Montemerlo *et al.*, 2008).

Up to now, such 3D laser scanners are rarely used on lightweight MAVs—due to payload limitations.

Instead, two-dimensional laser range finders are often used (Tomić *et al.*, 2012; Grzonka *et al.*, 2009; Bachrach *et al.*, 2009; Shen *et al.*, 2011; Grzonka *et al.*, 2012; Huh *et al.*, 2013). Using a statically mounted 2D laser range finder restricts the field-of-view to the two-dimensional measurement plane of the sensor. This poses a problem especially for reliably perceiving obstacles surrounding the MAV. When moving however, and in combination with accurate pose estimation, these sensors can very well be used to build 3D maps of the measured surfaces. Fossel *et al.* (2013), for example, use Hector SLAM (Kohlbrecher *et al.*, 2011) for registering horizontal 2D laser scans and OctoMap (Hornung *et al.*, 2013) to build a three-dimensional occupancy model of the environment at the measured heights.

Morris *et al.* (2010) follow a similar approach and in addition use visual features to aid motion and pose estimation. Still, perceived information about environmental structures is constrained to lie on the 2D measurement planes of the moved scanner. In contrast, we use a continuously rotating laser range finder that does not only allow capturing 3D measurements without moving, but also provides omnidirectional sensing at comparably high frame rates (2 Hz in our setup by aggregating scans over one half rotation).

A similar sensor is described by Scherer *et al.* (2012) and Cover *et al.* (2013). Their MAV is used to autonomously explore rivers using visual localization and laser-based 3D obstacle perception. In contrast to their work, we use the 3D laser scanner for both omnidirectional obstacle perception and mapping the environment in 3D.

For building maps with a hand-held rotating 2D laser range finder, Zhang and Singh (2014) compute edge points and planar points in the acquired range scans. They split the SLAM task in two problems: matching range scans to obtain motion estimates at a high frame rate and accurate registration for mapping at a lower frame rate. The method produces accurate 3D maps of smaller environments but does not detect loop closures, i.e., entering previously mapped regions.

4.2.2 *3D Scan Registration*

The fundamental problem in 3D map building is registration in order to align the acquired 3D laser scans and estimate the poses (positions and orientations) where the scans have been acquired. Over the past two decades, many different registration algorithms have been proposed. Prominent examples for estimating the motion of mobile ground robots using 3D scan registration are the works of Segal *et al.* (2009), Nüchter *et al.* (2005), and Magnusson *et al.* (2007).

3D laser scanners built out of an actuated 2D laser range finder are usually (especially on ground robots) rotated comparably slower than ours to gain a higher and more uniform density of points. Most of the approaches to register such scans are derived from the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992). It iteratively searches for corresponding points between to point clouds and computes a transformation that minimizes the point-to-point distances between the found matches. In contrast, approaches based on the normal distributions transform (NDT) algorithm by Biber and Straßer (2003) model local statistics in the neighborhoods of the points and align these in a surface-to-surface alignment. Magnusson *et al.* (2007) extend this approach to 3D. Generalized-ICP (GICP) algorithm (Segal *et al.*, 2009) unifies the ICP formulation for various error metrics such as point-to-point, point-to-plane, and plane-to-plane. The effect of using this generalized error metric is that corresponding points in two 3D laser scans are not directly dragged onto another, but onto the underlying surfaces. For our non-uniform density point clouds, however, GICP tends to fail since the local neighborhoods of points do not adequately represent the underlying surface. We adapt the GICP approach here to use extracted information from approximate surface reconstructions in the acquired 3D scans.

Our approach explicitly addresses the non-uniform point densities and tries to compensate for the resulting effects by using the approximated surface information. An alternative for using such sparse data in registration and mapping is to aggregate the point clouds in local maps and thereby increase the point density as is done in another work (Droeschel *et al.*, 2014c) within the same project on MAV-based mapping as the work at hand. Both ways constitute problems in their own right.

Bosse *et al.* (2012) use a spring to passively articulate a 2D laser range finder and present a registration algorithm for building accurate 3D point cloud maps. Due to the passivity of the spring-based articulation, however, their sensor setup cannot guarantee complete omnidirectional point clouds at fixed controllable intervals as is the case for a continuously rotating scanner. Furthermore, it requires the carrying vehicle to move in order to induce oscillation. For registration, Bosse *et al.* use a surfel-based approach and efficiently solve both

aggregating point clouds and building globally consistent 3D maps. Since surfels are computed on local neighborhoods, the approach may suffer from the same degradation effects as GICP when applied to the non-uniform density data of our sensor setup.

4.2.3 *Multi-View Scan Registration and SLAM*

Simultaneous localization and mapping (SLAM) is a key problem in mobile robotics. Registering pairs of consecutive laser scans on its own can only provide estimates about the movement in between the poses where the scans have been acquired but cannot be used for building consistent maps due to inaccuracies and drift (when propagating estimated movements over registrations). Instead, pure pairwise registration algorithms are usually used in the front-end of SLAM systems to obtain a rough initial vehicle trajectory and to detect loop closures, i.e., regions where the robot has been before.

For globally aligning all acquired scans and building a consistent map, the registration problem is usually formulated in terms of a graph where poses or landmark positions form the vertices, and view or movement constraints form the edges. A standard approach is to encode relative pose estimates between connected view poses (vertices) in a single edge, e.g., a homogeneous transformation matrix, together with a covariance estimate. We present such a system making use of the proposed registration algorithm as a baseline system for comparison in Section 4.4.2. For optimizing a graph of poses with initial estimates many different approaches have been proposed (Frese *et al.*, 2005; Olson *et al.*, 2006; Grisetti *et al.*, 2010; Grisetti *et al.*, 2009). For a survey on different mapping problems and their relation to graph-based SLAM, we refer the interested reader to the survey of Agarwal *et al.* (2014). The difficulty in our case is that our laser scans are particularly sparse. Consequently, our estimated transformations are accurate but not as accurate as each individual laser measurement (see the results of our experimental evaluation of pairwise registration in Section 4.5.1).

As a second mean for compensating for the non-uniform densities in our scans, we do not use a single edge between 3D scans to encode their relative position but estimate point correspondences in between the scans and iteratively refine the resulting system. For each correspondence, we add an edge to the graph that follows the same error metric as our registration algorithm—again using the information extracted from approximate surface reconstruction. To optimize the resulting graph, we use g2o (Kümmerle *et al.*, 2011), a state-of-the-art open-source graph optimization framework. In a final optional processing step, we build a 3D map with the optimized poses using OctoMap (Hornung *et al.*, 2013) for being able to plan paths in future missions of the MAV.

In multi-view scan matching, multiple poses from which scans have been taken are determined simultaneously by aligning all scans in a single error function or optimization framework. In the 2D domain, a popular multi-view scan registration approach is the algorithm proposed by Lu and Milios (Lu and Milios, 1997) which is often referred to as LUM. Borrmann *et al.* extend this approach to six degrees of freedom for the alignment of 3D scans and present methods to efficiently deal with the resulting nonlinearities (Borrmann *et al.*, 2008). Several further extensions and optimizations have been proposed by the same and other authors since then. The resulting SLAM approach first applies the ICP algorithm to align consecutive point clouds and then builds a graph based on the determined connectivity of view poses similar to our approach. Both the determined relative transformations between view poses and the sets of point correspondences are represented in the edges. From both transformation and correspondences a measurement vector and its covariance matrix are computed which are then fed as one block into a large linear system. The linear system is then solved for the optimal relative transformations and view poses. In contrast, in the proposed multi-edge approach, every correspondence pair forms a block in the final non-linear error function. Its simplification is thereby left to g2o, e.g., using sparse Cholesky decomposition. Furthermore, LUM uses a point-to-point error metric as in the original ICP algorithm which conflicts with the particularly sparse nature of our point clouds. Instead, we approximate the underlying surface and use a probabilistic surface-to-surface error metric in both the initial pairwise registration and the point correspondence edges of the graph.

4.2.4 *Landmark-based SLAM*

Using multiple edges constraining the relative transformation between two view poses also forms the underlying idea of landmark-based SLAM and bundle adjustment. In landmark-based SLAM, features are extracted from the data acquired by the moving sensor and used as landmarks in the graph. In order to form constraints in the graph of poses and landmarks, the extracted features are matched. Matching is usually performed in a higher-dimensional descriptor space to ease the involved data association problem. Prominent examples include using 3D features such as FPFH (fast point feature histogram, Rusu *et al.*, 2009a) or appearance-based features such as SIFT (scale-invariant feature transform, Lowe, 2004), SURF (speeded-up robust features, Bay *et al.*, 2008), BRIEF (binary robust independent elementary features, Calonder *et al.*, 2010) or ORB (Oriented FAST and Rotated BRIEF, Rublee *et al.*, 2011; Rosten and Drummond, 2006; Rosten *et al.*, 2010; Calonder *et al.*, 2010) as compared in the evaluation of Endres *et al.* (2012a). Repeatable features are not easily extractable from our 3D laser scans, especially since the different scans are likely to not include the same

parts of environmental structures due to the low angular resolution between individual scan lines. Instead, our matching is purely based on proximity of the raw points. Due to the low angular resolution, it is very likely that none of the matching pairs is formed by two measurements of the same point, but by measuring two points in close vicinity, possibly on different surfaces. By using a robust surface-to-surface error metric, we compensate for this inaccuracy. To compensate for false correspondences in the initial matching steps, we iteratively refine both transformation and matches in the initial registration and the global alignment with a decreasing distance threshold in an ICP-like fashion. Hence, our approach can be categorized as being somewhere between multi-view scan matching and landmark-based SLAM.

Similar to our multi-edge global alignment step is the approach of Ruhnke *et al.* (2012). They also use raw point matches as constraints in the graph and apply a surfel-based error metric to iteratively refine both the sensor poses and the positions of the points. Their approach can build highly accurate object models but requires a rough initial alignment of the dense RGB-D data. In contrast, we present a complete pipeline that is tailored for the challenging non-uniform density point clouds instead of dense RGB-D data and that can cope with unavailable and erroneous initial pose estimates. Both the initial registration and the multi-edge global alignment make use of approximate surface reconstructions and the same surface-to-surface error metric.

Recently, Zlot and Bosse (2014) presented a 3D mapping system for mines that uses a continuously spinning SICK scanner. They use non-rigid surfel registration and graph optimization for aggregating point clouds and building consistent maps. Compared to our work, their scanner is rotated slower, equipped with an accurate inertial measurement unit, and mounted on a slowly driving truck. Moreover, the aggregation is performed in larger local windows to increase the density of the data in a similar fashion as Droeschel *et al.* (2014c) who build local egocentric maps. Instead, we address the problem of registration and mapping directly using the sparse non-uniform density point clouds.

4.3 REGISTRATION OF SPARSE LASER SCANS

Under the assumption of good motion estimates (e.g., GPS, visual odometry, or inertial measurement units) at least over short periods of time, acquired range scans can be aggregated to form locally consistent 3D point clouds. Throughout this paper, we will assume such an estimate as given (robust visual odometry) and process point clouds aggregated over one half rotation of the laser range scanner. For details on scan aggregation, initial motion estimate and sensor characteristics, we refer to the detailed description in (Droeschel *et al.*, 2014a).

In contrast to the motion estimate being reliable over short periods of time for aggregating point clouds, we do not assume a good pose estimate between aggregated point clouds and good motion estimates over longer periods of time in general. Instead, we design our approach to be robust against noisy, erroneous and no initial pose estimates so as to estimate the motion of the MAV between the acquisition of aggregated point clouds. This allows mapping and localization purely based on point cloud registration even in case of sensor outages and other localization errors.

4.3.1 *Registration of 3D Point Clouds*

A point cloud is a data structure P used to represent a collection of multi-dimensional points $\boldsymbol{p} \in P$. In a 3D point cloud, the elements usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface. When more information is available (such as color information) or information about local surface normal $\boldsymbol{n}$ or curvature κ, the points $\boldsymbol{p} \in P$ become n-dimensional.

Given a *source* point cloud P with points $\boldsymbol{p} \in P$, and a *target* point cloud Q with points $\boldsymbol{q} \in Q$, the problem of registration is to find *correspondences* between P and Q, and estimate a transformation $\boldsymbol{T}$ that, when applied to P, aligns all pairs of corresponding points $(\boldsymbol{p}_i \in P, \boldsymbol{q}_j \in Q)$. One fundamental problem of registration is that these correspondences are usually not known and need to be determined by the registration algorithm.

Iterative registration algorithms align pairs of 3D point clouds by alternately searching for correspondences between the clouds and minimizing the distances between matches. A standard algorithm is the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992). In order to align a point cloud P with a point cloud Q, it searches for closest neighbors in Q for points $\boldsymbol{p}_i \in P$ and minimizes the point-to-point distances $\boldsymbol{d}_{ij}^{(\boldsymbol{T})} = \boldsymbol{q}_j - \boldsymbol{T}\boldsymbol{p}_i$ of the set of found correspondences $\mathcal{C}$ in order to find the optimal transformation $\boldsymbol{T}^\star$:

$$\boldsymbol{T}^\star = \arg\min_{\boldsymbol{T}} \sum_{(ij)\in\mathcal{C}} \|\boldsymbol{d}_{ij}^{(\boldsymbol{T})}\|^2. \tag{4.1}$$

As a result, points in P are dragged onto their corresponding points in Q. Assuming (predominantly) correct correspondences, the ICP algorithm can reliably register regular uniform density point clouds (if the initial alignment is not considerably off). In case of our non-uniform density point clouds, closest points do not correspond to the same physical point in the measured environment. Consequently, the point-to-point error metric leads to dragging the high-density 2D scan lines onto another instead of correctly aligning sensed environmental structures.

4.3.2 *Generalized Iterative Point Cloud Registration*

A particularly robust variant is the Generalized-ICP (GICP) algorithm proposed by Segal *et al.* (2009) which generalizes over the different available error metrics (point-to-point, point-to-plane, plane-to-plane) and thus takes into account information about the underlying surface. Instead of minimizing the distances $\boldsymbol{d}_{ij}^{(\boldsymbol{T})}$ between corresponding points $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ as in the ICP algorithm, it inspects the distribution

$$\boldsymbol{d}_{ij}^{(\boldsymbol{T})} \sim \mathcal{N}\left(\boldsymbol{q}_j - \boldsymbol{T}\boldsymbol{p}_i,\ \boldsymbol{\Sigma}_j^Q + \boldsymbol{R}\boldsymbol{\Sigma}_i^P\boldsymbol{R}^T\right) \tag{4.2}$$

where $\boldsymbol{R}$ is the rotation matrix of $\boldsymbol{T}$. The underlying assumption is that both points in P and points in Q are drawn from independent normal distributions, i.e., $\boldsymbol{p}_i \sim \mathcal{N}(\hat{\boldsymbol{p}}_i, \boldsymbol{\Sigma}_i^P)$ and $\boldsymbol{q}_j \sim \mathcal{N}(\hat{\boldsymbol{q}}_j, \boldsymbol{\Sigma}_j^Q)$. The optimal transformation $\boldsymbol{T}^\star$ best aligning P to Q can be found using maximum likelihood estimation (MLE):

$$\boldsymbol{T}^\star = \arg\max_{\boldsymbol{T}} \prod_{ij\in\mathcal{C}} p\left(d_{ij}^{(\boldsymbol{T})}\right) = \arg\max_{\boldsymbol{T}} \sum_{ij\in\mathcal{C}} \log\left(p\left(d_{ij}^{(\boldsymbol{T})}\right)\right) \tag{4.3}$$

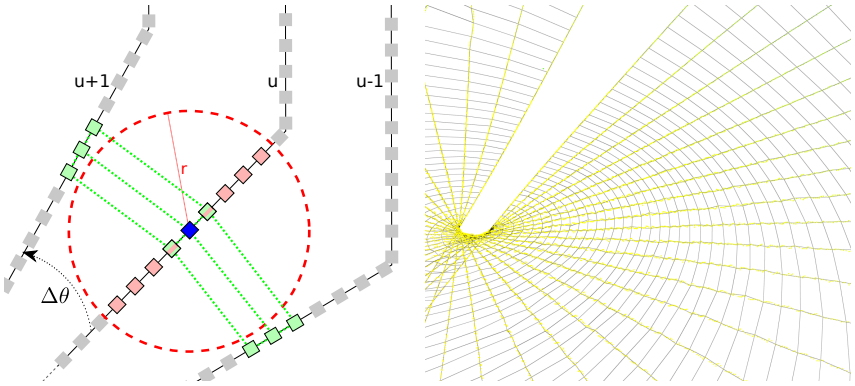
$$\simeq \arg\min_{\boldsymbol{T}} \underbrace{\sum_{ij\in\mathcal{C}} {d_{ij}^{(\boldsymbol{T})}}^T \left(\boldsymbol{\Sigma}_j^Q + \boldsymbol{R}\boldsymbol{\Sigma}_i^P\boldsymbol{R}^T\right)^{-1} d_{ij}^{(\boldsymbol{T})}}_{\text{= simplified likelihood } L(\boldsymbol{T})}. \tag{4.4}$$

The effect of minimizing Equation (4.4) is that corresponding points are not directly dragged onto another, but the underlying surfaces represented by the local covariance matrices $\boldsymbol{\Sigma}_i^P$ and $\boldsymbol{\Sigma}_j^Q$. The covariance matrices are computed so that they express the expected uncertainty along the local surface normals at the points. Consequently, the convergence of GICP degrades with inaccurate estimates of the covariances with regular neighborhood searches as illustrated in Figure 4.2a. If the neighborhood radius is too small, the covariance only reflects a single scan line and not the surface. If it is too large, the covariance can become inaccurate compared to the underlying surface.

At the heart of our approach is the idea to approximate the surfaces in the point clouds in order to compensate for the non-uniform point densities and to compute accurate covariances that better reflect the underlying surfaces.

4.3.3 *Approximate Surface Reconstruction*

In order to get a better estimate of the underlying covariances, we perform an approximate surface reconstruction as done in Chapter 3 in the context of range image segmentation (Holz and Behnke, 2014a). We traverse an organized point cloud P once and build a simple quad mesh by connecting every point $\boldsymbol{p} = P(u, v)$ (v-th point in the u-th scan line) to its neighbors $P(u, v+1)$, $P(u+1, v+1)$, and $P(u+1, v)$ in the same and the subsequent scan line (see Figure 4.2). We only add



(a) Topology visualization (b) Example reconstruction

Figure 4.2: Measurement topology and neighborhood searches: (a) Classic neighbor searches in non-uniform density clouds may only find points in the same scan line (red), whereas a topological neighborhood (green) may better reflect the underlying surface. (b) Example of an approximate surface reconstruction: edges in the quad mesh connect neighboring points in the same scan and between neighboring scans. Points in the first and last scan of a half rotation are not connected.

a new quad to the mesh if $P(u, v)$ and its three neighbors are valid measurements, and if all connecting edges between the points are not occluded. The first check accounts for possibly missing or invalid measurements in the organized structure. The second check avoids that neighboring points which have been measured on different surfaces are not connected in the resulting quad mesh. If all checks pass, we add a new quad to the incrementally built mesh representation. Otherwise, holes arise. After construction, we simplify the resulting mesh by removing all vertices that are not used in any quad (see Section 3.4 for more details on the approximate surface reconstruction). A typical result of applying our approximate surface reconstruction to a 3D scan acquired by our MAV is shown in Figure 4.2b.

For the sparse point clouds acquired by the MAV the occlusion check in Equation (3.1) is, however, inaccurate since the larger angle $\Delta\theta$ between scan lines causes that occluding edges may not get scanned as in dense point clouds. That is, the scanner may sample the surfaces farther away from the occluding boundaries. Hence, it is often not possible to directly deduce an occlusion from the raw measurements. Instead, reformulate the edge validity check to only limit the maximum length of edges in the mesh:

$$valid = \left(d_{i,j} \leq \epsilon_d^2\right), \tag{4.5}$$

$$\text{with} \quad d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|^2, \tag{4.6}$$

We adapt the threshold ϵ_d for the maximum edge length to capture the expected distance between neighboring points on the same surface and the expected angular resolution within and between scan lines:

$$\epsilon_d(d_i) = \begin{cases} \sqrt{2}\, d_i\, \tan\Delta\theta & \text{between scan lines, and} \\ \sqrt{2}\, d_i\, \tan\Delta\phi & \text{within scan lines.} \end{cases} \tag{4.7}$$

The threshold $\epsilon_d(d_i)$ depends on the measured distance to point $\boldsymbol{p}_i$, i.e., $d_i = \|\boldsymbol{p}_i - \boldsymbol{v}\|$ where $\boldsymbol{v}$ is the viewpoint from where the measurements has been acquired. It is computed for every point. For neighboring points within a scan line, we use a different threshold that corresponds to the angular resolution $\Delta\phi$ of the range scanner (and taking into account subsampling if applied).

4.3.4 *Approximate Covariance Estimates*

To estimate the covariance matrix of a point, we directly extract its local neighborhood from the topology in the mesh instead of searching for neighbors. Depending on the desired smoothing level (usually controlled with the search radius), we can extend the neighborhood of a point to include the neighbors of neighbors and ring neighborhoods farther away from the point.

Instead of computing the empirical covariances as done by Segal *et al.* (2009), we approximate them using the local surface normals. We first compute the normal $\boldsymbol{n}_i$ for a point $\boldsymbol{p}_i$ directly on the mesh as the weighted average of the plane normals of the N_T faces surrounding the point $\boldsymbol{p}_i$:

$$\boldsymbol{n}_i = \frac{\sum_{j=0}^{N_T} (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,b}) \times (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,c})}{\| \sum_{j=0}^{N_T} (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,b}) \times (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,c}) \|}, \tag{4.8}$$

with face vertices $\boldsymbol{p}_{j,a}$, $\boldsymbol{p}_{j,b}$ and $\boldsymbol{p}_{j,c}$ (the same method is used in Section 3.4.3). In the actual implementation, we first compute the face normal (unnormalized cross product) for each polygons and add them to the point normals of the vertices spanning the polygon. Afterwards, all point normals are normalized. This automatically gives a higher influence to larger (and thus more stable) polygons on the computation of the point normals. That is, point normals are computed as a weighted sum with weights being proportional to the ares of the connected polygons.

After computing the point normals, the local covariance matrices $\boldsymbol{\Sigma}_i^A$ and $\boldsymbol{\Sigma}_i^B$ are computed as initially proposed by Segal *et al.* (2009):

$$\boldsymbol{\Sigma}_i^P = \boldsymbol{R}_{\boldsymbol{n}_i}^P \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{R}_{\boldsymbol{n}_i}^{P\,T}, \quad \text{and} \quad \boldsymbol{\Sigma}_j^Q = \boldsymbol{R}_{\boldsymbol{n}_j}^Q \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{R}_{\boldsymbol{n}_j}^{Q\,T} \tag{4.9}$$

with rotation matrices $\boldsymbol{R}_{\boldsymbol{n}_i}^P$ and $\boldsymbol{R}_{\boldsymbol{n}_i}^Q$ so that ϵ reflects the uncertainty along the approximated normals $\boldsymbol{n}_i^P$ and $\boldsymbol{n}_i^Q$. The intuition behind this

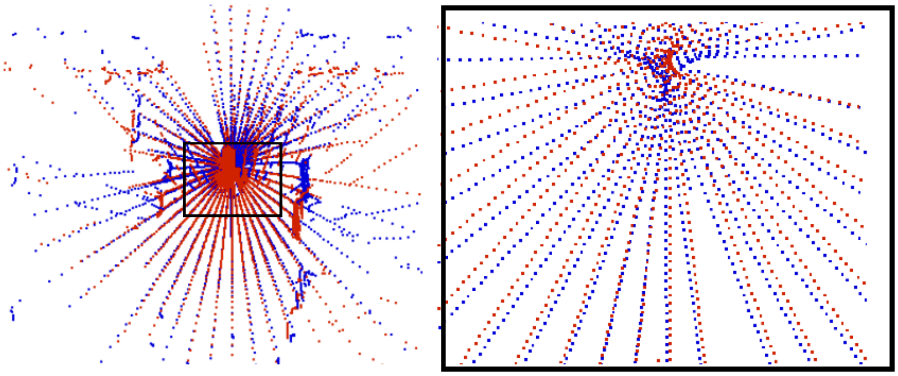
is that we assume the point to lie on the approximated surface while not knowing where the point is lying on the surface. The lower the ϵ the more local planarity is assumed around the point. Consequently, with a low value ($\epsilon \leq 10^{-3}$), the registration error in Equation (4.4) to be minimized converges to a plane-to-plane error metric.

4.3.5 *Registration with Approximate Covariance Estimates*

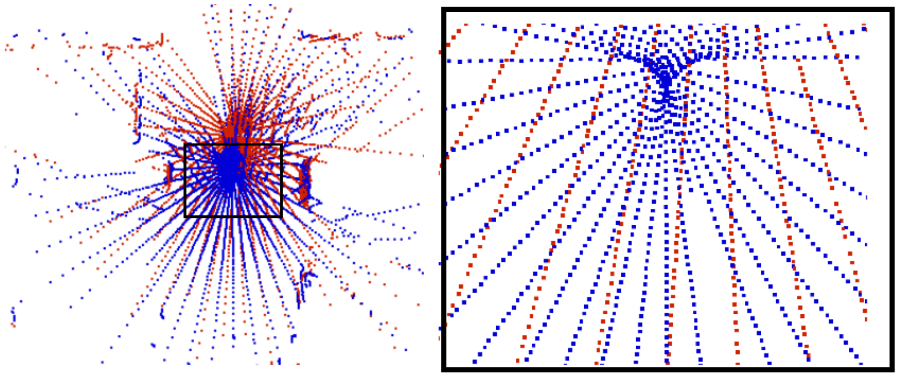
The actual registration of, respectively, two point clouds and the two approximated surface meshes does not deviate from the original GICP algorithm or any other ICP variant. Given a source point cloud P and a target point cloud Q (usually the current and the last aggregated 3D point cloud), we first compute approximate surface reconstructions for both clouds and remove all points not contained in any polygon of the mesh. Using the surface approximations, we compute for all residual points (subsets P' and Q') approximate covariance estimates using Equation (4.9). In each iteration, we then search for closest points in Q' for all points $\boldsymbol{p'} \in P'$. Each found correspondence pair (ij) contributes a measurement error to the non-linear optimization problem using the generalized error metric in Equation (4.4). For finding the optimal transformation minimizing Equation (4.4), we use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. BFGS approximates Newton's method for nonlinear optimization problems. The required second-order derivatives can be efficiently computed analytically due to the simple form of the simplified likelihood $L(\boldsymbol{T})$ in Equation (4.4). Optimization using Newton's method with the found correspondence pairs is stopped when the algorithm converges (usually in 3 to 5 iterations in our experiments) or when the maximum number of iteration steps is reached (in our implementation 10). We inspect the computed pose change and stop the iterative alignment if the pose no longer changes. In case of changes, we apply the computed transformation and start the next iteration with new correspondence pairs.

A typical example of registering non-uniform density point clouds using both the original GICP and our variant with approximate covariance estimates is shown in Figure 4.3. The low angular resolution in these point clouds affects the convergence of the original GICP. In effect, it aligns the individual scan lines and not the sensed environmental structures. Hence, it diverges even from a good initial pose estimate. Our approach accurately aligns the two 3D point clouds.

For a thorough experimental evaluation of the convergence and divergence behavior of our approach, we refer to the pairwise registration experiments in Section 4.5. Overall, the approximate mesh registration can robustly align sparse point clouds, but shows minimal inaccuracies in the final alignment (e.g., deviations in the range of centimeters when compared to ground truth pose estimates).



(a) Generalized-ICP (top and detail view)



(b) Ours (top and detail view)

Figure 4.3: Registering non-uniform density point clouds. (a) Generalized-ICP suffers from inaccurate covariance estimates and incorrectly aligns the two point cloud origins (see detail view) and the individual scan lines. (b) Our approach correctly aligns the two point clouds.

4.4 MAPPING WITH SPARSE 3D LASER SCANS

Registration of sparse 3D point clouds (Section 4.3) can be used to compute the relative transformation between the view poses where two point clouds have been acquired (or aggregated in our case). Likewise, sequential pairwise scan-to-scan registration can be used to obtain an initial trajectory estimate. However, by only using the last point cloud to align a newly acquired one, even small registration errors accumulate and lead to a drift in the estimated trajectory. The resulting trajectories are usually locally accurate and smooth but globally not consistent. The drift can lead to inconsistencies in the map when returning to a previously visited place (loop closures).

4.4.1 *Graph-Based Simultaneous Localization and Mapping*

Graph-based simultaneous localization and mapping aims at computing globally consistent trajectories and maps by building and opti-

mizing a pose graph in which the edges encode the spatial relation between connected poses. The graph optimization forms the back-end of the system while a front-end detects loop closures and computes relative poses to feed the back-end.

In its simplest form, pairwise registration as in Section 4.3 can be used as a front-end to determine an initial trajectory estimate and the vicinity of estimated poses determines the connectivity in the graph, e.g., by connecting all poses within a certain radius and having a similar orientation. Since the laser scanner of our MAV perceives the environment almost omnidirectionally, we can neglect the orientation of view poses and instead connect purely based on Euclidean distance (see Figure 4.4a). In the following, we will present two versions of vertex connectivity:

1. a *single-edge baseline system*: a classic version with a single edge encoding the relative transformation between two view poses and the associated covariance matrix representing the uncertainty in the relative transformation, and

2. the *proposed multi-edge system*: a graph where the connectivity between vertices is not represented using a single edge encoding a relative transformation but instead using multiple edges each encoding a point-to-point correspondence.

In both cases, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents view poses $\boldsymbol{v}_i$ as a set of vertices $\mathcal{V}$ and spatial relations between two view poses $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ as edges e_{ij} in the set of edges $\mathcal{E}$. Each edge in the graph encodes two entities: a local contribution to the measurement error $\boldsymbol{e}$ and an information matrix $\boldsymbol{H}$ which represents the uncertainty of the measurement error. The information matrix is defined as the inverse of the covariance matrix, i.e., it is symmetric and positive semi-definite. The difference between the two systems is the type and number of edge constrains e_{ij}, i.e., the choice of $\boldsymbol{e}$ and $\boldsymbol{H}$. In both cases, however, we model and optimize the graph using the graph optimization framework g2o (Kümmerle *et al.*, 2011).

4.4.2 *Baseline System — Single Edge Connections*

A straightforward extension of our approximate surface registration approach to a graph-based mapping system can be achieved by first applying registration sequentially to pairs of consecutive point clouds $(P_i, P_{j=i+1})$ in order to determine both the relative transforms ${}^i_j\boldsymbol{T}$ and the initial graph connectivity. In this stage, all poses within a radius r get connected in the graph (see Figure 4.4a). In the second phase, we register all connected pairs $(P_i, P_{j\neq i+1})$ that have not yet been registered in the initial registration of consecutive point clouds. In both stages we collect, for every registered pair of point clouds (P_i, P_j),

the estimated transformation ${}^{i}_{j}\boldsymbol{T}$ as well as the associated covariance matrix $\boldsymbol{\Sigma}_{{}^{i}_{j}\boldsymbol{T}}$ and its inverse, the information matrix $\boldsymbol{\Sigma}^{-1}_{{}^{i}_{j}\boldsymbol{T}}$.

In order to get an estimate of the relative pose uncertainty in the form of $\boldsymbol{\Sigma}_{{}^{i}_{j}\boldsymbol{T}}$, we do not only use the estimated transform ${}^{i}_{j}\boldsymbol{T}$ but also the set C of found point correspondences. We compute $\boldsymbol{\Sigma}_{{}^{i}_{j}\boldsymbol{T}}$ using the approximation by Censi (2007):

$$\boldsymbol{\Sigma}_{{}^{i}_{j}\boldsymbol{T}} \approx \left(\frac{\partial^2 L}{\partial \boldsymbol{x}^2}\right)^{-1} \frac{\partial^2 L}{\partial z \partial \boldsymbol{x}}\, \boldsymbol{\Sigma}(z)\, \frac{\partial^2 L}{\partial z \partial \boldsymbol{x}}^{T} \left(\frac{\partial^2 L}{\partial \boldsymbol{x}^2}\right)^{-1} \tag{4.10}$$

where L is the simplified likelihood function from Equation (4.4), z denotes the individual found correspondences C between the two point clouds P_i and P_j, and $\boldsymbol{\Sigma}(z)$ the covariance of the correspondence pairs. Here, the relative transformation between two view poses is not represented as a homogeneous transformation matrix ${}^{i}_{j}\boldsymbol{T}$, but in a parameterized form $\boldsymbol{x} = (\boldsymbol{t}, \boldsymbol{q})^T$ with translation $\boldsymbol{t}$ and rotation by the unit quaternion $\boldsymbol{q} \in \mathbb{H}$.

After registration and covariance estimation, we add a single edge e_{ij} with

$$\text{measurement error} \qquad \boldsymbol{e}_{ij}(\boldsymbol{T}) = {}^{i}_{j}\boldsymbol{T} \tag{4.11}$$

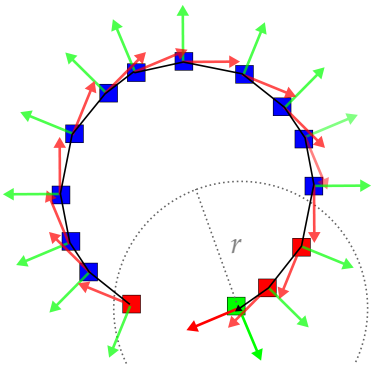
$$\text{and information matrix} \qquad \boldsymbol{H}_{ij} = \boldsymbol{\Sigma}^{-1}_{{}^{i}_{j}\boldsymbol{T}} \tag{4.12}$$

to $\mathcal{G}$ as a spatial constraint between view poses $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$.

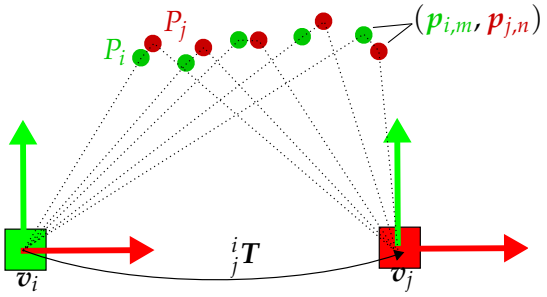
For the actual optimization, we use sparse Cholesky decomposition and Levenberg-Marquardt within the g2o framework (Kümmerle *et al.*, 2011). In order to compensate for loop closures not present in the initial trajectory estimate but introduced by the optimization, we re-compute the connectivity graph. In case of changes (added connections, removed connections or changed connections), we optimize the newly constructed graph again. If no such changes are detected or if a maximum number of iteration steps is exceeded (10 in our experiments), we stop optimizing the graph and compute the final map by aggregating all point clouds using the updated view poses.

4.4.3 *Proposed Approach — Multi-Edge Connections*

The acquired point clouds are quite sparse and, consequently, our estimated transformations are accurate but not as accurate as each laser measurement itself (see Section 4.5.1). When connecting view poses using a single edge encoding transformation and relative pose uncertainty, all the individual correspondence covariances are merged into a single estimate. The merged covariances provide adequate information for refining individual transformations when optimizing over multiple point cloud connections, but can lose the accuracy in individual point correspondences.



(a) Graph structure



(b) Connecting edges (one per correspondence)

Figure 4.4: Graph construction and vertex connectivity: (a) For each pose, we add a vertex to the graph. We connect a vertex (green) to all neighboring vertices (red) within search radius r. (b) Instead of adding a single edge (solid line) encoding the transformation ${}^i_j\boldsymbol{T}$ between two vertices v_i and v_j (as in the baseline approach in Section 4.4.2), we add an edge (dotted lines) for every point correspondence between the two point clouds P_i and P_j in the proposed multi-edge approach (Section 4.4.3).

Instead, we do not use a single edge between vertices to encode their relative pose but connect directly using estimated point correspondences in between the point clouds (see Figure 4.4b). In particular, for every pair of neighboring vertices (v_i, v_j) we search for point correspondences between the respective 3D point clouds P_i and P_j. The central idea behind this decision is three-fold: 1.) we maintain local surface-to-surface alignment accuracy (over multiple point clouds), 2.) we gain a second mean for compensating for the non-uniform densities in our scans and 3.) using point correspondences as edges allows iteratively optimizing the graph and re-estimating the updated correspondences.

For each point correspondence, we add an edge to the graph again using the information extracted from approximate surface reconstruction. Assuming that we already computed local surface normals and approximate covariance estimates as in Equations (4.8) and (4.9), the idea is to use the same error metric as in the pairwise registration, see Equation (4.4). As a straightforward error measurement

between, respectively, two vertices v_i and v_j and the correspondence pair $(p_{i,m}, p_{j,n})$, we use the point-to-point difference vector and approximate its information matrix using the error metric of our registration algorithm:

$$\text{measurement error} \quad e_{ij,mn}({}^i_j\boldsymbol{T}) = p_{j,n} - {}^i_j\boldsymbol{T}\, p_{i,m}, \tag{4.13}$$

$$\text{and information matrix} \quad \boldsymbol{H}_{ij,mn}({}^i_j\boldsymbol{T}) = \left(\boldsymbol{\Sigma}_n^{P_j} + \boldsymbol{R}\boldsymbol{\Sigma}_m^{P_i}\boldsymbol{R}^T\right)^{-1} \tag{4.14}$$

The effect is that every edge contributes its approximate surface-to-surface error term to the system information matrix—thus automatically giving lower influence to incompatible or false correspondences and quickly leading to alignment even for the sparse non-uniform density point clouds.

For the actual optimization, we follow an iterative procedure by 1.) estimating correspondence pairs for all (or a subset of) points $p_{i,m} \in P_i$ in P_j for every two vertices (v_i, v_j) that are to be connected and 2.) optimizing the resulting linearized system for a maximum of ten inner iterations. We repeat these two steps for a maximum of ten outer iterations. For a fast initial coarse alignment in early and an accurate refinement in later outer iterations, we use a linearly decreasing distance threshold between correspondence pairs, starting with double the distance between the vertices. In every outer iteration step, the graph is optimized using dense Cholesky decomposition and Levenberg-Marquardt within the g2o framework (Kümmerle *et al.*, 2011). For both inner and outer iterations, we stop when the system has converged. Convergence in graph optimization (inner iterations) can be detected based on the changes in both view poses and system error as well as the damping factor applied by Levenberg-Marquardt. For detecting convergence in the overall graph refinement in the outer iterations, we check whether the view pose connectivity and the correspondences between connected view poses have changed. When no more changes are found and the inner optimization has converged, we stop optimizing the trajectory estimate and build the final map of the environment.

4.5 EXPERIMENTS AND RESULTS

In order to assess the performance of our approach and the involved components, we have run a series of experiments. For making the presented results both reproducible and comparable, we have recorded different datasets which we make publicly available[1].

1 We have made all datasets in this chapter publicly available. They can be obtained from: `http://www.ais.uni-bonn.de/mav_mapping`.

4.5.1 *Experiments on Pairwise Registration*

The first series of experiments concerns the robustness of our registration approach in terms of both the convergence and the divergence behavior under different resolutions (angles between individual scan lines) and under different initial conditions (e.g., noise in the initial pose estimates).

Registration problems considerably vary depending on the availability and quality of initial pose estimates. Assuming an optimal (ground truth) pose estimate, the point clouds are already aligned and a correct registration result is equal to the initial estimate. That is, any transformation applied by the registration causes the alignment to diverge from the optimal solution. Consequently, a deviation from the ground truth transformation is considered an error in translation and rotation. The divergence behavior is usually not examined in related works but is of utmost importance here since the sparse point clouds acquired by our MAV quickly cause standard registration approaches to diverge when the angles between individual scan lines increase. We also analyze the convergence behavior of the registration approach as is done in related works. Here, the central question is if and how well a registration algorithm converges to the optimal solution for initial pose estimates that are noisy or considerably deviate from the optimal alignment.

In order to evaluate convergence and divergence behavior for different angular resolutions, we have created a dataset of organized point clouds containing ground truth pose information. It was recorded using the same rotating laser scanner but on a mobile ground robot standing still while acquiring 3D point clouds—thus avoiding inaccuracies in laser scan aggregation. The dataset contains point clouds from eight different poses with a total of 6890 2D laser scans acquired over multiple full rotations at each pose. The total trajectory length between the eight poses is roughly 50 m. It was recorded by Schadler *et al.* (2014) in the arena of the DLR SpaceBot Cup[2] competition for semi-autonomous exploration and mobile manipulation in rough terrain (see Figure 4.5). For the dataset, we collected all 2D scan lines acquired at each of the poses, sorted them by rotation angle and re-organized the data to obtain eight full resolution organized point clouds ($\Delta\theta \approx 0.3°$). We annotated each point cloud with the ground truth pose estimate obtained from an accurate multi-resolution surfel mapping approach for dense point clouds (Schadler *et al.*, 2014). For the experimental evaluation, we generated thinned out versions of these eight original point clouds with different angular resolutions and angles $\Delta\theta \in [1°, 90°]$.

2 More information on the DLR SpaceBot Cup can be found on the NimbRo Centauro project website: `http://www.ais.uni-bonn.de/nimbro/Centauro`.

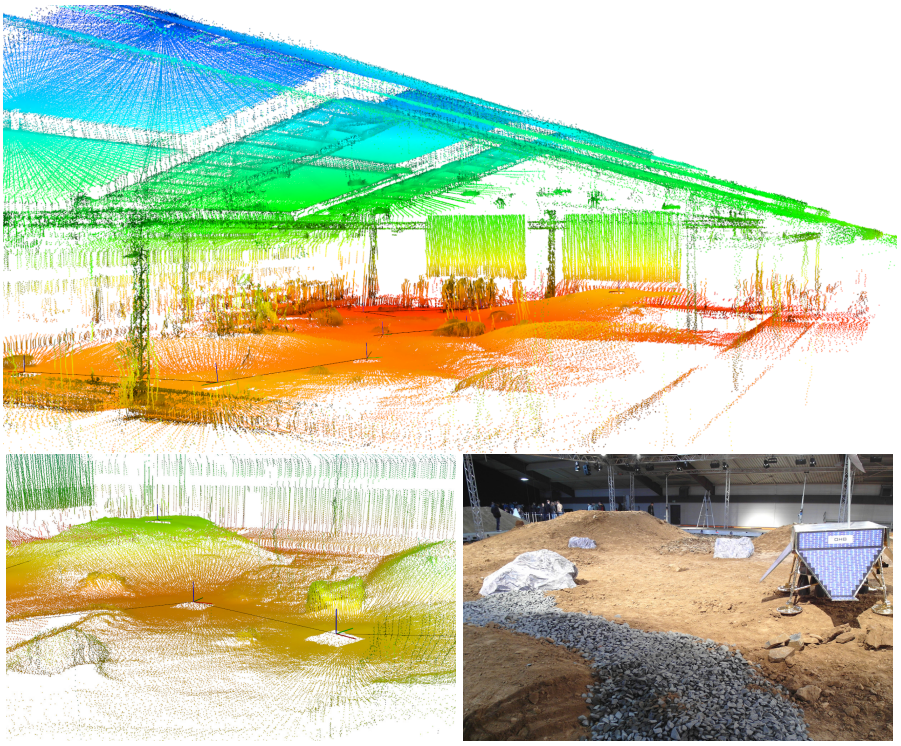


Figure 4.5: Arena of the DLR SpaceBot Cup 2014 where the dataset was recorded. Shown are all aligned point clouds (dense, $\Delta\theta = 1°$) and a photo of the arena.

For both convergence and divergence behavior, we measure registration success in terms of the registration error. In particular, for consecutive point clouds acquired at times i and $i+1$, we inspect the relative deviations $\mathbf{E}_i$ with

$$\mathbf{E}_i := \left(\mathbf{Q}_i^{-1}\mathbf{Q}_{i+1}\right)^{-1}\left(\mathbf{P}_i^{-1}\mathbf{P}_{i+1}\right) \tag{4.15}$$

between ground truth poses $\mathbf{Q}$ and estimated poses $\mathbf{P}$. As suggested by Sturm *et al.* (2012), we focus on the translation error

$$e_{t,i} = \| \operatorname{trans}(\mathbf{E}_i) \|_2, \tag{4.16}$$

i.e., the Euclidean distance between the estimated (relative) pose estimates (trans $(\cdot)$ extracts the translation component). In case of $\boldsymbol{P}_i = \boldsymbol{Q}_i$ and $\boldsymbol{P}_{i+1} = \boldsymbol{Q}_{i+1}$, $\boldsymbol{E}_i$ is the identity matrix and $e_{t,i} = 0$.

4.5.1.1 *Divergence Behavior*

In order to evaluate the divergence behavior, we have chosen pairs of consecutive point clouds from the dataset and registered the respective thinned out copies. In a comparative evaluation, we registered the point clouds of each pair using both the original GICP algorithm and our variant with approximate surface registration. Figure 4.6 shows

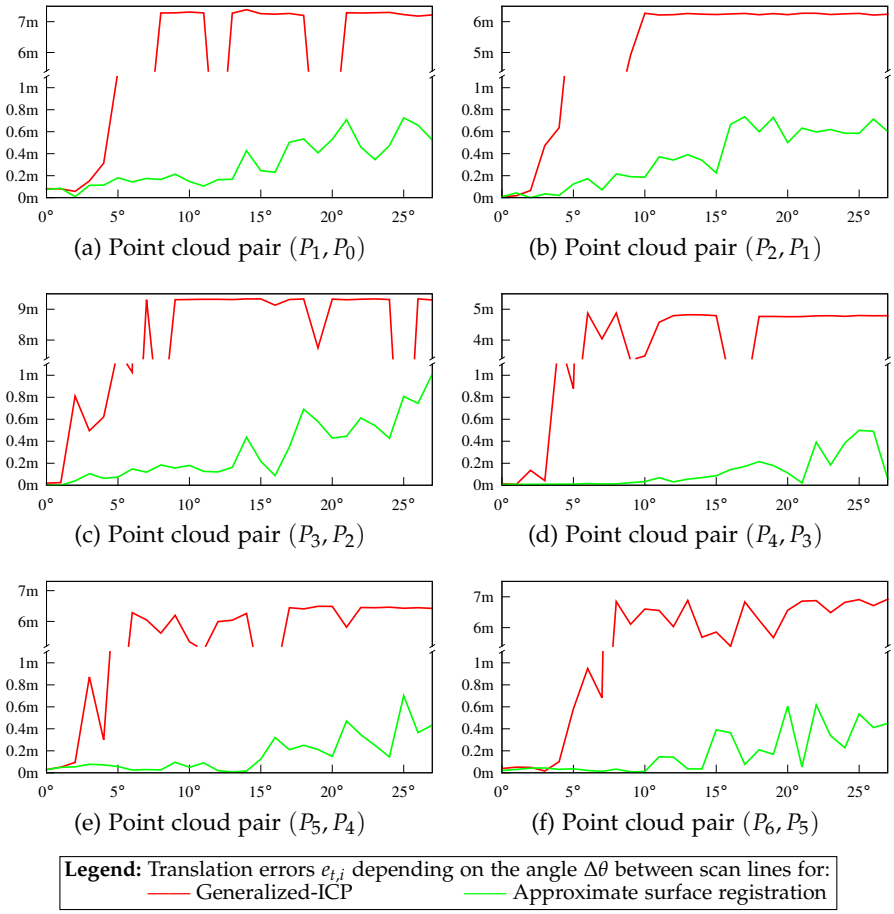


Figure 4.6: Divergence behavior for decreasing density (increasing angles $\Delta\theta$ between scan lines) of our approximate surface registration approach compared to Generalized-ICP for the alignment of six pairs of point clouds (left and right). Translation errors $e_{t,i}$ increase with an increasing angle $\Delta\theta$ between scan lines.

the results of this comparison with decreasing angular resolution (increasing angle $\Delta\theta$ between scans).

Both algorithms achieve optimal registration results for the dense point clouds with deviations from ground truth of only few centimeters. In fact, it is hard to tell whether the pose estimate used as ground truth is better or worse than the achieved alignment. For increasing angles between scan lines, the GICP algorithm quickly starts to fail showing the aforementioned behavior of dragging individual scan lines (and the scan origins) onto another instead of aligning sensed environmental structures. In its extreme, both scan origins coincide and the maximum error in the registration results reflects the Euclidean distance between the ground truth poses. Our approach achieves fairly acceptable results even for very low angular resolutions (angles between scans of $\Delta\theta > 15°$). For smaller angles ($\Delta\theta \leq 10°$), the resulting alignments are very accurate.

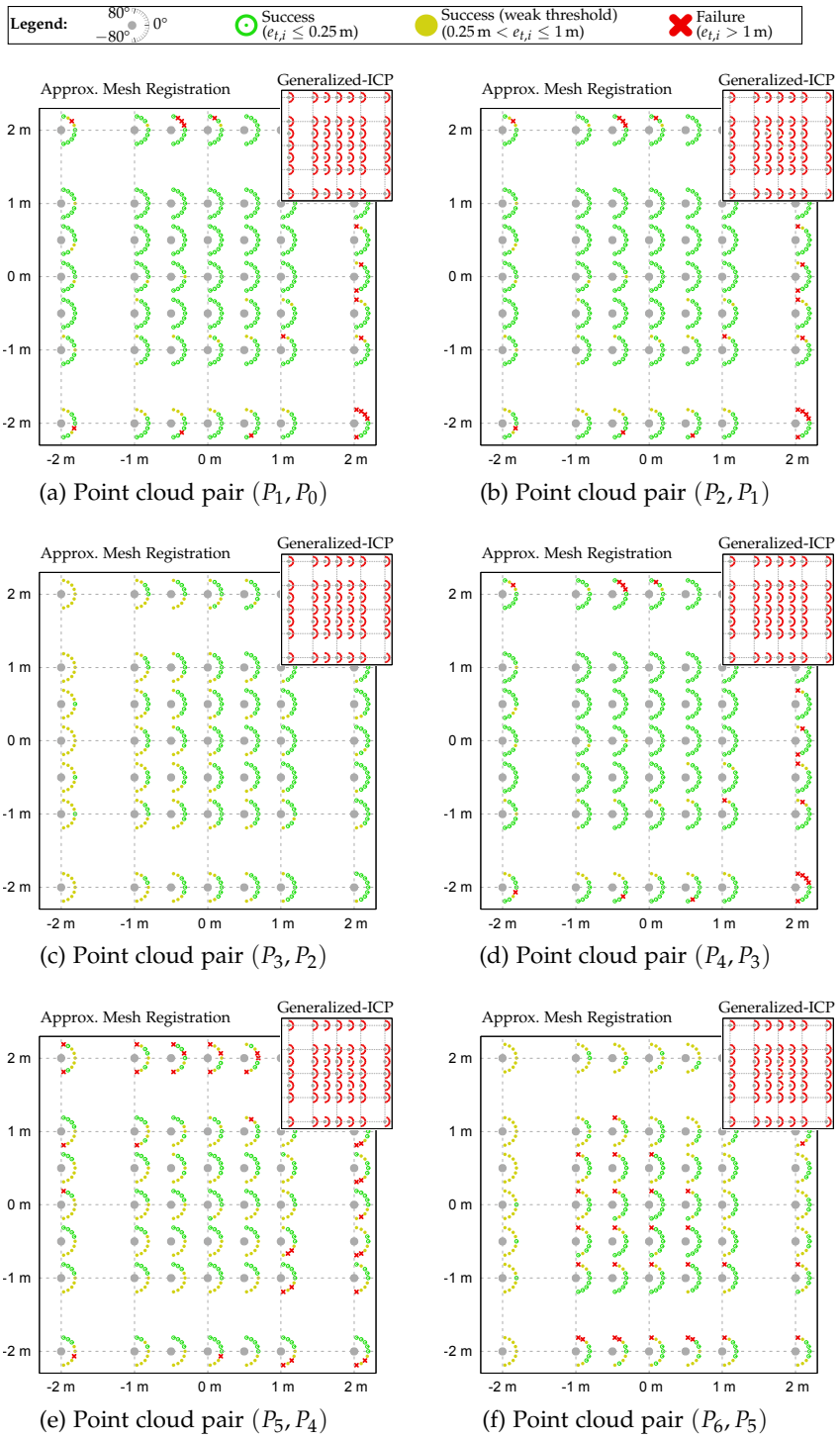
4.5.1.2 *Convergence Behavior*

In order to evaluate the convergence behavior, we have used the same pairs of point clouds as in the evaluation of the divergence behavior. Instead of using all available angular resolutions, we focus on the expected angular resolution of our scanner when flying (i.e., $\Delta\theta \approx 9°$). For each scan pair, we registered the respective point clouds under different initial conditions and quality of initial pose estimates. In particular, we simulate inaccuracies using translation errors of up to 2 m along the x and y axes (i.e., the plane the robot is moving on) and rotation errors of up to 80° about the z axis (i.e., affecting the robot's heading estimate).

In order to measure registration success for the different initial conditions, we have chosen two thresholds for the final translation error $e_{t,i}$ in Equation (4.16): a stricter one (0.25 m) and a weaker one (1 m) similar to the evaluation of registration algorithms by Magnusson *et al.* (2009a). The intuition behind the two thresholds is that poses within the stricter translation threshold are difficult to tell apart for a human observer; poses within the weaker threshold are inaccurate but still fairly well aligned. We consider a registration as failed if the translation error exceeds the weaker threshold. Figure 4.7 presents the results of the evaluation with different initial conditions for the same two scan pairs as used in the evaluation of the divergence behavior. As can be seen, our approach fails in only very few cases (with high initial rotational error and/or high translation error). In the majority of registrations (even with high initial rotational and translation errors), our approach achieves an acceptable alignment even with the strict threshold. With very few exceptions, where the translation error stays within the weaker threshold, the GICP algorithm fails in almost all cases.

4.5.2 *Experiments on Simultaneous Localization and Mapping*

In order to evaluate the performance of our complete mapping pipeline, we recorded a dataset with the flying MAV in a smaller indoor scenario equipped with a motion capture system. It contains a total of 1772 laser range scans in 82 aggregated point clouds. The overall trajectory length is 18.10 m. In a comparative evaluation, we process the complete dataset with different approaches: the original GICP algorithm vs. our approximate mesh registration and the baseline SLAM approach with single edge connections vs. the proposed multi-edge approach. We also compare the two SLAM variants without prior registration, i.e., optimizing the initial vehicle trajectory. In order to evaluate the performance of the approaches under different initial conditions (quality of initial pose estimates), we run three series of experiments: with visual odometry estimates as initial pose estimates, without initial pose estimates (all transformations being identity), and with pose



(a) Point cloud pair (P_1, P_0)

(b) Point cloud pair (P_2, P_1)

(c) Point cloud pair (P_3, P_2)

(d) Point cloud pair (P_4, P_3)

(e) Point cloud pair (P_5, P_4)

(f) Point cloud pair (P_6, P_5)

Figure 4.7: Convergence behavior for poor initial pose estimates at $\Delta\theta = 10°$. Registration success is measured w.r.t. the translation error $e_{t,i}$ using a strict threshold and a weaker threshold. Exceeding the weaker threshold is considered a failure. Each subplot encodes translation errors along the x and y axes with seven initial orientations from $-80°$ to $80°$ rotation error with $0°$ pointing along the horizontal axis. For comparison, despite few exceptions Generalized-ICP fails in all cases.

estimates considerably affected by noise (translation errors of up to 2 m and rotation errors of up to 45°).

For evaluating the accuracy of the pose estimates, we use an error metric proposed by Sturm *et al.* (2012): the absolute trajectory error (ATE). The absolute trajectory error (ATE) focuses on global consistency by aligning and directly comparing absolute pose estimates (and trajectories):

$$\mathrm{ATE}\left(\mathbf{F}_{i:n}\right) := \left(\frac{1}{m}\sum_{i=1}^{m} \| \operatorname{trans}\left(\mathbf{F}_i\left(\Delta\right)\right) \|^2\right)^{1/2} \tag{4.17}$$

with $\mathbf{F}_i(\Delta) := \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i$, where $\mathbf{S}$ is the rigid-body transformation mapping the estimated trajectory $\mathbf{P}_{i:n}$ to the ground truth trajectory $\mathbf{Q}_{i:n}$. The operator trans $(\mathbf{F})$ extracts the 3×1 translational component of the individual transformation errors $\mathbf{F}$.

For evaluating the quality of the resulting map (aggregated point map of all aligned point clouds), we use a measure of mean map entropy. For every point $\mathbf{p}_i$ in the resulting point map P, we compute the local entropy h by:

$$h(\boldsymbol{p}_i) = \frac{1}{2}\ln\left|2\pi e\boldsymbol{\Sigma}(\boldsymbol{p}_i)\right|, \tag{4.18}$$

where $\boldsymbol{\Sigma}(\boldsymbol{p}_i)$ is the covariance of the points in a radius r around $\boldsymbol{p}_i$. The mean map entropy $H(P)$ is then averaged over all points $\boldsymbol{p}_i$ in the map:

$$H(P) = \frac{1}{|P|}\sum_{i=1}^{|P|} h(\boldsymbol{p}_i), \tag{4.19}$$

where $|P|$ is the number of points in P. The intuition behind this metric is the following: the sharper a map region is the lower is the value of the local point entropies in this region. That is, it encodes how planar the region appears in the final map. Consequently, a high-quality map with flat walls, floor and ceiling as well as sharp corners and edges will have a lower map entropy compared to a map resulting from a globally consistent but slightly inaccurate trajectory estimate. However, the metric assumes that the maps to be compared are roughly globally consistent for a fair comparison. Because of that, we also visually inspect the resulting maps and mark those that show inconsistencies.

As an example of accurately aligned point clouds for this dataset, we present an overlay of all acquired point clouds as aligned using our approach in Figure 4.8. In Table 4.1, we report both the mean map entropy of the resulting point maps and the absolute trajectory errors (with root mean square error (RMSE), mean, standard deviation (stdev), min. and max. translation error). The most important finding here is that the proposed multi-edge graph-based approach with the approximate surface registration error metric outperforms the

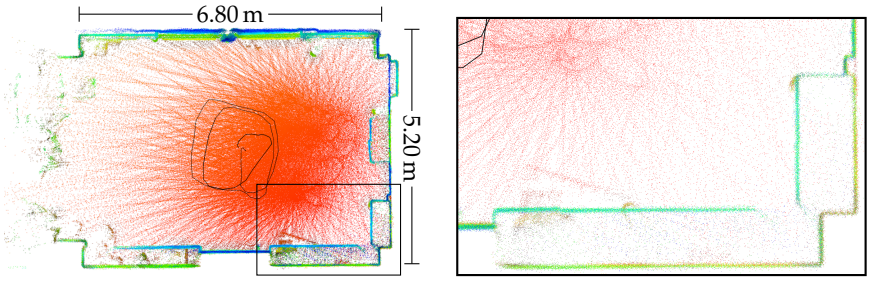


Figure 4.8: Typical registration results: shown are top view(s) of the scans aligned using our approach (mesh registration + global optimization). The room has a size of roughly 6.80 m × 5.20 m. All walls, floor and ceiling are in sight of the MAV at any time. The trajectory has a length of roughly 18.10 m (black line strip around the center of the room).

baseline approach with standard single-edge connections. Moreover, our approach yields exactly the same optimal results for all initial conditions.

4.5.2.1 *Approximate Mesh Registration vs. GICP*

As can be expected from the direct comparison on pairwise registration in Section 4.5.1, our variant with approximate surface information clearly outperforms the original GICP algorithm. The more accurate covariance estimates computed directly on the approximated surfaces allow correcting local alignments. Naturally, the estimated trajectory still drifts away from the ground truth estimate since smaller inaccuracies are accumulated in pairwise registration. Consequently, the resulting point map is globally not consistent.

4.5.2.2 *SLAM with Multi-Edge vs. Single-Edge Connectivity*

Both approaches can adequately compensate for the drift and produce globally consistent trajectories and maps. Still, by using locally very accurate correspondence covariances in the edges of the graph, the multi-edge variant achieves more accurate alignments and scores better in both absolute trajectory error and mean map entropy. Even in case of large simulated errors in the pose estimates, the initial registration with approximate surface information allows both variants converging to a globally consistent estimate.

4.5.2.3 *SLAM without initial registration*

In addition to the proposed pipelines of initial registration and subsequent pose graph optimization, we also evaluated the performance of pose graph optimization without initial registration, i.e., directly on the initial trajectory estimates. Here, the multi-edge variant quickly converges to its solution (within at most 5 outer iterations), regardless

		ATE (m)				Map entropy
Init.	Sequence	RMSE	mean ± stdev	min	max	mean
Visual Odometry	Initial	0.0297	0.0279±0.0101	0.0095	0.0544	−2.6575
	GICP	0.5723	0.5448±0.1753	0.1348	1.0191	−2.1956*
	MR	0.0988	0.0874±0.0459	0.0088	0.1899	−3.4229*
	MR+BL	0.0296	0.0267±0.0135	0.0050	0.0644	−3.4344
	MR+OPT	0.0249	0.0223±0.0110	0.0030	0.0542	−3.8095
	BL only	0.0286	0.0256±0.0128	0.0043	0.0622	−3.4920
	OPT only	0.0249	0.0223±0.0110	0.0031	0.0563	−3.8098
None (all Identity)	Initial	0.9371	0.8971±0.2707	0.3329	1.5810	−2.0035*
	GICP	0.7958	0.7632±0.2252	0.2600	1.2173	−2.2896*
	MR	0.0988	0.0874±0.0459	0.0088	0.1899	−3.4229*
	MR+BL	0.0299	0.0267±0.0134	0.0055	0.0640	−3.4380
	MR+OPT	0.0249	0.0223±0.0110	0.0031	0.0563	−3.8096
	BL only	0.2723	0.1492±0.2277	0.0152	1.3935	−2.5073*
	OPT only	0.0249	0.0223±0.0111	0.0031	0.0563	−3.8097
Simulated Errors	Initial	6.2885	5.9298±2.0934	1.1895	11.790	−3.3392*
	GICP	4.3437	3.8863±1.9400	0.8205	8.5243	−2.5488*
	MR	0.1216	0.1075±0.0568	0.0129	0.2265	−3.4135*
	MR+BL	0.0297	0.0265±0.0134	0.0054	0.0649	−3.4435
	MR+OPT	0.0249	0.0223±0.0111	0.0031	0.0563	−3.8096
	BL only	0.0824	0.0747±0.0346	0.0148	0.175 93	−3.0243*
	OPT only	0.0250	0.0226±0.0113	0.0035	0.0567	−3.8080

* The resulting maps are globally not consistent, e.g., due to drifts.

Table 4.1: Results for the Motion Capture Dataset: absolute trajectory error (ATE) and map quality for the different approaches under different initial conditions: visual odometry as initial pose estimates, no initial pose estimates, and noise-affected pose estimates simulating errors (±200 cm, ±45°, uniformly distributed).
Legend: GICP (Generalized-ICP), MR (Mesh Reg., Generalized-ICP with approximate covariances), BL (single edge connections, Section 4.4.2), OPT (multi-edge connections, Section 4.4.3).

of the quality of the initial pose estimates. In fact, the resulting trajectories and maps are almost identical. The baseline approach converges considerably slower (especially for large errors in the initial pose estimates) and does not find a globally consistent trajectory estimate in 10 iterations.

4.5.3 *Runtime Evaluation*

In order to obtain runtime estimates per component and processing step of the registration and mapping pipeline, we have collected measured runtimes in all experiments. In each processing step, runtimes have been measured in each component either per point cloud (e.g., preprocessing steps), per pair of registered point clouds (in the initial alignment), and for the complete global optimization. Furthermore, the overall total runtime for processing the whole dataset was measured. Naturally, the measured vary depending on the dataset (e.g., number of vertices and edges in the optimization) and depending on the individual point clouds being processed. In the initial alignment, for example, the runtime greatly depends on the convergence speed when optimizing the surface-to-surface metric. Consequently, for pairs of point clouds that are already well aligned, optimization converges earlier than for pairs where the initial relative pose estimate is considerably off. In order to obtain amortized runtime estimates for all components, the individual runtimes were averaged per point cloud being processed. We report the detailed results with mean execution time and standard deviation in Table 4.2.

The most important result here is that 1. point clouds are processed faster than they are acquired (or aggregated), 2. the complete dataset is processed faster than it is recorded. Since the global optimization processes all point clouds in the graph, it may take longer than the time for aggregating a single new point cloud. However, the optimization can easily be decoupled from the rest of the pipeline by letting it run in a separate thread or process thus allowing to pre-process and initially align new point clouds while optimizing the so far built graph. The overall result of the runtime experiments is that the approach can be used onboard to incrementally build environment map online.

4.5.4 *Mapping an Indoor Environment*

As a first proof-of-concept, we have recorded a dataset with the continuously spinning laser scanner on the flying MAV. The MAV was manually remote-controlled through a parking garage of 40 m $\times$ 15 m. Overall, the dataset contains a total of 4420 2D scan lines which are aggregated to 200 3D point clouds (each aggregated over one half rotation of the scanner). The overall trajectory length is 73 m (traveled in 100 s). The measurements cover the complete parking garage and

Component	Runtime
Preprocessing (per 3D scan)	
Aggregation & Organization	$\ll$1 ms
Approx. surface reconstruction	$<$1 ms
Normal and covariance estimation	$<$1 ms
Setting up the search tree (for registration)	$<$1 ms
Registration (per pair of consecutive 3D scans)*	
Estimating correspondences	12$\pm$3 ms
Alignment	29$\pm$9 ms
Global graph optimization (5 loops)**	
Estimating graph adjacency	$\ll$1 ms
Estimating correspondences	47$\pm$26 ms
Optimization	152$\pm$91 ms

*Average over all pairs of consecutive point clouds (all datasets).

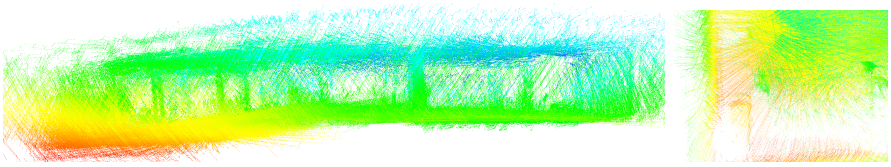
**Average per point cloud in all datasets.

Table 4.2: Runtimes per component and processing step. The reported runtimes are have been collected per component in all datasets (motion capture volume, indoor and outdoor datasets) and averaged over all processed point clouds to obtain an amortized runtime estimate.

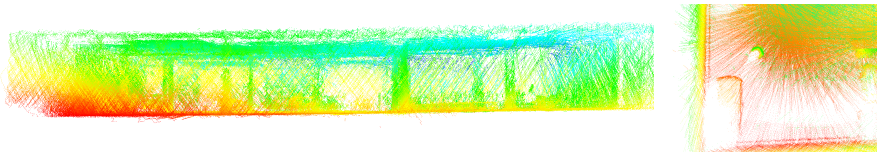
allow for creating a complete model including pillars and other environmental structures as well as parking cars. We used two fish-eye stereo camera pairs on the MAV and visual odometry (Schneider *et al.*, 2013) to obtain an initial pose estimate and to aggregate the individual 2D scan planes to 3D point clouds. As can be seen in Figure 4.9, the visual odometry estimate drifts—leading to an inconsistent map when used on its own, but is accurate enough for scan aggregation, i.e., estimating the movement of the MAV during one half rotation of the spinning laser scanner (500 ms).

In order to obtain a consistent and accurate 3D map out of the acquired data, we register all pairs of consecutive scans (using visual odometry as an initial estimate), and then create a graph where each of the 200 pose vertices is connected to all neighboring vertices within a search radius of 3 m. We iteratively refine the whole graph over five iterations where, in each iteration, the correspondences between connected scans are re-estimated.

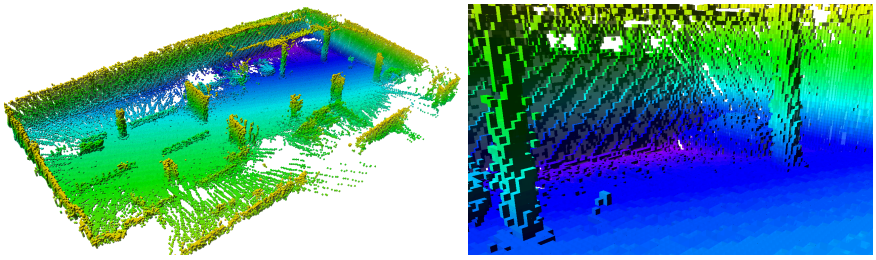
Overall, our approach takes roughly 45 s to build a consistent accurate 3D map, including pre-processing, registration, and graph optimization (approximately 6000 connections with a total of roughly 580 000 edges). Figure 4.9 shows the aligned scans before (visual odometry only) and after our alignment in a side view to visualize the



(a) Aligned scans using visual odometry only (side view) (top view)



(b) Aligned scans after registration and optimization (side view) (top view)



(c) Views on the built 3D map (ceiling removed) with pillar and car silhouette

Figure 4.9: Results of an indoor mapping mission. Shown are all point clouds acquired in the parking garage and aligned using (a) visual odometry only, and (b) after registration and graph optimization (points colored by height). Using visual odometry only leads to a drift while our approach provides a consistent and accurate map, as can be seen in the detail views of a corner with a cylindrical pillar and the silhouette of a car. Note that axes (and color coding) are not aligned to environmental structures but reflect the orientation of the flying MAV.

removal of drift, and in a top view on environmental details (a cylindrical pillar and the silhouette of a car) that show the accuracy in the final model. Using the mean map entropy as a measure of map quality, we obtain an entropy of -2.30 for visual odometry, whereas our approach achieves a lower entropy (sharper map) of -3.58. Visually inspecting the constructed map shows neither major inconsistencies nor minor inaccuracies.

4.5.5 *Complete Outdoor Mapping Missions*

The indoor environment in the experiment series in the motion capture volume does not pose major challenges. In fact, since it is only a single room where all environmental structures can be sensed from every view pose, the dataset constitutes the best case for registration algorithms. The environment in the previous mapping experiment is larger and the vehicle trajectory is more complex to optimize since

it does not form a complete loop. Still, the overall dataset has to be considered rather simple since, again, almost all environmental structures such as walls and pillars are visible from any view point in the dataset.

As a proof-of-concept for the complete system, we conducted two complete outdoor mapping missions. In these missions, the micro aerial vehicle autonomously navigated to a set of predefined waypoints in order to map a building of *Gut Frankenforst*—a research station operated by the Institute for Veterinary Research at the University of Bonn (see Figure 4.10a). Not only at the predefined waypoints but over the whole trajectory, the MAV collected laser range scans which where then processed offline by our approximate mesh registration and multi-edge pose graph optimization approach. This environment poses far more challenges than the indoor scenario: most notably a larger part of the measurements do not lie on the distinctive structures of the building but on vegetation around the building, thus forming rather random measurements when seen in individual sparse point clouds. Moreover, the point clouds only capture parts of the environment making it necessary to correctly detect and align loop closures.

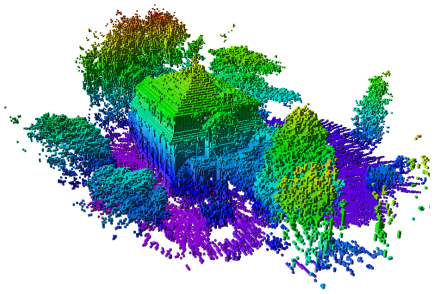
The first mission aims at mapping the front facade of the building. The MAV captured a total of 2409 laser range scans over a trajectory of 30.43 m along the facade of the building. On a single core of an Intel Core i7-3740QM CPU, our approach took 92 s to construct a globally consistent point map out of the 118 aggregated point clouds. Most of this time was spent on the pose graph optimization while aggregating and pre-processing point clouds as well as registering consecutive point clouds was a matter of only few milliseconds per cloud. The pose graph optimization converged after four iterations. The resulting point map and trajectory estimate are shown in Figure 4.10c.

In the second mission, waypoints were distributed around the complete building. The MAV traveled a total of 307.13 m to reach all waypoints and collected 21 475 laser range scans. The scans were aggregated to 859 point clouds. Our approach took roughly 200 s to align all point clouds and construct a globally consistent map of the building and the surrounding vegetation. The resulting point map and trajectory estimate are shown in Figure 4.10d. Finally, we construct an OctoMap (Hornung *et al.*, 2013) as a memory-efficient representation of the environment, e.g., for being able to plan paths for future missions. It is shown in Figure 4.10b.

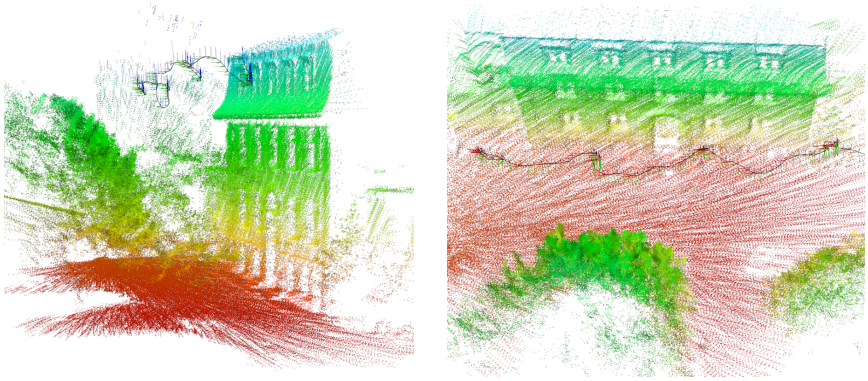
4.6 CONCLUSIONS AND FUTURE WORK

We have presented a complete pipeline for registration and mapping with particularly sparse laser scans acquired by an autonomous MAV. The non-uniform point densities within and between individual laser range scans in the aggregated point clouds negatively affect standard

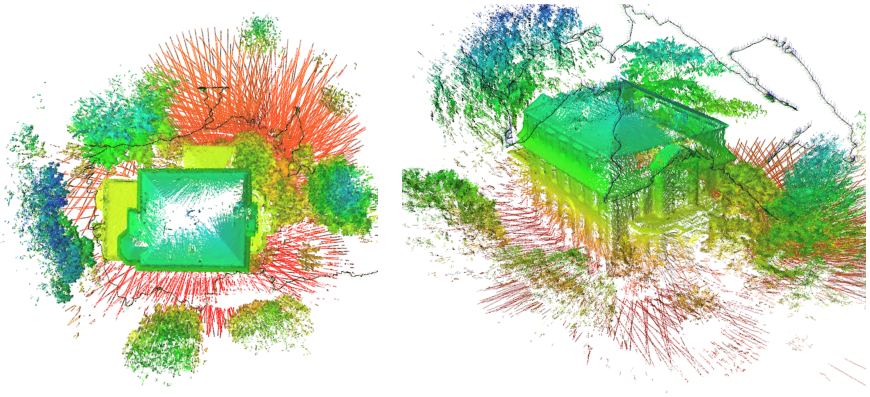
(a) Photo of the building



(b) Final 3D map of the building



(c) Mapping the front facade: point map and trajectory



(d) Mapping the building: point map and trajectory

Figure 4.10: Results of two complete missions for mapping a building (a): a shorter mission for mapping the front facade (c) and a longer mission for mapping the complete building (d). Shown are both the resulting map as an aggregation of all aligned point clouds and the estimated vehicle trajectory. The OctoMap (Hornung *et al.*, 2013) constructed for the longer mission is shown in (b).

approaches to registration as well as neighborhood searches and local feature estimation. In order to compensate for the non-uniform point densities in the point clouds, we exploited the organized data structure and computed an approximate surface reconstruction. Point features such as local surface normals and covariances were then deduced from the topology in the resulting mesh in order to obtain proper estimates of the underlying surface even in regions where the measurement density is particularly sparse.

We presented a registration approach based on the Generalized-ICP algorithm that makes use of the approximated surface information. It is able to adequately align aggregated point clouds regardless of the quality of initial pose estimates. Moreover, experiments have shown that the proposed approach clearly outperforms the original GICP algorithm for the sparse point clouds acquired by our MAV. It is very likely that the combination of approximate surface reconstruction and deducing surface statistics from the resulting mesh can also improve the performance of other surface-based registration algorithms, e.g., the NDT registration algorithm of Magnusson *et al.* (2007).

For being able to construct globally consistent 3D environment maps, we have presented an approach to pose graph optimization again making use of the approximated surface information. Instead of representing relative pose estimates in single edges between connected vertices in the graph, it uses one edge per point correspondence between the acquired point clouds. Each of these edges encodes the same error metric as in the registration approach. Our experiments indicate that this multi-edge variant shows superior performance compared to a baseline system following the single-edge approach with robust covariance estimates.

In a final experiment, we could demonstrate that our approach is able to adequately align point clouds aggregated in real mapping missions and to provide both globally consistent environment maps and reliable trajectory estimates.

Regarding possible extensions and future work, the presented approach was only used offline to process the data after it had been acquired in a mapping mission. Furthermore, our approach adequately aligns the aggregated point clouds but does not change the pose of individual laser range scans within an aggregated point cloud. That is, our approach can compensate for pose estimation errors between view poses where point clouds have been aggregated but not for errors in the motion estimation during point cloud aggregation. In the worst case, a single point cloud may become ill-formed and not correctly aligned to the other point clouds (i.e., an outlier point cloud in the resulting map). It is a matter of future work to apply the registration and pose graph optimization pipeline online and also to correct the pose of individual laser range scans once the neighboring aggregated point clouds are aligned.

In addition, the proposed approach clearly distinguished between initial alignment and final optimization. Moreover, the final optimization was performed on the complete pose graph. In the following chapter, we further investigate optimizing the pose graph hierarchically by aligning newly acquired point clouds in local windows of point clouds and view poses, and to only globally optimize the complete pose graph if the optimization of the subgraph in the local window shows larger view pose deviations or conflicts.

5 REGISTRATION AND MAPPING FOR RGB-D CAMERAS

Consumer color and depth cameras (RGB-D cameras) have attracted much attention in the fields of robotics and computer vision, especially for object modeling and environment mapping. A key problem in all these applications is the registration of sequences of RGB-D images. In this chapter, we present an efficient yet reliable approach to align pairs and sequences of RGB-D images that makes use of local surface information. We extend our works (Chapter 4) on three-dimensional (3D) mapping with micro aerial vehicles (MAVs) to sequences of RGB-D images. The resulting alignment is based on a robust surface-to-surface error metric and uses multiple surface-to-surface patch matches between pairs of RGB-D images. We focus on the questions 1. if the approach can also work with sequences of RGB-D images and 2. what needs to be changed to make it both efficient and reliable. The latter is achieved with several extensions which are introduced in this chapter. These extensions involve filtering to smooth the underlying data, sampling to reduce the amount of data being processed and aligning newly acquired point clouds in local windows of poses in order to make the alignment more stable while at the same time gaining efficiency. Quantitative evaluations show that the resulting approach is competitive with state-of-the-art approaches to RGB-D simultaneous localization and mapping (SLAM).

5.1 INTRODUCTION

RGB-D cameras have huge potential in improving the perception capabilities of robots and automated vision systems in general. They acquire color (RGB) images and depth (D) images both at high frame rates, e.g., 30 Hz. Intrinsic and extrinsic calibration of the two image sources yields colored 3D point clouds (see Figure 5.1). Due to their comparably low cost, low weight, and small form factor, RGB-D cameras have attracted much attention in the fields of robotics and computer vision, especially for object modeling and environment mapping. What most applications have in common is that they require RGB-D images to be taken from multiple different viewpoints and that the acquired images need to be reliably registered such that the overlapping regions in the images match as well as possible. In the literature, this problem is usually referred to as SLAM: building a map of the environment and localizing the information acquiring sensor(s) therein so as to consistently update and extend the map.

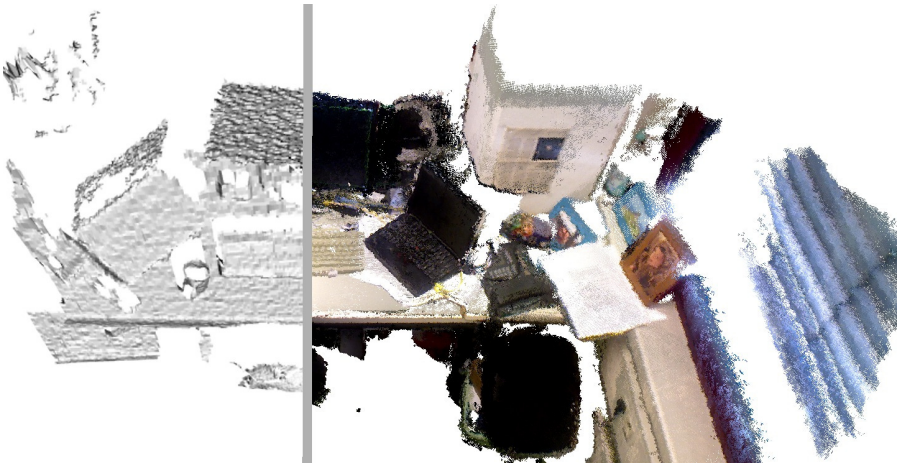


Figure 5.1: Typical registration result for a sequence of RGB-D images. Left: approximate surface reconstruction (unfiltered) of the first cloud. Right: the points of all aligned colored point clouds.

In recent years, many different approaches to visual odometry (Kitt *et al.*, 2010; Huang *et al.*, 2011; Engel *et al.*, 2013), SLAM (Scherer and Zell, 2013; Henry *et al.*, 2012; Kerl *et al.*, 2013; Forster *et al.*, 2014; Endres *et al.*, 2014; Engel *et al.*, 2014), and dense mapping (Newcombe *et al.*, 2011; Stückler and Behnke, 2014; Whelan *et al.*, 2013; Steinbruecker *et al.*, 2013) have been proposed of which some are specifically tailored for RGB-D cameras. These methods are either based on features matched and tracked over sequences of images, or directly operate on the (semi-) dense color and depth images. Most approaches select a set of keyframes and optimize the resulting pose graph in order to obtain a globally consistent trajectory and map. State-of-the-art methods achieve globally consistent trajectories with low errors in pose estimation at high frame rates. We include some of them in a comparative evaluation.

In this chapter, we state and address the problem of RGB-D SLAM in terms of multi-view 3D registration based on point correspondences between frames that encode a surface-to-surface error metric. The approach is based on the method in Chapter 4 for 3D laser scan registration and mapping with micro aerial vehicles. In order to compensate for the non-uniform point densities within and between individual scan lines of the fast rotating scanner, we approximated the underlying surface and used a generalized error metric (Segal *et al.*, 2009) for obtaining robust registrations and accurate 3D maps of the sensed environmental structures such as buildings. In this chapter, we extend the approach to be applicable to sequences of RGB-D images and make the following contributions:

1. In order to reduce the drift during the initial tracking of the camera, we register a newly acquired image against a local window of frames as opposed to the last (key) frame.

2. We integrate our previous works on range image segmentation (Holz and Behnke, 2014a) for efficiently computing local features such as surface normals on an approximate mesh representation, and for edge-aware filtering of the underlying points and the computed features to compensate for noise especially in the depth images.

3. To cope with the larger amount of data of RGB-D images in our multi-edge alignment approach, we efficiently sample both points in the images and found correspondences.

As a result, our approach of using multiple edges between views that encode surface-to-surface constraints can be applied to RGB-D video. Moreover, its performance is competitive with other state-of-the-art approaches. In fact, the proposed local window multi-edge alignment has a huge potential of contributing to other SLAM and object modeling pipelines. We present results of a thorough comparative experimental evaluation that proof these claims.

The approach presented in this chapter was first published at the European Conference on Mobile Robots (Holz and Behnke, 2015b).

5.2 RELATED WORK

Approaches to SLAM using monocular cameras, RGB-D cameras and stereo cameras can in general be split into two different categories: feature-based methods that compute and track distinct repeatable key points and associate them using feature descriptors, and direct methods densely registering the acquired data.

For monocular cameras, a hybrid approach is the semi-dense visual odometry method proposed by Engel *et al.* (2013). It first computes inverse depth maps which are then used to align subsequent frames. A similar approach is followed by Forster *et al.* (2014). Engel *et al.* (2014) extend their approach to build globally consistent maps even of large-scale environments.

Scherer and Zell (2013) present an RGB-D SLAM approach that is efficient enough to be computed onboard an autonomous micro aerial vehicle. It is based on tracking FAST keypoints (Rosten and Drummond, 2006; Rosten *et al.*, 2010) and the fast hierarchical graph optimization of Grisetti *et al.* (2010). The FAST corner detector is also used by Huang *et al.* (2011) in their visual odometry method FOVIS.

A popular approach developed specifically for RGB-D images is RGB-D SLAM (Endres *et al.*, 2012b; Endres *et al.*, 2014). It uses the color image to extract and match visual keypoints and descriptors (SURF, SIFT and ORB). The alignment relies on the 3D coordinates of keypoints obtained from the depth image. A similar approach is followed by Henry *et al.* (2012) with FAST keypoints.

One of the first successful demonstrations of dense registration and mapping has been presented by Newcombe *et al.* and was coined KinectFusion (Newcombe *et al.*, 2011). KinectFusion uses signed distance functions in a grid-based environment representation and ICP-based registration (Besl and McKay, 1992) for aligning newly acquired depth images. All components are implemented on a GPU and allow (near) real-time operation. Assuming the camera movements between frames are small, this incremental registration can reliably align the data and update the environment model. By using more information than only a single frame against which KinectFusion registers considerably reduces drift usually arising in pairwise registration. In many cases, incremental registration can achieve globally consistent environment maps without the need for detecting loop closures and global optimization, e.g., as shown in a previous work on two-dimensional (2D) laser-based mapping (Holz and Behnke, 2010). The drawback of incremental registration is that errors made in the update of the used environment representation cannot be corrected later. In this chapter, we address the alignment of captured frames in terms of multi-view registration and do not build a particular environment representation.

Newcombe *et al.* (2015) later presented DynamicFusion, an extension of KinectFusion for modeling dynamic objects, e.g., the heads of human users with different facial expressions and moving in front of an RGB-D camera. Another very recent extension is ElasticFusion (Whelan *et al.*, 2015) that in contrast to most other SLAM approaches does not use a pose graph which is optimized. Instead, ElasticFusion is map-centric and builds a deformable environment map. As the camera moves, parts in the immediate vicinity of the visible scene become active and are updated and optimized as new information is acquired. This formulation is particularly efficient and allows for real-time scene modeling without an explicit pose graph.

Steinbruecker *et al.* (2013) also use signed distance functions for dense mapping but organize the map in an octree structure. Stückler and Behnke (2014) proposed a surfel-based registration method for constructing muli-resolution surfel maps (MRSMAPs) that are also represented in an octree. Kerl *et al.* (2013) follow a different visual SLAM approach (called DVO-SLAM where DVO stands for dense visual odometry) by minimizing the photometric and the depth error over all pixels. We include RGB-D SLAM (Endres *et al.*, 2012b), MRSMAP (Stückler and Behnke, 2014), DVO-SLAM (Kerl *et al.*, 2013), and an open source implementation of KinectFusion (Newcombe *et al.*, 2011) in a comparative experimental evaluation. In a second series of experiments, we also include results reported for Kintiniuous (Whelan *et al.*, 2012) and ElasticFusion (Whelan *et al.*, 2015).

Recently, Maier *et al.* (2014) presented an efficient approach to RGB-D object modeling. They split the camera trajectory into chunks of equal size, and first optimize the alignments within the chunks before glob-

ally aligning the chunks to each other. Since the camera is moved around the object to model, these splits along the trajectory yield spatially coherent partitions. We achieve a similar behavior by using local windows in the initial alignment of newly acquired frames. Moreover, the local windows include earlier frames in case of loop closures. The local alignment can then compensate for the accumulated drift or trigger a global optimization of the trajectory in case conflicts are found. Our local alignment approach is inspired by the double window approach in the SLAM framework of Strasdat *et al.* (2011).

In multi-view scan matching, poses are determined simultaneously by aligning all scans. In the 2D domain, a popular approach is the one by Lu and Milios (1997) which is often referred to as LUM. Borrmann *et al.* (2008) extend this approach to six degrees of freedom for the alignment of 3D scans and present methods to efficiently deal with the resulting nonlinearities (Borrmann *et al.*, 2008). The resulting approach first applies the ICP algorithm to align consecutive point clouds and then builds a graph based on the determined connectivity of view poses similar to our approach. Both the determined transformations and the sets of point correspondences are represented in the edges. From both, a measurement vector and its covariance matrix are computed which are then fed as one block into a large linear system for optimization. In contrast, in our approach, every correspondence pair forms a block in the final non-linear error function. Furthermore, LUM uses a point-to-point error metric as in the original ICP algorithm. Instead, we approximate the surface and use a probabilistic surface-based error metric.

Similar to our multi-edge alignment step are the approaches of Zlot and Bosse (2014) and Ruhnke *et al.* (2012). For mapping mines with a continuously spinning laser scanner, Zlot and Bosse (2014) use non-rigid surfel registration and graph optimization for aggregating point clouds and building consistent maps. Ruhnke *et al.* (2012) also use raw point matches as constraints in the graph and apply a surfel-based error metric to iteratively refine both the sensor poses and the positions of the points. Their approach can build highly accurate object models but requires a rough initial alignment of the dense RGB-D data. Moreover, by optimizing the position of every point in the resulting object model, the approach is computationally complex. In contrast, we aim at both initially aligning the acquired point clouds and building globally consistent environment models while trying to reduce the complexity of the involved processing steps, e.g., by using only descriptive subsets o the dense RGB-D data and local windows. The idea behind this chapter is to 1. apply our pipeline for 3D mapping with MAVs (Chapter 4) to dense RGB-D data, 2. make the necessary adaptions to make it both applicable and feasible, and 3. evaluate how the resulting system compares to state-of-the-art RGB-D SLAM approaches.

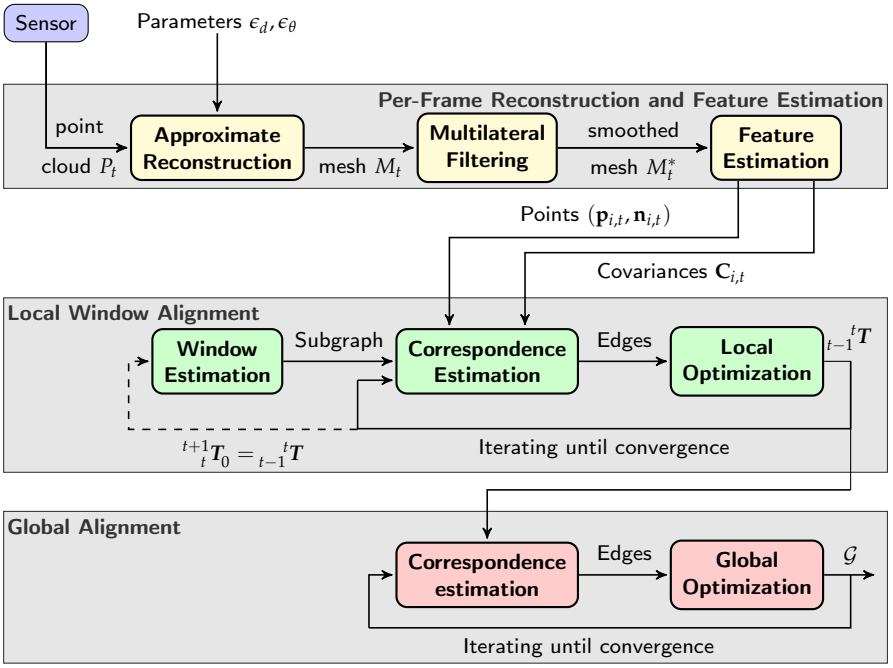


Figure 5.2: System overview and data flow. For a newly acquired point cloud, we first approximate the underlying surface reconstruction. Using the mesh topology, we compute approximate local surface normals and apply a multilateral filter to smooth both points and normals. We then compute local covariances and feed the point clouds as well as the computed features into the local alignment. Global optimization then yields globally consistent trajectories in the form of the optimized posed graph $\mathcal{G}$.

5.3 METHOD

Our approach is split into three stages. In the first stage, we approximate, for each frame, the underlying surface in the form of a quad mesh. The mesh serves three purposes: it allows 1. computing features such as surface normals directly on the mesh, 2. extracting neighborhoods from the mesh topology, and 3. caching values such as computed distances and normal deviations in its edges. Referring to Figure 5.2, the extracted mesh is then smoothed and used for feature extraction. In the second stage, both the mesh and the computed features are fed into the local alignment so as to keep track of the camera pose. If loop closures are detected, the so far estimated trajectory is globally optimized in the third stage. The number of frames being processed increases in the stages. Approximate surface reconstruction and feature estimation is done once per frame, i.e., only a single frame is processed at a time. The local window alignment processes k frames where k is, respectively, the size of the local window and the number of poses in the subgraph. Finally, the global alignment stage processes all frames and optimizes all poses in the graph.

The resulting processing pipeline resembles the mapping pipeline for MAV-mounted 3D laser scanners (Chapter 4), and extends it by 1. a local window alignment in order to get more robust initial pose estimates, 2. efficient sampling of points and inter-frame correspondences in order to reduce the amount of data and speed up computations, and 3. re-using the original approximate surface reconstruction for RGB-D images and noise models as used for range image segmentation (Chapter 3). In the following, we will present all components in the pipeline and show experimental results which show that the resulting approach can reliably register sequences of RGB-D images.

5.3.1 *Approximate Surface Reconstruction*

Surface reconstructions are compact representations of the underlying sensed environmental structures and can become handy in a variety of pre-processing tasks such as computing point neighborhoods, local surface normals or smoothed (and upsampled or downsampled) representations (Holz and Behnke, 2014a). In order to compute an approximate surface reconstruction, we follow the same approach as in Chapter 3 in the context of range image segmentation. We traverse an organized point cloud P once and build a simple quad mesh by connecting every point $\boldsymbol{p} = P(u, v)$ (in the u-th row and the v-th column) to its neighbors $P(u, v+n)$, $P(u+n, v+n)$, and $P(u+n, v)$ in the next row and column (with $n \neq 1$ the input point cloud is directly subsampled). Subsampling, e.g., with $n = 4$ drastically reduces the amount of data per frame while keeping information about all dominant structures needed for reliably aligning the resulting mesh.

As in Chapter 3, we only add a new quad to the mesh if $P(u, v)$ and its three neighbors are valid measurements, and if all connecting edges between the points are not occluded. The first check accounts for possibly missing or invalid measurements in the organized data structure. For the latter occlusion checks, we examine if one of the connecting edges falls into a common line of sight with the viewpoint $\boldsymbol{v} = \mathbf{0}$. We define an edge to be valid iff the maximum edge length is not exceeded (threshold ϵ_d) and the angle between the connected points passes the occlusion check (threshold ϵ_θ), see Equation (3.1).

The first check accounts for sensor noise, i.e., tolerable depth discontinuities such as quantization effects in the depth images. We use a simple isotropic noise model for Microsoft Kinect™ cameras that we have already used for segmenting RGB-D images (see Chapter 3). It depends on the measured distance z to $\boldsymbol{p}_i$:

$$\epsilon_d(z) = n\ \sqrt{2}\ \sigma(z), \tag{5.1}$$

$$\text{with } \sigma(z) = 0.00263z^2 - 0.00519z + 0.00755, \tag{5.2}$$

where n is the subsampling factor applied, i.e., using only every n-th row and column for constructing the quad mesh. If both distance

and occlusion checks pass, we add a new quad. Otherwise, holes arise. After construction, we simplify the resulting mesh by removing unused vertices. We evaluate different subsampling factors in the experimental evaluation in Section 5.4.

5.3.2 *Multilateral Filtering*

Naturally, sensor measurements are affected by noise. Especially depth images suffer from distance-dependent noise and quantization effects. In order to compensate for local noise in depth measurements, we apply a filter for smoothing both the points and their normals while preserving edges in the sensed geometric structures. The formulation of our filter is motivated by the concept of multilateral filtering (Butt and Rajpoot, 2009) and measures the similarity of points w.r.t. their position, surface orientation, and appearance. We use the same filter as in Section 3.4.4. In this pipeline, we filter both a point $\mathbf{p}_i$ and its normal $\mathbf{n}_i$ over its 1-ring-neighborhood N_i, i.e., all points that are directly connected to $\mathbf{p}_i$ by an edge in the mesh. As in Section 3.4.4, three weights α, β, and γ can be used to adjust the behavior of the filter. Equally weighting distance, surface normal and color deviation term already achieves considerable smoothing while preserving edges and corners ($\alpha = \beta = \gamma = 1$). Depending on the desired smoothing level, we extend the point neighborhood to include the neighbors of neighbors and ring neighborhoods farther away from the point. We include several parameterizations of the bilateral in the experimental evaluation in Section 5.4.

5.3.3 *Approximate Normal and Covariance Estimates*

In order to estimate local surface normal and covariance matrix of a point, we directly extract its local neighborhood from the topology in the mesh instead of searching for neighbors. We compute the normal $\boldsymbol{n}_i$ for a point $\boldsymbol{p}_i$ directly on the mesh as the weighted average of the plane normals of the N_T faces surrounding $\boldsymbol{p}_i$ (extracted from the topology):

$$\boldsymbol{n}_i = \frac{\sum_{j=0}^{N_T} (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,b}) \times (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,c})}{\| \sum_{j=0}^{N_T} (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,b}) \times (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,c}) \|}, \tag{5.3}$$

with face vertices $\boldsymbol{p}_{j,a}$, $\boldsymbol{p}_{j,b}$ and $\boldsymbol{p}_{j,c}$. We then compute the local covariance matrix $\boldsymbol{\Sigma}_i$ as done by Segal *et al.* (2009):

$$\boldsymbol{\Sigma}_i = \boldsymbol{R}_{\boldsymbol{n}_i} \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{R}_{\boldsymbol{n}_i}{}^T \tag{5.4}$$

with a rotation matrix $\boldsymbol{R}_{\boldsymbol{n}_i}$ so that ϵ reflects the uncertainty along the approximated local surface normal $\boldsymbol{n}_i$. The intuition behind this is that we assume the point to lie on the approximated surface while

not knowing where the point is lying on the surface. The lower the uncertainty ϵ the more we assume local planarity around measured points. Consequently, with a low value ($0 < \epsilon \leq 10^{-3}$), the registration error to be minimized (introduced in the following) converges to a plane-to-plane error metric.

5.3.4 *Surface-to-Surface Alignment*

In order to align, respectively, two approximated surfaces and two organized point clouds P and Q, we follow the same approach as in the context of 3D mapping with MAVs: we search for closest neighbors in Q for points $\boldsymbol{p}_i \in P$ and iteratively minimize the distances between the found matches. Instead of minimizing the point-to-point distances $\boldsymbol{d}_{ij}^{(\boldsymbol{T})} = \boldsymbol{q}_j - \boldsymbol{T}\boldsymbol{p}_i$ of the set of found correspondences $\mathcal{C}$ to determine the transformation $\boldsymbol{T}$ as in the original Iterative Closest Point algorithm (Besl and McKay, 1992), we use the generalized error metric introduced by Segal *et al.* (2009). It generalizes over the different available error metrics (point-to-point, point-to-plane, plane-to-plane) and thus takes into account information about the underlying surface. The Generalized-ICP (GICP) models the distribution

$$\boldsymbol{d}_{ij}^{(\boldsymbol{T})} \sim \mathcal{N}\left(\boldsymbol{b}_j - \boldsymbol{T}\boldsymbol{a}_i,\ \boldsymbol{\Sigma}_j^B + \boldsymbol{R}\boldsymbol{\Sigma}_i^A\boldsymbol{R}^T\right)$$

where $\boldsymbol{R}$ is the rotation matrix of $\boldsymbol{T}$ under the assumption that both points in P and points in Q are itself drawn from independent normal distributions, i.e., $\boldsymbol{p}_i \sim \mathcal{N}(\hat{\boldsymbol{p}}_i, \boldsymbol{\Sigma}_i^P)$ and $\boldsymbol{q}_j \sim \mathcal{N}(\hat{\boldsymbol{q}}_j, \boldsymbol{\Sigma}_j^Q)$. Given the correspondences $ij \in \mathcal{C}$, the optimal transformation $\boldsymbol{T}^\star$ best aligning P to Q can then be found using maximum likelihood estimation (MLE):

$$\begin{aligned}\boldsymbol{T}^\star &= \arg\max_{\boldsymbol{T}} \prod_{ij\in\mathcal{C}} p\left(d_{ij}^{(\boldsymbol{T})}\right) = \arg\max_{\boldsymbol{T}} \sum_{ij\in\mathcal{C}} \log\left(p\left(d_{ij}^{(\boldsymbol{T})}\right)\right) \\ &\simeq \arg\min_{\boldsymbol{T}} \underbrace{\sum_{ij\in\mathcal{C}} d_{ij}^{(\boldsymbol{T})^T}\left(\boldsymbol{\Sigma}_j^Q + \boldsymbol{R}\boldsymbol{\Sigma}_i^P\boldsymbol{R}^T\right)^{-1} d_{ij}^{(\boldsymbol{T})}}_{\text{= simplified Likelihood } L(\boldsymbol{T})}. \end{aligned} \tag{5.5}$$

The effect of minimizing Equation (5.5) is that corresponding points are not directly dragged onto another, but the underlying surfaces represented by the local covariance matrices $\boldsymbol{\Sigma}_i^P$ and $\boldsymbol{\Sigma}_j^Q$. The covariance matrices are computed so that they express the expected uncertainty along the local surface normals at the points, see Equation (5.4).

In Chapter 4. we have used this approach for 3D SLAM with a lightweight continuously rotating 3D laser scanner carried by a micro aerial vehicle (MAV). In order to compensate for smaller inaccuracies in pairwise registration, we used multiple edges between neighboring poses in the final trajectory optimization, where every single edge encoded a surface-to-surface error correspondence using the error metric in Equation (5.5). In contrast, in this chapter we no longer distinguish

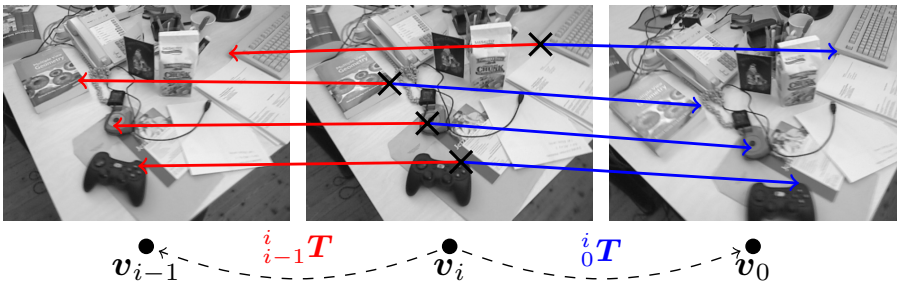


Figure 5.3: Multiple edge connections. Example of connecting a frame at vertex v_i to the last frame v_{i-1} and the first frame v_0. Instead of using a single edge encoding the transformations (dashed lines), we use one edge per point correspondence. Instead of repeatable features, we use raw points and iteratively refine the matching.

between pairwise initial alignment and subsequent global optimization. Instead, both the initial alignment in the local windows and the global trajectory optimization in case of loop closures are formulated in exactly the same multi-edge graph optimization approach.

5.3.5 *Multi-Edge Graph Optimization*

In a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, neighboring poses in the trajectory form the vertices $v_i \in \mathcal{V}$ and spatial constraints (transformations) between two vertices v_i and v_j are represented by edges $e_{ij} \in \mathcal{E}$. Instead of adding only a single edge between two vertices that encodes a transformation and the corresponding covariance matrix, we search for corresponding points between the respective point clouds and add multiple edges, one for each found correspondence (see Figure 5.3).

Each edge in the graph encodes two entities: a local contribution to the measurement error $\boldsymbol{e}$ and an information matrix $\boldsymbol{H}$ which represents the uncertainty of the measurement error. The information matrix is defined as the inverse of the covariance matrix, i.e., it is symmetric and positive semi-definite. For the error measurement between, respectively, two vertices v_i and v_j and the correspondence pair $(\boldsymbol{p}_{i,m}, \boldsymbol{p}_{j,n})$, we use the point-to-point difference vector and approximate its information matrix using the error generalized error metric from Equation (5.5):

$$\text{mean} \quad \boldsymbol{e}_{ij,mn}({}^{i}_{j}\boldsymbol{T}) = \boldsymbol{p}_{j,n} - {}^{i}_{j}\boldsymbol{T}\boldsymbol{p}_{i,m}, \tag{5.6}$$

$$\text{and} \quad \boldsymbol{H}_{ij,mn}({}^{i}_{j}\boldsymbol{T}) = \left(\boldsymbol{\Sigma}^{P_j}_{m} + \boldsymbol{R}\boldsymbol{\Sigma}^{P_i}_{n}\boldsymbol{R}^{T}\right)^{-1}. \tag{5.7}$$

The effect is that every edge contributes its approximate surface-to-surface error term to the system information matrix—thus automatically giving lower influence on incompatible or false correspondences and quickly leading to alignment even in case of larger initial displacements.

For the actual optimization, we follow an iterative procedure by 1. estimating correspondence pairs for all (or a subset of) points $\boldsymbol{p}_{i,m} \in P_i$ in P_j for every two vertices $(\boldsymbol{v}_i, \boldsymbol{v}_j)$ that are to be connected and 2. optimizing the resulting linearized system for a maximum of ten inner iterations. We repeat these two steps for a maximum of ten outer iterations. For a fast initial coarse alignment in early and an accurate refinement in later outer iterations, we use a linearly decreasing distance threshold for correspondence pairs, starting with 2 m (∞ in the first iteration) and going to two times the expected local noise as defined in Equation (5.1). In every outer iteration step, the graph is optimized using dense Cholesky decomposition and Levenberg-Marquardt within the g2o framework (Kümmerle *et al.*, 2011). For both inner and outer iterations, we stop when the system has converged. Convergence in graph optimization (inner iterations) can be detected based on the changes in both view poses and system error as well as the damping factor applied by Levenberg-Marquardt. For detecting convergence in the overall graph refinement in the outer iterations, we check whether the view pose connectivity and the correspondences between connected view poses have changed. When no more changes are found and the inner optimization has converged, we stop optimizing the trajectory.

5.3.6 *Local Window Alignment*

In order to estimate a rough initial pose estimate for a newly acquired frame, standard SLAM procedures would first register the new frame against the last (key) frame, and then search for possible loop closures for a subsequent global optimization of the estimated trajectory. Instead, we determine a local window of neighboring poses and simultaneously align the new frame against all frames acquired at poses within the local window. Referring to Figure 5.4, for a pose at $\boldsymbol{v}_i$ we search for closest poses (i.e., camera origins) in 3D. The found candidates are checked for a similar viewing direction by means of the angle between the camera z axes. This rough initial check suffices since the following alignment accurately deals with overlapping and non-overlapping measurement volumes. We define the local window to contain 1. the last acquired frame and the last acquired key frame, respectively, as well as 2. all poses within a radius r around the camera origin w.r.t. the current pose estimate that have a similar orientation. In order to obtain constant-time initial alignments, we additionally use an upper limit of w for the number of neighboring poses in the local window and sample the matches in between neighboring frames to keep both the number of vertices and the number of edges in the constructed subgraph constant.

Once the local window is determined, the newly acquired frame is aligned to all frames in the local window by estimating only the

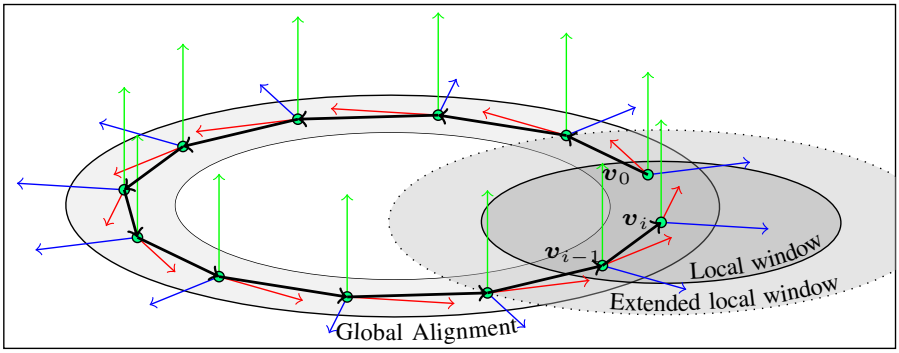


Figure 5.4: Local window alignment as opposed to pairwise alignments. Aligning a frame (x-axis red, y-axis green, and z-axis red) to multiple other frames tends to be more stable and to drift less. In this example of a camera moving along a circular trajectory, the local window around the vertex v_i includes the last frame (at v_{i-1}) and the starting pose v_0 upon loop closure.

pose of the frame being aligned. The other poses are fixed during optimization. In contrast to pairwise registration, this one-to-many alignment is more stable and tends to drift less (see the experimental evaluation in Section 5.4). Moreover, it allows for efficiently detecting loop closures and possible inconsistencies in the trajectory estimate as described next.

5.3.7 *Loop Closure Detection and Global Optimization*

Our primary mean for detecting loop closures is to inspect the poses found in the local window. Naturally, if a similar pose is found that has been acquired long ago (in terms of time and frame index), a loop closure is detected. Since our local initial alignment is quite stable without considerable drifts, loop closures can be easily detected as long as the camera trajectory is bound to a single room. In larger environments, drifts in the local alignments accumulate and more sophisticated means for recognizing previously visited places are needed, e.g., using visual features (Cummins and Newman, 2010; Lynen *et al.*, 2014; Lowry *et al.*, 2015), 3D features (Steder *et al.*, 2011; Magnusson *et al.*, 2009b; Bosse and Zlot, 2013), a combination of both, or even detected objects in the scene and relative poses in between them (Finman *et al.*, 2015).

Once, the local window contains an earlier pose along the trajectory, we compare the transformations obtained from the alignment in the local window with those in the so far built global graph. In case of conflicts, e.g., larger jumps in the estimated pose or no convergence in the optimization, we trigger an alignment in an extended local window. This extended window includes the 1-ring neighborhood of the local window. For the alignment, all poses in the local window are now optimized, and only the poses in the extended border (i.e., in the

1-ring neighborhood) are fixed. If the extended local window contains earlier poses or shows conflicts after local alignment, global trajectory optimization is triggered.

The global optimization of the trajectory follows the same principle as the local alignment. In order to save processing time, however, the global graph does not contain all poses but only a limited set of keyframes.

5.3.8 *Keyframe Selection*

Several strategies exist to select whether or not to add a new key frame, e.g., adding every n-th frame, applying rotational and translational thresholds (Konolige and Agrawal, 2008; Lim *et al.*, 2011; Strasdat *et al.*, 2011), or applying thresholds on registration error variances or the number of matched features (Henry *et al.*, 2012). Middelberg *et al.* (2014), for example, do not only inspect the number of inliers in the matching but also their distribution over the image, and add a new keyframe if the majority of inliers lie in the upper left, upper right, bottom left, or bottom right part of the image.

We apply a rotational threshold and a translational threshold as fixed upper limits in order to avoid larger distances between key frames even if the alignments in between are good. In addition, we use a measure based on matching quality and uncertainty along the dimensions of the transformation. After the local alignment of a newly acquired frame, we inspect the determined transformation $\boldsymbol{T}$ to the last keyframe and compute an estimate of the uncertainty. We use the approximation by Censi (2007) to compute the covariance matrix:

$$\boldsymbol{\Sigma}_{\boldsymbol{T}} \approx \left(\frac{\partial^2 L}{\partial \boldsymbol{x}^2}\right)^{-1} \frac{\partial^2 L}{\partial c \partial \boldsymbol{x}} \, \boldsymbol{\Sigma}(z) \, \frac{\partial^2 L}{\partial c \partial \boldsymbol{x}}^{T} \left(\frac{\partial^2 L}{\partial \boldsymbol{x}^2}\right)^{-1}, \tag{5.8}$$

where L is the simplified likelihood function in Equation (5.5), c denotes the individual found correspondences $\mathcal{C}$ between the two point clouds P_i and P_j, and $\boldsymbol{\Sigma}(c)$ is the covariance of the correspondence pairs. Stückler and Behnke (2014) use the same approximation to obtain a measure of matching quality and pose uncertainty. Note that in Equation (5.8), the relative transformation between two view poses is not represented as a homogeneous transformation matrix $\boldsymbol{T}$, but in a parameterized form $\boldsymbol{x} = (\boldsymbol{t}, \boldsymbol{q})^T$ with translation $\boldsymbol{t}$ and rotation by the unit quaternion $\boldsymbol{q} \in \mathbb{H}$. We then follow the approach of Kerl *et al.* (2013) to compute an entropy-based measure

$$H\left(\boldsymbol{T}, \boldsymbol{\Sigma}_{\boldsymbol{T}}\right) \propto \ln\left(\left|\boldsymbol{\Sigma}_{\boldsymbol{T}}\right|\right), \tag{5.9}$$

using the determinant of $\boldsymbol{\Sigma}_T$. The entropy of the current transformation (against the last key frame) is then compared to the stored entropy of the last key frame (when it was added). If the ratio between the two entropy measures falls below a predefined threshold, the last (not

the currently aligned) frame is added as a key frame, and the local alignment is repeated. A similar strategy is followed by Lim *et al.* (2014) but with direct thresholding and computing statistics on the Fisher information.

5.3.9 *Point Subsampling and Correspondence Filtering*

An important aspect in the alignment is determining correspondences between frames. Every such correspondence will contribute in the form of an edge to both subgraph and global graph optimization. In order to use only a small number of correspondences without loosing much information, we follow a multi-stage strategy: we first subsample query points from the frame to be aligned and then reject found correspondences that are unlikely to contribute to the alignment.

We sample query points from two distributions: uniformly over the rows and columns of the image and uniformly in normal space. The intuition behind the latter is that we want to draw samples from all surface orientations in the scene so as to robustify the alignment along all dimensions. Note that the sampled set of query points is stored for later correspondence searches if the frame is added as a keyframe.

In order to search for corresponding points in the other frames, we first project each query point into the camera coordinate frame of the target frame and check whether it is visible by frustum culling (May *et al.*, 2009). If the query point lies within the view frustum of the target frame, we use the closest point in the smoothed mesh of the target point cloud. The resulting set of correspondences is then filtered again 1. to remove false correspondences that can negatively affect the alignment, and 2. to further reduce the number of correspondences. We remove correspondences that include surface boundary points (e.g., introduced by occlusions), and apply filters on the residual correspondence pairs that remove 1. pairs whose point-to-point distance exceeds the median point-to-point distance over all correspondences, 2. pairs whose local surface normal orientations considerably deviate, and 3. pairs that contain the same matching point in the target frame. In the latter case, only the pair with the smallest point-to-point distance is kept. The local surface normals are considered to avoid that points with normals pointing in opposite directions form a correspondence.

5.4 EXPERIMENTS AND RESULTS

In order to assess the performance of our approach, we use the datasets and error metrics from the publicly available RGB-D SLAM Benchmark[1] by Sturm *et al.* (2012).

1 http://vision.in.tum.de/data/datasets/rgbd-dataset

	Pairwise Mesh Registration*			Mesh Registration + Filtering** ($n=4$)				Local Alignment*** + Filtering ($n=4, k=4$)				
Dataset	$n=1$	$n=2$	$n=4$	$k=1$	$k=2$	$k=3$	$k=4$	$w=2$	$w=3$	$w=4$	$w=5$	$w=10$
fr1_xyz	0.204	0.216	0.199	0.192	0.172	0.159	0.156	0.119	0.087	0.068	0.052	0.041
fr1_rpy	0.197	0.205	0.189	0.181	0.175	0.160	0.162	0.127	0.101	0.099	0.081	0.059
fr1_desk	0.303	0.313	0.324	0.235	0.218	0.207	0.201	0.165	0.138	0.108	0.094	0.084
fr1_desk2	0.457	0.436	0.429	0.324	0.301	0.281	0.256	0.152	0.129	0.117	0.102	0.096
fr1_room	0.383	0.375	0.390	0.281	0.262	0.239	0.241	0.189	0.148	0.120	0.098	0.084
fr1_360	0.831	0.802	0.806	0.651	0.612	0.495	0.492	0.271	0.229	0.191	0.171	0.162
fr1_teddy	0.102	0.148	0.114	0.101	0.099	0.098	0.098	0.089	0.085	0.081	0.078	0.078
fr1_plant	0.143	0.152	0.141	0.135	0.136	0.131	0.131	0.111	0.083	0.074	0.059	0.052
fr2_desk	0.198	0.192	0.201	0.152	0.148	0.138	0.139	0.091	0.072	0.051	0.039	0.030
fr3_long_office_...	0.103	0.124	0.101	0.093	0.093	0.091	0.089	0.071	0.060	0.055	0.041	0.034
Avg. improvement	—	−1.5%	1%	20%	24%	31%	32%	52%	61%	66%	72%	75%

* Mesh Registration: the input images are subsampled by using only every n-th row and column, e.g., $n = 4$ corresponds to a 160×120 image.
** Filtering: the local neighborhood used by the multilateral filter is sequentially expanded to include the 1 to k-ring neighrborhoods.
*** Local alignment: the window size w determines the number of (closest) vertices used for optimization of the subgraph.

Table 5.1: Relative pose error (RPE) in initial alignments (without global optimization), RMSE of RPE (Δ) in m/s with $\Delta = 1\,\mathrm{s}$

5.4.1 *Accuracy of Local Alignments*

For measuring the drift in initial alignments (without global optimization), we use the relative pose error (RPE) proposed by Sturm *et al.* (2012). It computes the root mean square error (RMSE) of the translational errors between the estimated poses and the corresponding ground truth poses in an interval Δ. We use $\Delta = 1\,\mathrm{s}$ to measure the drift in meters per second (m/s). For different datasets, we compare the results in terms of the RPE for different processing steps and parameters. In particular, we focus on the effect of subsampling the input image (in order to reduce processing time), filtering the approximated surfaces (to smooth the underlying data), and the size of the local window. We report all results in Table 5.1. Note that in these experiments, global trajectory optimization is disabled.

For all processing steps and parameter sets, we computed the average translational drifts and compared them with the average drift of the plain pairwise registration as a baseline to obtain an average improvement. The subsampling experiments indicate that, since both the query points and the found correspondences are already drastically sampled, the effect of subsampling the input image in the course of approximate surface reconstruction is only minor. For this reason, we have chosen to process 160×120 images ($n = 4$) to reduce computations in the pre-processing steps which are conducted for every single frame.

In contrast, smoothing the underlying data (and thus also the surface normals used in the alignment) significantly improves the alignments and reduces the translational drift. In most of the datasets, a larger portion of the data is not sensed on the object of interest (e.g., the teddy, the plant, or the desks), but on environmental structures such as walls farther away from the sensor. Therefore, the respective depth measurements are more affected by quantization effects. With an increasing smoothing factor (the included k-ring neighborhoods), the multilateral filter can effectively smooth over the emerging depth discontinuities while preserving edges. We achieved the best results with $k = 4$ and suggest to not use ring neighborhoods farther away (i.e., $k \geq 5$), since especially the local surface normals become too inaccurate and details will be smoothed away. Overall, an improvement of roughly 30 % can be achieved for pairwise registration if the data is smoothed before alignment.

Compared to pairwise registration that only uses the last keyframe (i.e., $w = 1$), using a local window for the alignment drastically reduces the drift. The more other frames are used in the alignment, the more stable and accurate the estimated pose becomes. However, this improvement comes at the price of optimizing a larger subgraph (see Figure 5.5). Since the average improvement does not considerably

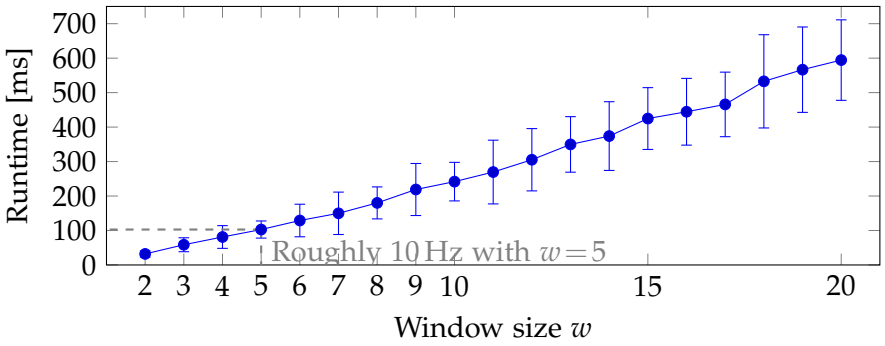


Figure 5.5: Average runtimes of the local alignment (per frame), measured over all datasets (with 10 complete runs per dataset) on a single core of an Intel Core i7-3740QM CPU (2.70GHz).

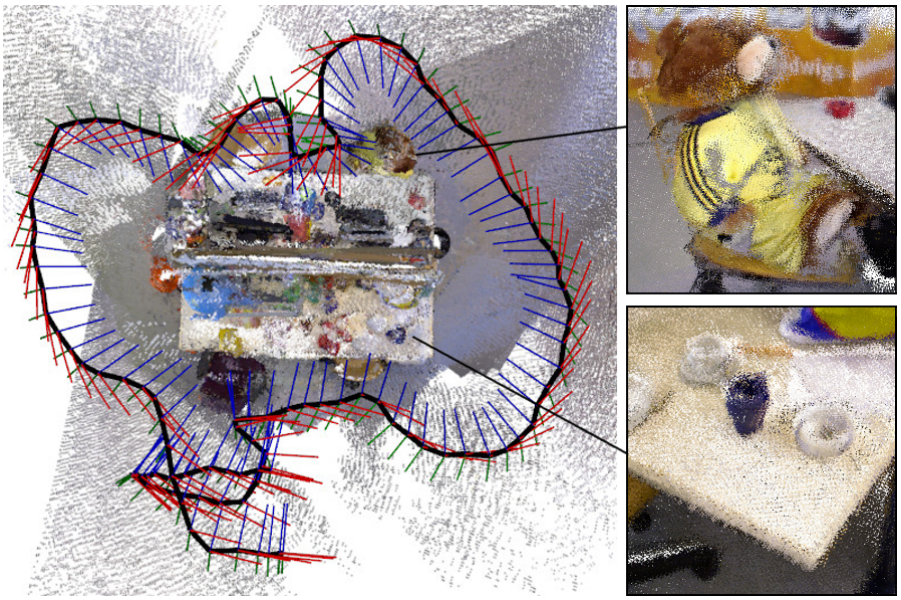
increase for larger local windows, we use a window size of $w = 5$ as it allows locally aligning new frames at roughly 10 Hz.

5.4.2 *Trajectory Optimization and Global Alignment*

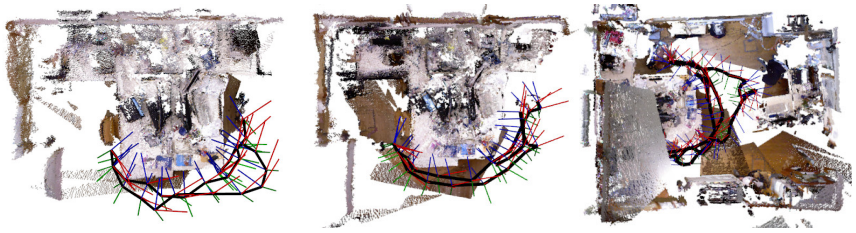
In a final series of experiments, we used our complete pipeline with global trajectory optimization enabled. We subsample the point clouds with $n = 4$ in the approximate surface reconstruction, include all local neighbors for smoothing up to ring $k = 4$, and initially align newly acquired point clouds in a local window of size of $w = 5$. For measuring the errors in the final optimized trajectory, we use the absolute trajectory error (ATE) (Sturm *et al.*, 2012). It determines a transformation best aligning the estimated trajectory and the ground truth trajectory in order to compute errors between individual frames regardless of the used base coordinate frame.

In the first series of experiments, We compare both our local and our global alignment approaches to DVO-SLAM by Kerl *et al.* (2013), MRSMAP by Stückler and Behnke (2014), RGB-D SLAM by Endres *et al.* (2012b) and Endres *et al.* (2014) and KinFu, the open source implementation of KinectFusion by Newcombe *et al.* (2011) which is available in the Point Cloud Library PCL. Since we have not been able to produce better results than Kerl *et al.* (2013) in our experiments, we compare against the values reported. As in the case of the translational drift, we report the RMSE in Table 5.2 with example results in Figure 5.6.

Naturally, our multi-view registration approach cannot outperform sophisticated state-of-the-art dense visual SLAM methods specifically designed for RGB-D data. For example, both DVO-SLAM (Kerl *et al.*, 2013) and MRSMAP (Stückler and Behnke, 2014) apply a coarse-to-fine registration on different resolutions. In contrast, we use only the initially sampled subsets of points and the found matches during the alignment. Hence, our alignment neglects details such as smaller objects on the tables or texture details in general. This is reflected in a

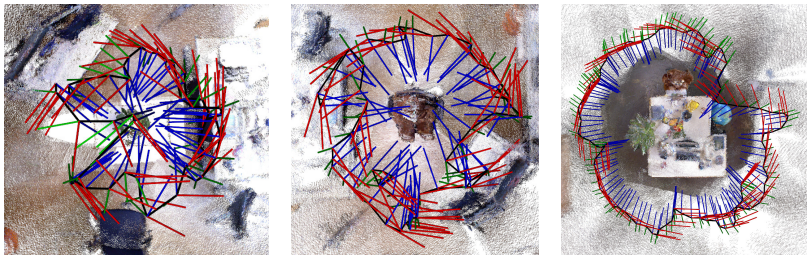


(a) fr3_office with detail views of a larger object (bear) and smaller objects (mugs)



(b) fr1_desk (c) fr1_desk2 (d) fr1_room

(e) fr1_xyz (f) fr1_rpy (g) fr1_360



(h) fr1_plant (i) fr1_teddy (j) fr2_desk

Figure 5.6: RGB-D SLAM results. Shown are the acquired RGB-D point clouds aligned using the determined poses (top view). The poses are visualized using the respective coordinate frames (x-axis red, y-axis green, and z-axis blue). In addition, we visualize the estimated camera trajectory as a black line connecting consecutive poses. All maps and trajectories are globally consistent.

Dataset	Ours (local)	Ours (global)	DVO-SLAM*	MRS-MAP*	RGB-D SLAM*	KinFu*
fr1_xyz	0.051	0.013	0.011	0.013	0.014	0.026
fr1_rpy	0.131	0.028	0.020	0.027	0.026	0.133
fr1_desk	0.052	0.028	0.021	0.043	0.023	0.057
fr1_desk2	0.081	0.039	0.046	0.049	0.043	0.420
fr1_room	0.142	0.073	0.053	0.069	0.084	0.313
fr1_360	0.254	0.082	0.083	0.069	0.079	0.913
fr1_teddy	0.082	0.090	0.034	0.039	0.079	0.154
fr1_plant	0.051	0.025	0.028	0.026	0.091	0.598
fr2_desk	0.090	0.046	0.017	0.052	—	—
fr3_long_..	0.043	0.037	0.035	—	—	0.064
Average	0.097	0.046	0.034	0.043	0.054	0.297

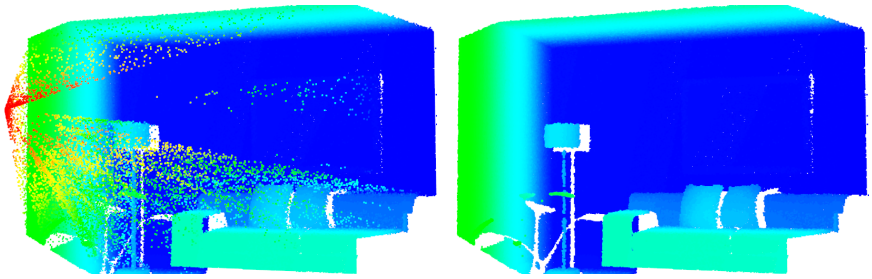
*Results as reported by Kerl *et al.* (2013).

Table 5.2: Absolute trajectory error (ATE) in comparison, RMSE in m

slightly higher ATE for our approach(es). The higher error stems from minor local inaccuracies in the overall globally consistent trajectory estimates. Still, we get a better ATE in two datasets (fr1_desk2 and fr1_plant).

While, on average, DVO-SLAM (Kerl *et al.*, 2013) and MRSMAP (Stückler and Behnke, 2014) outperform the other approaches, our global alignment comes in third and achieves a lower average ATE than both RGB-D SLAM (Endres *et al.*, 2012b) and KinFu. Most notably, however, is that our local alignment approach (without global optimization), achieves very good initial trajectory estimates without the need of optimizing more than a local window. Hence, we believe that our approach has a large potential for inspiring related approaches and for contributing to other SLAM pipelines.

In a second series of experiments, we have used the synthetic dataset of Handa *et al.* (2014). It contains four sequences of simulated RGB-D images in the a synthetic living room scene. Two versions of each sequence are available, one with the original simulated RGB-D images (used here for visualization purposes) and one with simulated noise that is used in the evaluation. We show a typical point cloud with simulated noise (1st point cloud from the sequence kt0) in Figure 5.7 together with the filtered point cloud extracted from the approximated mesh. As can be seen, initially the point clouds are quite noisy and contain a considerable amount of jump edges and spurious measurements at depth discontinuities respectively. The cloud extracted from the approximate mesh is considerably smoother and does not contain any spurious measurements along jump edges. Points caused by depth



(a) Noisy point cloud (b) Mesh vertices

Figure 5.7: Noise and filtering: example of a noisy point cloud (a) and a filtered cloud (b). The validity checks in the surface reconstruction reliably filter out the jump edges, while the multilateral filtering efficiently smooths the data.

discontinuities are reliably filtered out by the edge validity checks in the mesh reconstruction while the local noise is smoothed by the multilateral filter. As a result, the difference between the rendered and the noisy point is negligible.

Since they are simulated in a synthetic environment, the datasets can serve different purposes ranging from visual odometry over SLAM to surface reconstruction. As in the case of the RGB-D-SLAM Benchmark by Sturm *et al.* (2012), we focus on the global trajectory error (ATE) to estimate the accuracy of our trajectory estimates. We compare our results to DVO-SLAM by Kerl *et al.* (2013), RGB-D SLAM by Endres *et al.* (2012b) and Endres *et al.* (2014), MRSMAP by Stückler and Behnke (2014), Kintiniuous by Whelan *et al.* (2012), and ElasticFusion by Whelan *et al.* (2015). We report the detailed results for the four sequences in Table 5.3. In addition we show visualizations of the aligned point clouds in Figure 5.8 and Figure 5.9. While we used the sequences of noisy point clouds in the experiments, we used the rendered point clouds without noise for visualization purposes in Figures 5.8 and 5.9. For comparison, we included all results reported by Whelan *et al.* (2015). In addition, we use the publicly available implementations of DVO-SLAM, RGB-D SLAM, and MRSMAP (in contrast to Kintinuous and ElasticFusion all CPU-only). Since in some of the implementations the focal length is hard-coded, we patched them to include the proper focal lengths for the datasets (481.20 for the x-axis and -480.00 for the y-axis). Furthermore, since neither of the approaches explicitly handles jump-edges, we included our approximate surface reconstruction and the bilateral filtering to remove spurious points along jump edges and to locally smooth the measurements. As can be seen in the table, especially for the MRSMAP-based approach considerably better results can be achieved using these pre-processing steps.

Whereas ElasticFusion and Kintinuous achieve the best results in terms of the ATE, it is to be noted that our approach with the global optimization compares well and, in particular, is not considerably worse

		Absolute trajectory error (RMSE) in m			
	Method	kt0	kt1	kt2	kt3
Literature*	DVO SLAM	0.104	0.029	0.191	0.152
	RGB-D SLAM	0.026	0.008	0.018	0.433
	MRSMap	0.204	0.228	0.189	1.090
	Kintiniuous	0.072	0.005	0.010	0.355
	ElasticFusion	0.009	0.009	0.014	0.106
Impl.	DVO SLAM**	0.102	0.031	0.186	0.141
	RGB-D SLAM**	0.026	0.015	0.019	0.257
	MRSMap**	0.072	0.051	0.123	0.259
Ours	Local alignment	0.462	0.251	0.249	0.531
	Global alignment	0.082	0.043	0.176	0.148

*As reported by Whelan *et al.* (2015).
**With approximate surface reconstruction and bilateral filtering.

Table 5.3: Results for the synthetic dataset by Handa *et al.* (2014) as reported in the literature (top), with public CPU-only implementations with pre-processing using approximate surface reconstruction and multilateral filtering (middle) and ours.

than the other approaches. With the local alignment alone (global optimization disabled), poses can be reliably determined in smaller windows, but the overall trajectories tend to show inconsistencies reflected in a high ATE. With global optimization enabled, the approach lines up with the other methods with an average performance, i.e., neither being the best nor the worst method for a particular dataset sequence. For the sequences kt0, kt1, and kt2 we obtain accurate pose estimates and globally consistent maps of the living room. If at all, minor inconsistencies arise, e.g., in the kt0 sequence where several frames only show a single wall, i.e., no distinctive geometric structure. Since our method is purely based on 3D measurements and neglects color in the registration the television screen is not correctly mapped (see Figure 5.8). In case of kt1 and kt2 not even minor inconsistencies arise and both maps appear correct (see Figure 5.9). Only in case of kt3 a major inconsistency arises at the upper wall where the loop in the estimated trajectory is not correctly closed. In our current implementation, we only use 3D point matches (i.e., closest points in 3D space). It is expected that additionally matching visual keypoints and feature descriptors extracted in the RGB image and augmenting the multi-edge graph with the found matches will considerably increase the accuracy in situations as the one described above. That is, the surface-to-surface correspondences also work in scene geometries without textures while the keypoint-to-keypoint matches also work in scenes with texture but

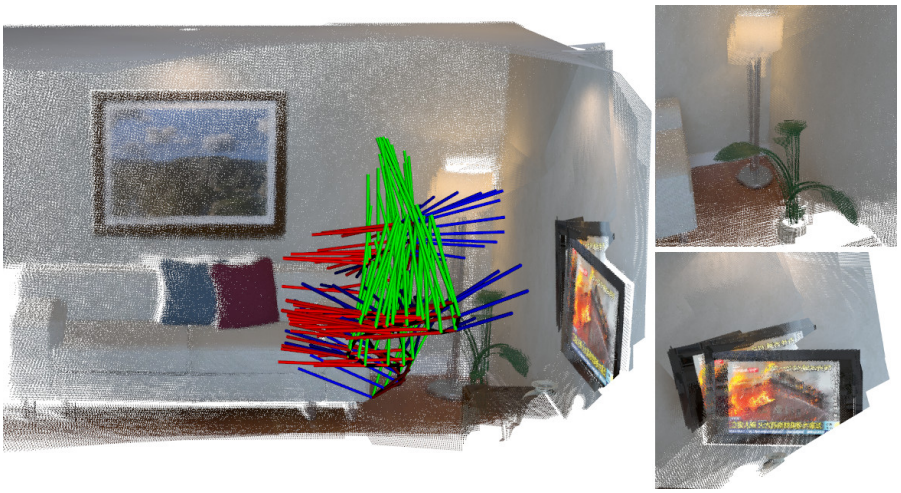
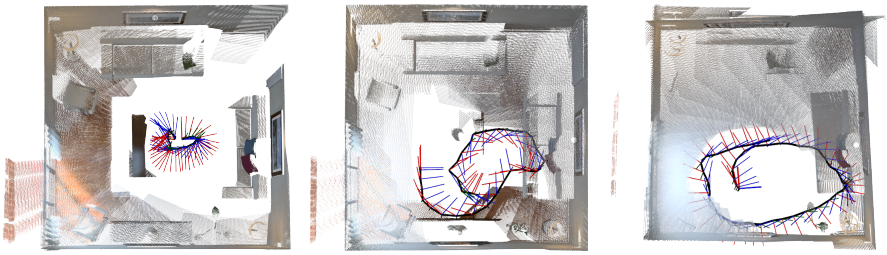


Figure 5.8: Result for the kt0 dataset: shown are the aligned point clouds and the estimated trajectory (left) as well as detail views showing minor inconsistencies (right).



(a) kt1 (b) kt2 (c) kt3

Figure 5.9: Results for datasets kt1, kt2 and kt3. Although the ATE for kt2 is higher, kt3 is the only sequence for which we do not obtain a globally consistent map.

low geometrical detail. However, it remains a matter of future work to evaluate such a combination.

5.5 LOCAL WINDOW ALIGNMENT FOR MAPPING WITH MAVS

In Chapter 4, newly acquired 3D laser scans where first initially registered against the scan and then globally aligned with all acquired scans. With the expected result of improving the accuracy of initial alignments, in this series of experiments the pairwise registration from Section 4.3.5 is compared to the local window alignment presented in this chapter. For this purpose, we use two of the datasets from Chapter 4—the dataset recorded in the *motion capture volume* and the dataset of the *Frankenforst facade*—and compute trajectory estimates using 1. pairwise registration using approximate covariance estimates (without global optimization), 2. local window alignment (without global optimization), and 3. global optimization of the trajectory ob-

tained from pairwise registration. As parameters for the local window alignment, we do not use subsampling in the approximate surface reconstruction ($n = 1$) and a fully connected mesh (i.e., no distance and angle checks) in order to process all data points. Furthermore, we do not use bilateral filtering ($k = 0$) since the points are already very accurate compared to the noisy RGB-D images. As a side note is to remark that filtering only makes sense within scan lines and not over several scan lines. For the alignment, we use a local window size of $k = 10$, i.e., a newly aggregated point cloud is aligned to the closest 10 other point clouds.

For comparison, we also include the results reported in an experimental evaluation of different registration algorithms for registering the acquired 3D scans in local egocentric maps (Razlaw *et al.*, 2015). The egocentric maps are represented as local multi-resolution surfel maps (Stückler and Behnke, 2014; Droeschel *et al.*, 2014c). They model the surroundings of the MAV in a grid in which the grid cell sizes increase with an increasing distance from the MAV. Each grid cell represents the underlying surface with a *surfel*—a Gaussian representing the position, orientation and flatness of the sensed environmental structures falling into the cell. Registering newly acquired scans with the so far built map, and incrementally extending and updating the map with the aligned scan is naturally more robust than pairwise registration (see the SLAM approach using ICP-based incremental registration by Holz and Behnke, 2010). The incremental registration algorithms under comparison are ICP, GICP, normal distributions transform (NDT) and the surfel registration of Droeschel *et al.* (2014b). Furthermore, we include the accuracy of the initial trajectory estimate obtained from visual odometry (VO). Whereas the visual odometry trajectory estimate has been used as is in the evaluation, for the other registration algorithms parameters were optimized with respect to the ATE in the motion capture volume dataset using hyper parameter optimization (see Razlaw *et al.*, 2015). The optimized parameters for each algorithm include, amongst others, the maximum number of iterations and other termination criteria, the distance thresholds between corresponding points, and the resolution and the number of levels in the built multi-resolution surfel map. In addition, algorithm-specific parameters such as the grid resolution in NDT or the parameters for the soft assignments in the surfel-based registration of Droeschel *et al.* (2014b).

Since no ground truth information is available for the outdoor dataset recorded at *Gut Frankenforst*, we focus the comparison on two measures of map quality and the average runtime for processing one range scan. The map quality measures include the mean map entropy from Section 4.5 and a ground truth registration error. The latter is obtained by first creating a ground truth map of the environment. For the motion capture volume, a dense 3D laser scan was recorded

	Method	Ground truth Reg. error [m]	Map entropy mean []	Runtime* mean [s]
Motion capture volume	VO	0.005 840	−2.520 13	—
	ICP	0.003 698	−3.6524	1.665 ± 0.386
	GICP	0.003 100	−3.4111	1.353 ± 1.192
	NDT	0.002 149	−3.7414	5.249 ± 1.736
	Surfel	0.002 039	−3.8087	0.046 ± 0.014
	Pairwise reg.	0.003 212	−3.4135	0.030 ± 0.010
	Local window	0.001 927	−3.8058	0.109 ± 0.038
	Global opt.	0.001 919	−3.8096	0.492 ± 0.103
Frankenforst facade	VO	0.272 254	−2.330 82	—
	ICP	0.066 056	−2.654 60	1.026 ± 0.382
	GICP	0.064 002	−2.555 05	0.319 ± 0.235
	NDT	0.069 701	−2.633 96	2.062 ± 0.712
	Surfel	0.093 749	−2.813 87	0.350 ± 0.152
	Pairwise reg.	0.091 376	−2.490 55	0.039 ± 0.013
	Local window	0.042 143	−2.723 91	0.131 ± 0.041
	Global opt.	0.042 096	−2.731 82	0.586 ± 0.172

* Runtimes are measured per 3D scan being registered.

Table 5.4: Map quality and runtime evaluation results.

over several full rotations of the scanner while standing still. For the Frankenforst dataset, a highly accurate georeferenced 3D point cloud was recorded by the Institute of Geodesy and Geoinformation at the University of Bonn. The first 3D point cloud of each dataset was then registered to the ground truth map (manual initial alignment and registration-based fine alignment) in order to obtain a common reference frame for comparison. After aligning all point clouds using the different approaches, the resulting map obtained from all aligned point clouds is registered against the ground truth map using ICP. The obtained RMSE from this registration provides a good measure of map correctness and can reveal both minor inaccuracies and major inconsistencies. All results are shown in Table 5.4.

As expected, the accuracy of aligning a newly aggregated point cloud in the local window is considerably better than for the pairwise registration. In fact, the subsequent global optimization is only slightly better in both map quality measures. Visually inspecting the results shows no differences between the aggregated clouds. That is, if processing time is limited and the area to be mapped is small, local window alignment is already sufficient to produce highly accurate and globally consistent maps. For larger environments, however, it is

expected that global optimization becomes necessary just as for the RGB-D sequences in Section 5.4.2.

5.6 CONCLUSIONS

We have presented a complete pipeline for aligning pairs and sequences of RGB-D images. Our approach is based on approximating the underlying surface in the form of a quad mesh, and using the mesh for fast feature estimation (normals, covariances, etc.) and edge-aware smoothing. The alignment makes use of graph optimization with multiple edges between vertices, where every edge encodes a surface-to-surface error constraint, inspired by the Generalized-ICP algorithm (Segal *et al.*, 2009). In order to reduce computation time, we efficiently subsample both points and used correspondences between frames for almost constant-time local alignments. Our experiments show that aligning newly acquired images in a local window as opposed to pair-wise alignment with the last frame reduces drift and achieves low relative pose estimation errors. Optimizing the complete graph after loop closures and as a last processing step yields globally consistent alignments and trajectory estimates.

In experiments, we could show that our approach is competitive with state-of-the-art approaches in terms of pose estimation accuracy. Naturally, our approach cannot be as good as sophisticated (RGB-D) SLAM approaches in terms of detail and speed (e.g., compared to the multi-resolution surfel maps of Stückler and Behnke (2014) or the dense models based on signed distance functions). However, our approach, and especially the surface-based alignment in local windows, have a huge potential for contributing to other SLAM frameworks. Moreover, the multi-edge approach can be easily extended to include point-to-point correspondences, e.g., from matching visual or 3D features.

In another set of experiments, the presented approach could successfully be applied to the 3D laser scanner data captured by a MAV in the previous chapter. By aligning newly aggregated point clouds against a local window of point clouds instead of only the previous point cloud, the resulting transformations were considerably more accurate than those achieved with pairwise registration. At the same time, keeping both the number of point clouds in the local window and the number of correspondences between connected point clouds constant allows for near constant time updates and, overall, real-time local alignment of point clouds. Compared to other approaches, the resulting alignment achieves a good trade-off between processing time and pose accuracy.

In its current implementation, our approach distinguishes only local and global alignment. A logical next step would be to extend it to a completely hierarchical approach (Olson, 2009; Zhang *et al.*, 2012a)

with the complete trajectory and subgraphs on higher levels, and sub-regions and single points in images on lower levels so as to allow the alignment to correct individual measurements and inaccuracies within 3D point clouds.

6 DISCUSSION AND CONCLUSION

The cornerstones of this thesis are *simplicity* and *efficiency*. The Stanford Encyclopedia of Philosophy[1] distinguishes two principles of simplicity in the context of Ockham's Razor: 1. the epistemic principle (*"if theory T is simpler than theory T*, then it is rational to believe T rather than T*."*), and 2. the methodological principle (*"if T is simpler than T* then it is rational to adopt T as one's working theory for scientific purposes."*). Whereas *"cases where competing hypotheses explain a phenomenon equally well are comparatively rare"* in many fields of science (Holsinger, 1981; Baker, 2007), technical problems where competing methods solve a problem equally well are quite common especially in robotics and computer vision research. Moreover, publicly available datasets and benchmarks allow to directly compare methods for a particular task, situation, and input. This comparability of methods makes it possible to identify the best method available for a given task and given criteria.

A particularly important criterion for methods in the domain of robotics research is efficiency, i.e., not only producing the expected results but also producing these results as fast as possible in order to avoid longer interruptions in the robot's workflow. Following the definition of Merriam-Webster[2], efficiency is *"the ability to do something or produce something without wasting materials, time, or energy"*. With the above in mind, Ockham's razor can be re-interpreted as

> *if method M is equally performing but faster than M* for a problem it is rational to use M to solve it.*

Since the runtime of (software) components primarily depends on the complexity of the implemented algorithms, the underlying idea of the thesis was to find solutions which are particularly simple and efficient by 1. minimizing both the number of computations and the amount of data being processed, as well as 2. using methods of particularly low computational complexity for all processing steps. Consequently, all methods presented in this thesis share several re-appearing components such as subsampling the input data where possible, focusing processing on relevant regions and subsets of data, and using efficient approximations where possible. Furthermore, the addressed problems have been carefully analyzed to deduce assumptions and simplifications. Whereas these assumptions may easily fail for general problems in the respective domains, they are valid for

1 Stanford Encyclopedia of Philosophy: `http://plato.stanford.edu`
2 "Efficiency." Merriam-Webster.com. Accessed January 7, 2016.

the particular problem being addressed and the environment and situation in which the problem has to be solved.

Three problems have been addressed in the context of this thesis:

1. Object perception for mobile manipulation, where the main task is to identify regions in three-dimensional (3D) input data that belong to task-relevant objects,

2. Scene segmentation, where the main task is to segment 3D input data into planes and other geometric primitives, and

3. Registration and mapping, where the main task is to align 3D data acquired from different positions and orientations, and to build 3D maps of the environment.

Object Perception for Mobile Manipulation

For detecting regions of interest and potential object candidates in mobile manipulation tasks, a real-time 3D processing pipeline is presented in Chapter 2. The pipeline exploits the organized image-like data structure and the assumption that (manipulable) objects are located on top of horizontal support surfaces such as tables, shelves, or the ground. Using a particularly efficient approximation of local surface normals the data is first enriched with information about the local orientation of the sampled surfaces. Processing then focuses on points with vertical surface normals since these are most likely being sampled on horizontal surfaces. After detecting the dominant horizontal planes in the extracted points, all points in the input data (except for the inliers of the found planes), are checked for lying above the found planes and within the planes' boundaries. Clustering these points results in image regions where we assume manipulable objects to be. Whereas this result is provided in the original resolution of the input data, the processing steps in between work on subsampled data in order to considerably reduce the amount of computations. The pipeline also features several pre-processing steps such as the subsampling but also filters to cope with special error and noise characteristics of the used 3D cameras. In addition, post-processing steps and potential applications are presented where the pipeline is used to detect, localize, and grasp objects.

Particular achievements of this approach are that table top segmentation pipelines designed for accurate high-resolution laser scanning (Rusu, 2009) have been made applicable to particularly noisy time-of-flight (ToF) cameras as well as consumer color and depth (RGB-D) cameras, and that the pipeline allowed for real-time object detection. In the reported applications, the pipeline was used for real-time object perception and grasp planning for unknown objects (Stückler *et al.*, 2013b), and for detecting and grasping automotive parts on pallets for automating kitting tasks (Holz *et al.*, 2015b).

Scene Segmentation

In order to compute complete segmentations of 3D input point clouds, another particularly efficient pipeline is presented in Chapter 3. The underlying idea of the pipeline is to approximate and segment the sampled surface. In the first step, an approximate surface reconstruction is deduced from the organized measurement topology in depth camera images. Local features such as surface normals are approximated directly on the approximated surface. In the next step, both the points and the features of the surface are smoothed using an edge-aware multilateral filter. Finally, the filtered surface is segmented using region growing. Different models are presented that allow for segmenting the scene into planar patches, locally smooth regions, or other geometric primitives such as cylinders and spheres. Experiments have shown that the pipeline compares well to state-of-the-art segmentation methods in terms of segmentation accuracy while allowing to compute segmentations at high frame rates.

Especially planar segmentations are particularly useful as an efficient representation of the scene as a compound of dominant planar environmental structures as well as for mapping and place recognition using such compounds (Fernández-Moral *et al.*, 2016).

Registration and Mapping

A wide variety of tasks such as navigation and manipulation planning require for more information about the environment than is contained in a single image or 3D point cloud. It becomes necessary to acquire data from different positions and orientations for a complete coverage of the scene. In order to register multiple point clouds in a common coordinate frame and to build a complete 3D map, the thesis presents different methods for different types of input data.

Particularly challenging is the registration of point clouds acquired by a fast rotating 3D laser scanner due to the different point densities in the cloud: a high density within the individual scan lines and a larger angular region without measurements in between. There are two ways for registering point clouds with non-uniform point densities: 1. aggregating multiple point clouds in local maps in order to increase the point density, or 2. using alternative means for registration that can cope with the non-uniform densities. Both strategies constitute problems in their own right.

The former strategy is followed by Droeschel *et al.* (2014b) who build local egocentric maps in which newly acquired point clouds are aggregated using either visual odometry as a relative pose estimate or registration once the map contains information of a certain number of point clouds, i.e., the density is higher. Registering the

local maps allows for building globally consistent maps of the whole environment (Droeschel *et al.*, 2014c).

In this thesis, we addressed the latter strategy. To compensate for the non-uniform densities, a pipeline is presented in Chapter 4 that uses the approximate surface reconstruction from the scene segmentation pipeline for approximating the sampled surface from the measurement topology and computing registration-relevant features on the deduced surface mesh. The point clouds are then aligned using a robust surface-to-surface error metric and using the features computed on the mesh. The pipeline allows for reliably registering the acquired point clouds even when the scanner is rotating fast and the angle between scan lines gets large (e.g., 15°). For creating consistent maps, the initial pose estimates from the registration are globally optimized. The optimization uses point correspondences between point clouds and the same surface-to-surface error metric as in the initial registration. Experimental results show that the pipeline can reliably register scans and build maps from laser scans acquired in-flight with a micro aerial vehicle (MAV). Moreover, a comparative experimental evaluation (Razlaw *et al.*, 2015) has shown that the two strategies and methods, i.e., the approach of Droeschel *et al.* (2014b) and the method presented in Chapter 4, compare well in terms of both registration error and runtime.

For registering sequences of RGB-D images, an extension of the mapping pipeline is presented in Chapter 5 that features additional steps to reduce the amount of data, the amount of correspondences between point clouds, and the amount of point clouds during (initial) registration in order to gain efficiency and make the pipeline applicable to RGB-D cameras. After computing and smoothing the approximate surface reconstructions and the local features, point clouds are initially aligned in local windows of neighboring poses and point clouds, respectively. In case of loop closures or after processing all point clouds, the graph of poses is completely optimized to obtain a globally consistent 3D map of the environment. Experiments have shown that both the initial alignment and the final optimization achieve reliable and accurate registrations that compare well with state-of-the-art methods. Moreover, even without the global optimization the local window alignment allows accurate and consistent 3D mapping for shorter sequences making it possible to register RGB-D images at 10 Hz.

In all of the aforementioned approaches it could be shown that by carefully analyzing the problem at hand, and deducing common assumptions and simplifications simple but efficient approaches can be derived that do not only compare well to state-of-the-art approaches but are also particularly efficient. All approaches can process 3D point clouds at high frame rates without parallelization or the use of a graphics processing unit (GPU).

Outlook and Future Work

In this thesis, the selected problems have been addressed on their own. Possible starting points for future work are to combine the methods and findings for addressing new applications and problem domains. The planar segmentation of the scene (Chapter 3), for example, can be coupled with the surface-to-surface alignment (Chapter 4) in order to align 3D point clouds solely based on the dominant planar surfaces in the scene and to build planar maps as a particularly efficient representation especially for man-made environments. Registration of 3D planar patches has been done, for example, by Pathak *et al.* (2010) and was found to be particularly efficient and robust. A very recent work going into this direction, which already uses the planar segmentation presented in this thesis, is the approach of Fernández-Moral *et al.* (2016). Fernández-Moral *et al.* segment the scene into planes and represent the connectivity of planes in a graph. Matching the graphs and subgraphs of connected planes allows them to align the 3D point clouds and also to recognize the scene and detect loop closures.

Another interesting line of thought is to combine the object segmentation with registration and mapping so as to build semantic object maps. Rusu *et al.* (2009b), for example, build semantic maps in household environments that, in addition to furniture and environmental structures, explicitly map objects using either geometric primitives or triangular meshes. Silberman *et al.* (2012) do not only detect the surfaces and objects in scenes but also deduce physical support relations. However, Silberman *et al.* only work on single color and depth images, and do not yet build complete 3D maps of surfaces, objects, and support relations. A first work into this direction in the context of this thesis was to combine the registration of 3D point clouds with segmenting support surfaces and objects in order to classify points and represent them with different resolutions in the final 3D map (Wurm *et al.*, 2011). In the voxel-based octree representation, objects were modeled in a high resolution of only few millimeters, the tables with a resolution of 1 cm and the background in a resolution of 5 cm. That is, the space-efficient final representation focused on objects while keeping obstacle-relevant information about support surfaces and other environmental structures. However, the objects were not tracked over time in order to update the respective parts of the model. A recent work in this direction is the approach of Stückler and Behnke (2015) who segment the scene and simultaneously build a 3D map in which coherent parts can be individually moved and updated.

Yet another application of the presented work is to combine the object detection pipeline with the region growing-based segmentation so as to further segment detected object clusters. The segmentation into geometric primitives, for example, allows for decomposing the detected objects into shapes and functional parts as well as to represent

objects as compounds of shape primitives (Holz *et al.*, 2014a). Also, objects touching each other or being stacked onto one another end up in a single cluster in the object detection pipeline. Segmenting locally smooth surfaces, convex shapes, or shape primitives allows splitting the clusters to better describe the actual composition of the cluster.

In addition to the aforementioned combinations, the presented methods can be extended individually. The object segmentation pipeline, for example, stopped after identifying regions in the input data that contain objects. As has been seen in two applications, the segmentation approach can be easily integrated into complete object perception pipelines, e.g., to recognize or classify the found objects. Especially for the recognition of objects a wide variety of local and global feature descriptors have been proposed, see the tutorial of Aldoma *et al.* (2012a) for an overview of publicly available implementations. Global feature descriptors describe complete objects rather than parts of it (as done by local descriptors) and, hence, need a prior segmentation of the scene. For a recognition task, the objects segmented using the presented approach can easily be described by global feature descriptors such as the viewpoint feature histogram (VFH) of Rusu *et al.* (2010) or the improved OUR-CVFH by Aldoma *et al.* (2012b). Lai *et al.* (2011), for example, present a tree-based semantic representation for jointly segmenting, recognizing and localizing objects.

All approaches presented in this thesis have been designed for real-time processing on a single CPU core without any parallelization. If, however, a GPU or multiple CPU cores are available most of the approaches can be parallelized in order to gain additional speed-ups. Orts-Escolano *et al.* (2015), for example, use the approximate surface reconstruction from Chapter 3 and implemented a variant where the individual triangles or quads are processed in parallel on a GPU. Both the segmentation and the mapping pipelines can be considerably accelerated when parallelizing all processing steps.

The registration and mapping pipeline as presented in Chapter 5 only distinguished between a local window for the initial alignment of new 3D point clouds and the complete graph to optimize all estimated poses. A possible extension to the pipeline is to address the registration and mapping problem in a completely hierarchical approach with several levels (Olson, 2009; Zhang *et al.*, 2012a) where new 3D point clouds are aligned in a local window that represents a region of the environment. On an upper level, regions are aligned to each other to represent the complete environment. The global optimization then processes a considerably reduced amount of data, but the changes done on the upper level need to be propagated to other levels. On lower levels, subsets of the point clouds such as individual scan lines can be are re-aligned using updated surface statistics after optimizing the pose of neighboring point clouds in the region. Since the point clouds in the MAV application are aggregated solely based on pose

estimates by visual odometry the exact pose of a scan line in the point cloud may be inaccurate and could be refined. On the lowest level, individual measurements can be corrected, e.g., to compensate for noise. A particularly challenging problem in such an approach is to decide when to do changes and which changes need to be made in which level.

Finally, a particularly prominent line of thinking (see, for example, the recent workshop on real-time simultaneous localization and mapping (SLAM) systems at ICCV 2015) is that of combining (deep) learning techniques with real-time SLAM systems to simultaneously build, segment, and label 3D environment representations. Salas-Moreno *et al.* (2014), for example, segment planar surface patches in RGB-D images on a GPU and propagate found segments in a real-time dense SLAM system. As a result, the environment can be efficiently represented solely using planar patches and a set of surfels representing the non-planar regions in the scene. Similarly, Stückler *et al.* (2015) combine object class segmentation using random forests with a real-time SLAM system. A central question for future developments is how to improve SLAM using semantic information and how to improve the extraction of semantic information using the results of the SLAM system, i.e., having the two approaches augment and improve each other rather then letting them run in parallel.

BIBLIOGRAPHY

Agarwal, P., W. Burgard, and C. Stachniss (2014). "Survey of Geodetic Mapping Methods: Geodetic Approaches to Mapping and the Relationship to Graph-Based SLAM." In: *IEEE Robotics and Automation Magazine* 21.3, pp. 63–80.

Aldoma, A., Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze (2012a). "Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation." In: *IEEE Robotics and Automation Magazine* 19.3, pp. 80–91.

Aldoma, A., F. Tombari, R. Rusu, and M. Vincze (2012b). "OUR-CVFH: Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation." In: *Pattern Recognition*. Ed. by A. Pinz, T. Pock, H. Bischof, and F. Leberl. Vol. 7476. Lecture Notes in Computer Science, pp. 113–122. ISBN: 978-3-642-32716-2.

Amenta, N., S. Choi, T. K. Dey, and N. Leekha (2000). "A Simple Algorithm for Homeomorphic Surface Reconstruction." In: *Proceedings of the Sixteenth Annual Symposium on Computational Geometry (SCG)*. Clear Water Bay, Hong Kong, pp. 213–222.

An, S.-Y., L.-K. Lee, and S.-Y. Oh (2012). "Fast incremental 3D plane extraction from a collection of 2D line segments for 3D mapping." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, pp. 4530–4537.

Anderson, D., H. Herman, and A. Kelly (2005). "Experimental Characterization of Commercial Flash Ladar Devices." In: *Proceedings of the International Conference of Sensing and Technology*. Palmerston North, New Zealand.

Attene, M., B. Falcidieno, and M. Spagnuolo (2006). "Hierarchical mesh segmentation based on fitting primitives." In: *The Visual Computer* 22.3, pp. 181–193.

Bab-Hadiashar, A. and N. Gheissari (2006). "Range image segmentation using surface selection criterion." In: *IEEE Transactions on Image Processing* 15.7, pp. 2006–2018.

Bachrach, A., R. He, and N. Roy (2009). "Autonomous flight in unstructured and unknown indoor environments." In: *Proceedings of the European Micro Aerial Vehicle Conference (EMAV)*, pp. 1–8.

Baker, A. (2007). "Occam's Razor in science: a case study from biogeography." In: *Biology & Philosophy* 22.2, pp. 193–215.

Bäuml, B., F. Schmidt, T. Wimböck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. Grebenstein, O. Eiberger, and G. Hirzinger (2011). "Catching flying balls and preparing coffee:

Humanoid Rollin'Justin performs dynamic and sensitive tasks." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 3443–3444.

Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool (2008). "Speeded-Up Robust Features (SURF)." In: *Computer Vision and Image Understanding* 110.3, pp. 346–359. ISSN: 1077-3142.

Beetz, M., U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth (2011). "Robotic Roommates Making Pancakes." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Bled, Slovenia, pp. 529–536.

Berger, M., J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva (2013). "A Benchmark for Surface Reconstruction." In: *ACM Transactions on Graphics (TOG)* 32.2, 20:1–20:17.

Berner, A., J. Li, D. Holz, J. Stückler, S. Behnke, and R. Klein (2013). "Combining Contour and Shape Primitives for Object Detection and Pose Estimation of Prefabricated Parts." In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Melbourne, Australia.

Besl, P. J. and N. D. McKay (1992). "A Method for Registration of 3-D Shapes." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2, pp. 239–256.

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

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. ISBN: 0387310738.

Bohren, J., R. Rusu, E. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer (2011). "Towards autonomous robotic butlers: Lessons learned with the PR2." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 5568–5575.

Borrmann, D., J. Elseberg, K. Lingemann, and A. Nüchter (2011). "The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design." In: *3D Research* 2 (2), pp. 1–13. ISSN: 2092-6731.

Borrmann, D., J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg (2008). "Globally consistent 3D mapping with scan matching." In: *Robotics and Autonomous Systems* 56.2, pp. 130–142.

Bosse, M. and R. Zlot (2013). "Place recognition using keypoint voting in large 3D lidar datasets." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Frankfurt, Germany, pp. 2677–2684.

Bosse, M., R. Zlot, and P. Flick (2012). "Zebedee: Design of a Spring-Mounted 3-D Range Sensor with Application to Mobile Mapping." In: *IEEE Transactions on Robotics* 28.5, pp. 1104–1119.

Butt, I. T. and N. M. Rajpoot (2009). "Multilateral filtering: a novel framework for generic similarity-based image denoising." In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Cairo, Egypt, pp. 2945–2948.

Calakli, F. and G. Taubin (2011). "SSD: Smooth Signed Distance Surface Reconstruction." In: *Graphics Forum* 30.7, pp. 1993–2002.

Calonder, M., V. Lepetit, C. Strecha, and P. Fua (2010). "BRIEF: Binary Robust Independent Elementary Features." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Hersonissos, Crete, Greece, pp. 778–792.

Carr, J. C., R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans (2001). "Reconstruction and Representation of 3D Objects with Radial Basis Functions." In: *Proceedings of the ACM SIGGRAPH*, pp. 67–76.

Censi, A. (2007). "An accurate closed-form estimate of ICP's covariance." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Rome, Italy, pp. 3167–3172.

Checchin, P., L. Trassoudaine, and J. Alizon (1997). "Segmentation of Range Images into Planar Regions." In: *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 156–163.

Choi, S., T. Kim, and W. Yu (2009). "Performance Evaluation of RANSAC Family." In: *Proceedings of the British Machine Vision Conference (BMVC)*. London, England, pp. 1–12.

Collet, A., S. Srinivasa, and M. Hebert (2011). "Structure discovery in multi-modal data: A region-based approach." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 5695–5702.

Cover, H., S. Choudhury, S. Scherer, and S. Singh (2013). "Sparse Tangential Network (SPARTAN): Motion Planning for Micro Aerial Vehicles." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2820–2825.

Cummins, M. and P. Newman (2010). "FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model." In: *Proceedings of the International Conference on Machine Learning (ICML)*.

Cupec, R., E. K. Nyarko, and D. Filko (2011). "Fast 2.5D Mesh Segmentation to Approximately Convex Surfaces." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Örebro, Sweden, pp. 49–54.

Décoret, X., F. Durand, F. Sillion, and J. Dorsey (2003). "Billboard Clouds for Extreme Model Simplification." In: *Proceedings of the ACM SIGGRAPH*. San Diego, CA, USA.

Dey, T. K., J. Giesen, and J. Hudson (2001). "Delaunay Based Shape Reconstruction from Large Data." In: *Proceedings of the IEEE 2001 Symposium on Parallel and Large-data Visualization and Graphics (PVG)*. San Diego, CA, USA, pp. 19–27.

Dey, T. K. and S. Goswami (2003). "Tight Cocone: A Water-tight Surface Reconstructor." In: *Journal of Computing and Information Science in Engineering* 3.4, pp. 302–307.

Droeschel, D., D. Holz, and S. Behnke (2010a). "Probabilistic Phase Unwrapping for Time-of-Flight Cameras." In: *Proceedings of the International Symposium on Robotics (ISR) and the German Conference on Robotics (ROBOTIK)*. Munich, Germany, pp. 318–324.

Droeschel, D., D. Holz, and S. Behnke (2010b). "Multi-frequency Phase Unwrapping for Time-of-Flight cameras." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, pp. 1463–1469.

Droeschel, D., D. Holz, and S. Behnke (2014a). "Omnidirectional Perception for Lightweight MAVs using a Continously Rotationg 3D Laser Scanner." In: *Photogrammetrie Fernerkundung Geoinformation (PFG)* 5, pp. 451–464.

Droeschel, D., J. Stückler, and S. Behnke (2014b). "Local Multi-Resolution Representation for 6D Motion Estimation and Mapping with a Continuously Rotating 3D Laser Scanner." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Droeschel, D., J. Stückler, and S. Behnke (2014c). "Local Multi-Resolution Surfel Grids for MAV Motion Estimation and 3D Mapping." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*.

Einstein, A. (1934). "On the Method of Theoretical Physics." In: *Philosophy of Science* 1.2, pp. 163–169.

Elmasry, M. (2013). "Stair Detection and Modeling using RGB-D Cameras." Diploma thesis. Bonn, Germany: University of Bonn.

Endres, F., J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard (2012a). "An evaluation of the RGB-D SLAM system." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1691–1696.

Endres, F., J. Hess, J. Sturm, D. Cremers, and W. Burgard (2014). "3D Mapping with an RGB-D Camera." In: *IEEE Transactions on Robotics* 30.1, pp. 177–187.

Endres, F., J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard (2012b). "An Evaluation of the RGB-D SLAM System." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Engel, J., T. Schöps, and D. Cremers (2014). "LSD-SLAM: Large-Scale Direct Monocular SLAM." In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 834–849.

Engel, J., J. Sturm, and D. Cremers (2013). "Semi-Dense Visual Odometry for a Monocular Camera." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1449–1456.

Enjarini, B. and A. Graser (2012). "Planar segmentation from depth images using gradient of depth feature." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, pp. 4668–4674.

Feng, C., Y. Taguchi, and V. R. Kamat (2014). "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, pp. 6218–6225.

Fernández-Moral, E., Rives, V. Arévalo, and J. González-Jiménez (2016). "Scene structure registration for localization and mapping." In: *Robotics and Autonomous Systems* 75, Part B, pp. 649–660.

Finman, R., L. Paull, and J. J. Leonard (2015). "Toward Object-based Place Recognition in Dense RGB-D Maps." In: *Proceedings of the ICRA workshop on visual place recognition in changing environments*.

Fischler, M. A. and R. C. Bolles (1981). "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." In: *Communications of the ACM* 24.6, pp. 381–395.

Fleishman, S., I. Drori, and D. Cohen-Or (2003). "Bilateral Mesh Denoising." In: *Proceedings of the ACM SIGGRAPH*. San Diego, California, pp. 950–953. ISBN: 1-58113-709-5.

Forster, C., M. Pizzoli, and D. Scaramuzza (2014). "SVO: Fast Semi-Direct Monocular Visual Odometry." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15–22.

Fossel, J., D. Hennes, D. Claes, S. Alers, and K. Tuyls (2013). "OctoSLAM: A 3D mapping approach to situational awareness of unmanned aerial vehicles." In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 179–188.

Frese, U., P. Larsson, and T. Duckett (2005). "A multilevel relaxation algorithm for simultaneous localization and mapping." In: *IEEE Transactions on Robotics* 21.2, pp. 196–207.

Frey, B. J., R. Koetter, and N. Petrovic (2001). "Very Loopy Belief Propagation for Unwrapping Phase Images." In: *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, BC, Canada.

Frigui, H. and R. Krishnapuram (1999). "A robust competitive clustering algorithm with applications in computer vision." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.5, pp. 450–465.

Frintrop, S., G. M. García, and A. B. Cremers (2014). "A Cognitive Approach for Object Discovery." In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. Stockholm, Sweden, pp. 2329–2334.

Fuchs, S. and S. May (2007). "Calibration and Registration for Precise Surface Reconstruction." In: *Proceedings of the DAGM Dynamic 3D Imaging Workshop (Dyn3D)*.

Fuchs, S. and G. Hirzinger (2008). "Extrinsic and depth calibration of ToF-cameras." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Anchorage, AK, USA.

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

Gopi, M. and S. Krishnan (2002). "A Fast and Efficient Projection-Based Approach for Surface Reconstruction." In: *Proceedings of the ACM SIGGRAPH*, pp. 179–186.

Gotardo, P. F. U., O. R. P. Bellon, K. L. Boyer, and L. Silva (2004). "Range image segmentation into planar and quadric surfaces using an improved robust estimator and genetic algorithm." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34.6, pp. 2303–2316.

Gotardo, P. F. U., O. R. P. Bellon, and L. Silva (2003). "Range image segmentation by surface extraction using an improved robust estimator." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Madison, WI, USA, pp. 33–38.

Gouraud, H. (1971). "Continuous Shading of Curved Surfaces." In: *IEEE Transactions on Computers* 20.6, pp. 623–629.

Graham, R. L. (1972). "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set." In: *Information Processing Letters* 1.4, pp. 132–133.

Grisetti, G., C. Stachniss, and W. Burgard (2009). "Nonlinear Constraint Network Optimization for Efficient Map Learning." In: *IEEE Transactions on Intelligent Transportation Systems* 10.3, pp. 428–439.

Grisetti, G., R. Kuemmerle, C. Stachniss, U. Frese, and C. Hertzberg (2010). "Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 273–278.

Grzonka, S., G. Grisetti, and W. Burgard (2009). "Towards a Navigation System for Autonomous Indoor Flying." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2878–2883.

Grzonka, S., G. Grisetti, and W. Burgard (2012). "A Fully Autonomous Indoor Quadrotor." In: *IEEE Transactions on Robotics* 28.1, pp. 90–100.

Gutmann, J.-S., M. Fukuchi, and M. Fujita (2008). "3D Perception and Environment Map Generation for Humanoid Robot Navigation." In: *The International Journal of Robotics Research* 27.10, pp. 1117–1134.

Hähnel, D., W. Burgard, and S. Thrun (2003). "Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot." In: *Robotics and Autonomous Systems* 44.1, pp. 15–27.

Handa, A., T. Whelan, J. McDonald, and A. Davison (2014). "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1524–1531.

Hans, M., B. Graf, and R. Schraft (2002). "Robotic home assistant Care-O-bot: past-present-future." In: *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*. Berlin, Germany, pp. 380–385.

Harati, A., S. Gächter, and R. Siegwart (2006). "Fast Range Image Segmentation for Indoor 3D-SLAM." In: *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*. Toulouse, France.

Henry, P., M. Krainin, E. Herbst, X. Ren, and D. Fox (2012). "RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments." In: *The International Journal of Robotics Research* 31.5, pp. 647–663.

Herbst, E., P. Henry, X. Ren, and D. Fox (2011a). "Toward object discovery and modeling via 3-D scene comparison." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 2623–2629.

Herbst, E., X. Ren, and D. Fox (2011b). "RGB-D object discovery via multi-scene analysis." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, pp. 4850–4856.

Holsinger, K. E. (1981). "Comment: The blunting of Occam's Razor, or to hell with parsimony." In: *Canadian Journal of Zoology* 59.1, pp. 144–146.

Holz, D. and S. Behnke (2010). "Sancta Simplicitas – On the efficiency and achievable results of SLAM using ICP-Based Incremental Registration." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, Alaska, USA, pp. 1380–1387.

Holz, D. and S. Behnke (2012). "Fast Range Image Segmentation and Smoothing using Approximate Surface Reconstruction and Region Growing." In: *Proceedings of the 12th International Conference on Intelligent Autonomous Systems (IAS)*. Jeju Island, Korea.

Holz, D. and S. Behnke (2014a). "Approximate triangulation and region growing for efficient segmentation and smoothing of range images." In: *Robotics and Autonomous Systems* 62.9, pp. 1282–1293. ISSN: 0921-8890.

Holz, D. and S. Behnke (2014b). "Registration of Non-Uniform Density 3D Point Clouds using Approximate Surface Reconstruction." In: *Proceedings of the 45th International Symposium on Robotics (ISR) and 8th German Conference on Robotics (ROBOTIK)*. Munich, Germany.

Holz, D. and S. Behnke (2014c). "Mapping with Micro Aerial Vehicles by Registration of Sparse 3D Laser Scans." In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS)*. Padova, Italy.

Holz, D. and S. Behnke (2015a). "Registration of Non-Uniform Density 3D Laser Scans for Mapping with Micro Aerial Vehicles." In: *Robotics and Autonomous Systems* 74, Part B, pp. 318–330.

Holz, D. and S. Behnke (2015b). "Approximate Surface Reconstruction and Registration for RGB-D SLAM." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Lincoln, UK.

Holz, D., S. Holzer, R. B. Rusu, and S. Behnke (2011). "Real-Time Plane Segmentation using RGB-D Cameras." In: *Proceedings of the 15th RoboCup International Symposium*. Vol. 7416. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, pp. 307–317.

Holz, D., M. Nieuwenhuisen, D. Droeschel, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke (2014a). "Active Recognition and Manipulation for Mobile Robot Bin Picking." In: *Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe:* ed. by F. Röhrbein, G. Veiga, and C. Natale. Vol. 94. Springer Tracts in Advanced Robotics. Springer International Publishing, pp. 133–153.

Holz, D., J. Ruiz-del-Solar, K. Sugiura, and S. Wachsmuth (2014b). "On RoboCup@Home – Past, Present and Future of a Scientific Competition for Service Robots." In: *RoboCup 2014: Robot World Cup XVIII [papers from the 18th Annual RoboCup International Symposium, João Pessoa, Brazil, July 15*, pp. 686–697.

Holz, D., R. Schnabel, D. Droeschel, J. Stückler, and S. Behnke (2010). "Towards Semantic Scene Analysis with Time-of-Flight Cameras." In: *Proceedings of the RoboCup International Symposium*. Vol. 6556. Lecture Notes in Computer Science. Singapore, Singapore: Springer, pp. 121–132.

Holz, D., A. Topalidou-Kyniazopoulou, M. R. P. Francesco Rovida, V. Krüger, and S. Behnke (2015a). "A Skill-Based System for Object Perception and Manipulation for Automating Kitting Tasks." In: *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Luxemburg.

Holz, D., A. Topalidou-Kyniazopoulou, J. Stückler, and S. Behnke (2015b). "Real-Time Object Detection, Localization and Verification for Fast Robotic Depalletizing." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, pp. 1459–1466.

Holzer, S., R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab (2012). "Real-Time Surface Normal Estimation from Organized Point Cloud Data Using Integral Images." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, pp. 2684–2689.

Hoover, A., G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher (1996). "An Experimental Comparison of Range Image Segmentation Algorithms." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (7), pp. 673–689. ISSN: 0162-8828.

Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle (1992). "Surface Reconstruction from Unorganized Points." In: *ACM SIGGRAPH Computer Graphics* 26.2, pp. 71–78.

Hornung, A., K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard (2013). "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees." In: *Autonomous Robots* 34.3, pp. 189–206.

Hsiao, K., S. Chitta, M. Ciocarlie, and E. G. Jones (2010). "Contact-Reactive Grasping of Objects with Partial Shape Information." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, pp. 1228–1235.

Huang, A. S., A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy (2011). "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera." In: *Proceedings of the International Symposium on Robotics Research (ISRR)*.

Huh, S., D. Shim, and J. Kim (2013). "Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of UAVs." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3158–3163.

Husain, F., B. Dellen, and C. Torras (2015). "Consistent Depth Video Segmentation Using Adaptive Surface Models." In: *IEEE Transactions on Cybernetics* 45.2, pp. 266–278.

Iocchi, L., D. Holz, J. Ruiz-del-Solar, K. Sugiura, and T. van der Zant (2015). "RoboCup@Home: Analysis and results of evolving competitions for domestic and service robots." In: *Artificial Intelligence* 229, pp. 258–281.

Jain, A. and C. C. Kemp (2010). "EL-E: An Assistive Mobile Manipulator that Autonomously Fetches Objects from Flat Surfaces." In: *Autonomous Robots* 28 (1), pp. 45–64.

Jiang, X. (2000). "An adaptive contour closure algorithm and its experimental evaluation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11, pp. 1252–1265.

Jiang, X., K. W. Bowyer, Y. Morioka, S. Hiura, K. Sato, S. Inokuchi, M. Bock, C. Guerra, R. E. Loke, and J. M. H. du Buf (2000). "Some Further Results of Experimental Comparison of Range Image Segmentation Algorithms." In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. Barcelona, Spain, pp. 4877–4882.

Jiang, X. and H. Bunke (1994). "Fast segmentation of range images into planar regions by scan line grouping." In: *Machine Vision and Applications* 7 (2), pp. 115–122.

Jiang, X. and H. Bunke (1996). "Robust Edge Detection in Range Images Based on Scan Line Approximation." In: *Computer Vision and Image Understanding* 73, pp. 183–199.

Jin, S., R. R. Lewis, and D. West (2005). "A comparison of algorithms for vertex normal computation." In: *The Visual Computer* 21.1, pp. 71–82.

Karpathy, A., S. Miller, and L. Fei-Fei (2013). "Object discovery in 3D scenes via shape analysis." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany, pp. 2088–2095.

Kazhdan, M., M. Bolitho, and H. Hoppe (2006). "Poisson Surface Reconstruction." In: *Proceedings of the Eurographics Symposium on Geometry Processing (SGP)*. Cagliari, Sardinia, Italy, pp. 61–70.

Kazhdan, M. and H. Hoppe (2013). "Screened Poisson Surface Reconstruction." In: *ACM Transactions on Graphics (TOG)* 32.3, 29:1–29:13.

Kerl, C., J. Sturm, and D. Cremers (2013). "Dense Visual SLAM for RGB-D Cameras." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2100–2106.

Kitt, B., A. Geiger, and H. Lategahn (2010). "Visual Odometry based on Stereo Image Sequences with RANSAC-based Outlier Rejection Scheme." In: *Proceedings of the Intelligent Vehicles Symposium (IV)*.

Kohlbrecher, S., J. Meyer, O. von Stryk, and U. Klingauf (2011). "A Flexible and Scalable SLAM System with Full 3D Motion Estimation." In: *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*.

Konolige, K. and M. Agrawal (2008). "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping." In: *IEEE Transactions on Robotics* 24.5, pp. 1066–1077.

Koster, K. and M. Spann (2000). "MIR: an approach to robust clustering-application to range image segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.5, pp. 430–444.

Kuhn, H. W. (1955). "The Hungarian Method for the Assignment Problem." In: *Naval Research Logistics Quarterly* 2.1, pp. 83–97.

Kümmerle, R., G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard (2011). "g2o: A general framework for graph optimization." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3607–3613.

Lai, K., L. Bo, X. Ren, and D. Fox (2011). "A Scalable Tree-based Approach for Joint Object and Pose Recognition." In: *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI)*. San Francisco, CA, USA.

Lee, K.-M., P. Meer, and R.-H. Park (1998). "Robust adaptive segmentation of range images." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (2), pp. 200–205.

Leeper, A., K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow (2012). "Strategies for Human-in-the-Loop Robotic Grasping." In: *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Boston, MA, USA.

Leroy, J., N. Riche, M. Mancas, and B. Gosselin (2015). "3D Saliency Based on Supervoxels Rarity in Point Clouds." In: Hamburg, Germany.

Li, R., L. Liu, L. Phan, S. Abeysinghe, C. Grimm, and T. Ju (2010). "Polygonizing Extremal Surfaces with Manifold Guarantees." In: *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling (SPM)*. Haifa, Israel, pp. 189–194.

Lim, H., J. Lim, and H. Kim (2014). "Online 3D Reconstruction and 6-DoF Pose Estimation for RGB-D Sensors." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Zurich, Switzerland, pp. 238–254.

Lim, J., J.-M. Frahm, and M. Pollefeys (2011). "Online environment mapping." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Colorado Springs, CO, USA, pp. 3489–3496.

Lorensen, W. E. and H. E. Cline (1987). "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." In: *ACM SIGGRAPH Computer Graphics* 21.4, pp. 163–169.

Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints." In: *The International Journal of Computer Vision* 60.2, pp. 91–110. ISSN: 0920-5691.

Lowry, S., N. Sunderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford (2015). "Visual Place Recognition: A Survey." In: *IEEE Transactions on Robotics* PP.99. Early access articles, pp. 1–19.

Lu, F. and E. Milios (1997). "Globally Consistent Range Scan Alignment for Environment Mapping." In: *Autonomous Robots* 4.4, pp. 333–349.

Lynen, S., M. Bosse, P. Furgale, and R. Siegwart (2014). "Placeless Place-Recognition." In: *Proceedings of the International Conference on 3D Vision (3DV)*. Tokyo, Japan, pp. 303–310.

Ma, L. and G. Sibley (2014). "Unsupervised Dense Object Discovery, Detection, Tracking and Reconstruction." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Zurich, Switzerland, pp. 80–95.

Magid, E., O. Soldea, and E. Rivlin (2007). "A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data." In: *Computer Vision and Image Understanding* 107.3, pp. 139–159. ISSN: 1077-3142.

Magnusson, M., T. Duckett, and A. J. Lilienthal (2007). "Scan Registration for Autonomous Mining Vehicles Using 3D-NDT." In: *Journal of Field Robotics* 24.10, pp. 803–827.

Magnusson, M., A. Nüchter, C. Lörken, A. J. Lilienthal, and J. Hertzberg (2009a). "Evaluation of 3D Registration Reliability and

Speed – A Comparison of ICP and NDT." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3907–3912.

Magnusson, M., H. Andreasson, A. Nuchter, and A. Lilienthal (2009b). "Appearance-based loop detection from 3D laser data using the normal distributions transform." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, pp. 23–28.

Maier, R., J. Sturm, and D. Cremers (2014). "Submap-based Bundle Adjustment for 3D Reconstruction from RGB-D Data." In: *Proceedings of the German Conference on Pattern Recognition (GCPR)*.

Marton, Z. C., R. B. Rusu, and M. Beetz (2009). "On Fast Surface Reconstruction Methods for Large and Noisy Datasets." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, pp. 3218–3223.

Marton, Z.-C., F. Balint-Benczedi, O. Mozos, N. Blodow, A. Kanezaki, L. Goron, D. Pangercic, and M. Beetz (2014). "Part-Based Geometric Categorization and Object Reconstruction in Cluttered Table-Top Scenes." In: *Journal of Intelligent & Robotic Systems* 76.1, pp. 35–56.

Mason, J., B. Marthi, and R. Parr (2012). "Object disappearance for object discovery." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, pp. 2836–2843.

Matas, J. and O. Chum (2002). "Randomized RANSAC with T(d, d) test." In: *Proc. of the British Machine Vision Conference 2002 (BMVC)*.

May, S., B. Werner, H. Surmann, and K. Pervolz (2006). "3D time-of-flight cameras for mobile robotics." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Beijing, China, pp. 790–795.

May, S., D. Droeschel, D. Holz, S. Fuchs, E. Malis, A. Nüchter, and J. Hertzberg (2009). "Three-dimensional mapping with time-of-flight cameras." In: *Journal of Field Robotics* 26.11-12, pp. 934–965.

Meeussen, W., M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. P. Gerkey, and E. Berger (2010). "Autonomous Door Opening and Plugging In with a Personal Robot." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, AK, USA, pp. 729–736.

Memarsadeghi, N., D. M. Mount, N. S. Netanyahu, and J. L. Moigne (2007). "A Fast Implementation of the ISODATA Clustering Algorithm." In: *International Journal of Computational Geometry and Applications* 17, pp. 71–103.

Middelberg, S., T. Sattler, O. Untzelmann, and L. Kobbelt (2014). "Scalable 6-DOF Localization on Mobile Devices." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Zurich, Switzerland, pp. 268–283.

Montemerlo, M., J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.* (2008). "Junior: The Stanford Entry in the Urban Challenge." In: *Journal of Field Robotics* 25.9, pp. 569–597.

Moosmann, F., O. Pink, and C. Stiller (2009). "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion." In: *Proceedings of the IEEE Intelligent Vehicles Symposium*. Xi'an, China, pp. 215–220.

Morris, W., I. Dryanovski, J. Xiao, and S. Member (2010). "3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids." In: *In RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras*.

Mount, D. M. and S. Arya (1997). "ANN: A library for approximate nearest neighbor searching." In: *Proc. of the 2nd Annual Fall Workshop on Computational Geometry*.

Muja, M., R. B. Rusu, G. R. Bradski, and D. G. Lowe (2011). "REIN - A fast, robust, scalable REcognition INfrastructure." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 2939–2946.

Muja, M. and D. G. Lowe (2009). "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." In: *Proceedings of the International Conference on Computer Vision Theory and Application (VISSAPP)*. Lisbon, Portugal, pp. 331–340.

Muller, S. A. (2013). "Planar Segmentation of Range Images." MA thesis. South Africa: Stellenbosch University.

Musser, D. R. (1997). "Introspective Sorting and Selection Algorithms." In: *Software: Practice and Experience* 27.8, pp. 983–993. ISSN: 1097-024X.

Newcombe, R. A., D. Fox, and S. M. Seitz (2015). "DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA.

Newcombe, R. A., S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon (2011). "KinectFusion: Real-time dense surface mapping and tracking." In: *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 127–136.

Nguyen, C. V., S. Izadi, and D. Lovell (2012). "Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking." In: *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*. Zürich, Switzerland, pp. 524–530.

Nüchter, A., K. Lingemann, J. Hertzberg, and H. Surmann (2005). "6D SLAM with approximate data association." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 242–249.

Nüchter, A., H. Surmann, and J. Hertzberg (2003). "Automatic Model Refinement for 3D Reconstruction with Mobile Robots." In: *Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM)*. Banff, Canada, pp. 394–401.

Oehler, B., J. Stückler, J. Welle, D. Schulz, and S. Behnke (2011). "Efficient Multi-Resolution Plane Segmentation of 3D Point Clouds[3]." In: *Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA)*. Aachen, Germany, pp. 145–156.

Ohno, K., T. Nomura, and S. Tadokoro (2006). "Real-Time Robot Trajectory Estimation and 3D Map Construction using 3D Camera." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Bejing, China, pp. 5279–5285.

Olson, E. B. (2009). "Real-time correlative scan matching." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4387–4393.

Olson, E. B., J. Leonard, and S. Teller (2006). "Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2262–2269.

Orts-Escolano, S., V. Morell, J. Garcia-Rodriguez, M. Cazorla, and R. B. Fisher (2015). "Real-time 3D semi-local surface patch extraction using GPGPU." In: *Journal of Real-Time Image Processing* 10.4, pp. 647–666.

Osswald, S., A. Gorog, A. Hornung, and M. Bennewitz (2011). "Autonomous climbing of spiral staircases with humanoids." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, pp. 4844–4849.

Palmer, S. E. (1985). "The role of symmetry in shape perception." In: *Acta Psychologica* 59.1, pp. 67–90.

Papon, J., A. Abramov, M. Schoeler, and F. Worgotter (2013). "Voxel Cloud Connectivity Segmentation – Supervoxels for Point Clouds." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Portland, OR, USA, pp. 2027–2034.

Pathak, K., A. Birk, and J. Poppinga (2008). "Sub-pixel depth accuracy with a time of flight sensor using multimodal Gaussian analysis." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nice, France, pp. 3519–3524.

Pathak, K., N. Vaskevicius, and A. Birk (2009). "Revisiting uncertainty analysis for optimum planes extracted from 3D range sensor point-clouds." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, pp. 1631–1636.

3 Note: not all results referenced here have been published in the paper but all results can be found in Oehler's diploma thesis at `http://www.ais.uni-bonn.de/theses/Bastian_Oehler_Diplomarbeit_08_2011.pdf`

Pathak, K., A. Birk, N. Vaskevicius, and J. Poppinga (2010). "Fast Registration Based on Noisy Planes with Unknown Correspondences for 3D Mapping." In: *IEEE Transactions on Robotics* 3, pp. 424–441.

Poppinga, J., N. Vaskevicius, A. Birk, and K. Pathak (2008). "Fast Plane Detection and Polygonalization in noisy 3D Range Images." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nice, France, pp. 3378–3383.

Rabbani, T., F. A. van den Heuvel, and G. Vosselman (2006). "Segmentation of point clouds using smoothness constraint." In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 36, pp. 248–253.

Razlaw, J., D. Droeschel, D. Holz, and S. Behnke (2015). "Evaluation of Registration Methods for Sparse 3D Laser Scans." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Lincoln, UK.

Richtsfeld, A., T. Morwald, J. Prankl, M. Zillich, and M. Vincze (2012). "Segmentation of unknown objects in indoor environments." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, pp. 4791–4796.

Richtsfeld, A., T. Mörwald, J. Prankl, M. Zillich, and M. Vincze (2014). "Learning of Perceptual Grouping for Object Segmentation on RGB-D Data." In: *Journal of Visual Communication and Image Representation* 25.1, pp. 64–73.

Rosten, E. and T. Drummond (2006). "Machine Learning for High-Speed Corner Detection." In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 430–443.

Rosten, E., R. Porter, and T. Drummond (2010). "Faster and Better: A Machine Learning Approach to Corner Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1, pp. 105–119.

Roth, G. and M. D. Levine (1993). "Extracting geometric primitives." In: *CVGIP: Image Understanding* 58.1, pp. 1–22.

Rublee, E., V. Rabaud, K. Konolige, and G. Bradski (2011). "ORB: An Efficient Alternative to SIFT or SURF." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Barcelona, Spain, pp. 2564–2571.

Ruhnke, M., R. Kümmerle, G. Grisetti, and W. Burgard (2012). "Highly Accurate 3D Surface Models by Sparse Surface Adjustment." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 751–757.

Rusu, R. B. (2009). "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments." PhD thesis. Technische Universität München.

Rusu, R. B., N. Blodow, and M. Beetz (2009a). "Fast Point Feature Histograms (FPFH) for 3D Registration." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3212–3217.

Rusu, R. B., N. Blodow, Z. C. Marton, and M. Beetz (2009b). "Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, pp. 1–6.

Rusu, R. B., I. A. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki (2009c). "Real-time Perception-Guided Motion Planning for a Personal Robot." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, pp. 4245–4252.

Rusu, R., G. Bradski, R. Thibaux, and J. Hsu (2010). "Fast 3D recognition and pose using the Viewpoint Feature Histogram." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, pp. 2155–2162.

Salas-Moreno, R., B. Glocken, P. Kelly, and A. Davison (2014). "Dense planar SLAM." In: *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Munich, Germany, pp. 157–164.

Scaramuzza, D., M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, *et al.* (2014). "Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments." In: *IEEE Robotics and Automation Magazine*.

Schadler, M., J. Stückler, and S. Behnke (2014). "Rough Terrain 3D Mapping and Navigation Using a Continuously Rotating 2D Laser Scanner." In: *KI - Künstliche Intelligenz* 28.2, pp. 93–99.

Scherer, S. A. and A. Zell (2013). "Efficient onbard RGBD-SLAM for autonomous MAVs." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1062–1068.

Scherer, S., J. Rehder, S. Achar, H. Cover, A. D. Chambers, S. T. Nuske, and S. Singh (2012). "River mapping from a flying robot: state estimation, river detection, and obstacle mapping." In: *Autonomous Robots* 32.5. Ed. by G. Sukhatme, pp. 1–26.

Schmitt, F. and X. Chen (1991). "Fast segmentation of range images into planar regions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 710–711.

Schmittwilken, J., J. Saatkamp, W. Förstner, T. Kolbe, and L. Plümer (2007). "A Semantic Model for Stairs in Building Collars." In: *Photogrammetrie, Fernerkundung, Geoinformation PFG* 6, pp. 415–428.

Schnabel, R., R. Wahl, and R. Klein (2007). "Efficient RANSAC for point-cloud shape detection." In: *Computer Graphics Forum* 26.2, pp. 214–226.

Schneider, J., T. Läbe, and W. Förstner (2013). "Incremental Real-time Bundle Adjustment for Multi-camera Systems with Points at Infinity." In: *ISPRS Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XL-1/W2.

Segal, A., D. Hähnel, and S. Thrun (2009). "Generalized-ICP." In: *Proceedings of Robotics: Science and Systems*.

Shakarji, C. M. (1998). "Least-Squares Fitting Algorithms of the NIST Algorithm Testing System." In: 103.6, pp. 633–641.

Sheh, R., M. W. Kadous, and C. Sammut (2006). *On building 3D maps using a Range camera: Applications to Rescue Robotics*. Tech. rep. UNSW-CSE-TR-0609. UNSW, Sydney, Australia.

Shen, S., N. Michael, and V. Kumar (2011). "Autonomous multi-floor indoor navigation with a computationally constrained micro aerial vehicle." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2968–2969.

Shin, J., R. Triebel, and R. Siegwart (2010). "Unsupervised discovery of repetitive objects." In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. Anchorage, AK, USA, pp. 5041–5046.

Silberman, N., D. Hoiem, P. Kohli, and R. Fergus (2012). "Indoor Segmentation and Support Inference from RGBD Images." In: *ECCV*.

Silva, L., O. Bellon, and P. Gotardo (2002). "A global-to-local approach for robust range image segmentation." In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Rochester, NY, USA, pp. 773–776.

Smisek, J., M. Jancosek, and T. Pajdla (2011). "3D with Kinect." In: *Workshop Proceedings of the IEEE International Conference on Computer Vision (ICCV Workshops)*. Barcelona, Spain, pp. 1154–1160.

Srinivasa, S., D. Ferguson, C. Helfrich, D. Berenson, A. C. Romea, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and J. M. Vandeweghe (2010). "HERB: a home exploring robotic butler." In: *Autonomous Robots* 28.1, pp. 5–20.

Srinivasa, S., D. Ferguson, J. M. Vandeweghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat (2008). "The Robotic Busboy: Steps Towards Developing a Mobile Robotic Home Assistant." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*. Baden-Baden, Germany.

Steder, B., M. Ruhnke, S. Grzonka, and W. Burgard (2011). "Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, pp. 1249–1255.

Stein, S., F. Worgotter, M. Schoeler, J. Papon, and T. Kulvicius (2014). "Convexity based object partitioning for robot applications." In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. Hong Kong, China, pp. 3213–3220.

Steinbruecker, F., C. Kerl, J. Sturm, and D. Cremers (2013). "Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Sydney, Australia, pp. 3264–3271.

Steinbruecker, F., J. Sturm, and D. Cremers (2014). "Volumetric 3D Mapping in Real-Time on a CPU." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Hongkong, China, pp. 2021–2028.

Strasdat, H., A. Davison, J. Montiel, and K. Konolige (2011). "Double window optimisation for constant time visual SLAM." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2352–2359.

Stückler, J. and S. Behnke (2009). "Integrating indoor mobility, object manipulation, and intuitive interaction for domestic service tasks." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Paris, France, pp. 506–513.

Stückler, J. and S. Behnke (2011). "Dynamaid, an Anthropomorphic Robot for Research on Domestic Service Applications." In: *Journal for Control, Measurement, Electronics, Computing and Communications (AUTOMATIKA), Special Issue on ECMR'09* 52.3, pp. 233–243.

Stückler, J. and S. Behnke (2014). "Multi-resolution Surfel Maps for Efficient Dense 3D Modeling and Tracking." In: *J. Vis. Comun. Image Represent.* 25.1, pp. 137–147.

Stückler, J. and S. Behnke (2015). "Efficient Dense Rigid-Body Motion Segmentation and Estimation in RGB-D Video." In: *International Journal of Computer Vision* 113.3, pp. 233–245.

Stückler, J., D. Droeschel, K. Gräve, D. Holz, M. Schreiber, A. Topalidou-Kyniazopoulou, M. Schwarz, and S. Behnke (2013a). "Increasing Flexibility of Mobile Manipulation and Intuitive Human-Robot Interaction in RoboCup@Home." In: *RoboCup 2013: Robot World Cup XVII [papers from the 17th Annual RoboCup International Symposium, Eindhoven, The Netherlands, July 1, 2013]*. Pp. 135–146.

Stückler, J., R. Steffens, D. Holz, and S. Behnke (2011). "Real-Time 3D Perception and Efficient Grasp Planning for Everyday Manipulation Tasks." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Örebro, Sweden, pp. 177–182.

Stückler, J., R. Steffens, D. Holz, and S. Behnke (2013b). "Efficient 3D object perception and grasp planning for mobile manipulation in domestic environments." In: *Robotics and Autonomous Systems* 61.10, pp. 1106–1115.

Stückler, J., B. Waldvogel, H. Schulz, and S. Behnke (2015). "Dense real-time mapping of object-class semantics from RGB-D video." In: *Journal of Real-Time Image Processing* 10.4, pp. 599–609.

Sturm, J., N. Engelhard, F. Endres, W. Burgard, and D. Cremers (2012). "A Benchmark for the Evaluation of RGB-D SLAM Systems." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Sucan, I. A., M. Kalakrishnan, and S. Chitta (2010). "Combining planning techniques for manipulation using realtime perception." In:

Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Anchorage, AK, USA, pp. 2895–2901.

Tomić, T., K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka (2012). "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue." In: *IEEE Robotics and Automation Magazine* 19.3, pp. 46–56. ISSN: 1070-9932.

Torr, P. H. S. and A. Zisserman (2000). "MLESAC: A new robust estimator with application to estimating image geometry." In: *Computer Vision and Image Understanding* 78 (1), pp. 138–156.

Trevor, A. J. B., S. Gedikli, R. B. Rusu, and H. I. Christensen (2013). "Efficient Organized Point Cloud Segmentation with Connected Components." In: *Proceedings of the 3rd Workshop on Semantic Perception, Mapping and Exploration (SPME) at ICRA*. Karlsruhe, Germany.

Triebel, R., J. Shin, and R. Siegwart (2010). "Segmentation and Unsupervised Part-based Discovery of Repetitive Objects." In: *Robotics: Science and Systems VI*. Ed. by Y. Matsuoka, H. Durrant-Whyte, and J. Neira.

Turk, G. and M. Levoy (1994). "Zippered Polygon Meshes from Range Images." In: *Proceedings of the ACM SIGGRAPH*, pp. 311–318.

Tuytelaars, T., C. H. Lampert, M. B. Blaschko, and W. Buntine (2010). "Unsupervised Object Discovery: A Comparison." In: *The International Journal of Computer Vision* 88.2, pp. 284–302. ISSN: 0920-5691.

Urmson, C., J. Anhalt, H. Bae, J. A. (Bagnell, C. R. Baker, R. E. Bittner, T. Brown, M. N. Clark, M. Darms, D. Demitrish, J. M. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y.-W. Seo, S. Singh, J. M. Snider, J. C. Struble, A. (Stentz, M. Taylor, W. (L. Whittaker, Z. Wolkowicki, W. Zhang, and J. Ziglar (2008). "Autonomous driving in urban environments: Boss and the Urban Challenge." In: *Journal of Field Robotics* 25.8, pp. 425–466.

Vosselman, G., B. G. H. Gorte, G. Sithole, and T. Rabbani (2004). "Recognising structure in laser scanner point clouds." In: *Information Sciences* 46.8, pp. 1–6.

Weikersdorfer, D., D. Gossow, and M. Beetz (2012). "Depth-adaptive superpixels." In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. Tsukuba, Japan, pp. 2087–2090.

Weingarten, J. W., G. Gruener, and R. Siegwari (2004). "A State-of-the-Art 3D Sensor for Robot Navigation." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sendai, Japan, pp. 2155–2160.

Whelan, T., H. Johannsson, M. Kaess, J. Leonard, and J. McDonald (2013). "Robust real-time visual odometry for dense RGB-D map-

ping." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5724–5731.

Whelan, T., M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald (2012). "Kintinuous: Spatially Extended KinectFusion." In: *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. Sydney, Australia.

Whelan, T., S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison (2015). "ElasticFusion: Dense SLAM Without A Pose Graph." In: *Proceedings of Robotics: Science and Systems*. Rome, Italy.

Wiemann, T., M. Mrozinski, D. Feldschnieders, K. Lingemann, and J. Hertzberg (2016). "Data Handling in Large-Scale Surface Reconstruction." In: *Intelligent Autonomous Systems 13*. Ed. by E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi. Vol. 302. Advances in Intelligent Systems and Computing, pp. 499–511.

Wurm, K. M., D. Hennes, D. Holz, R. B. Rusu, C. Stachniss, K. Konolige, and W. Burgard (2011). "Hierarchies of octrees for efficient 3D mapping." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, pp. 4249–4255.

Zhang, H., Z. Hou, N. Li, and S. Song (2012a). "A Graph-Based Hierarchical SLAM Framework for Large-Scale Mapping." In: *Intelligent Robotics and Applications*. Ed. by C.-Y. Su, S. Rakheja, and H. Liu. Vol. 7507. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 439–448. ISBN: 978-3-642-33514-3.

Zhang, J. and S. Singh (2014). "LOAM: Lidar Odometry and Mapping in Real-time." In: *Proceedings of Robotics: Science and Systems*. Berkeley, USA.

Zhang, Y., C. Luo, and J. Liu (2012b). "Walk&Sketch: Create Floor Plans with an RGB-D Camera." In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ittsburgh, PA, USA, pp. 461–470.

Zhang, Y., C. Luo, and J. Liu (2012c). "Walk&Sketch: create floormaps with an RGB-D camera." In: *Proceedings of the ACM International Conference on Ubiquitous Computing*. Pittsburgh, PA, USA.

Zlot, R. and M. Bosse (2014). "Efficient Large-scale Three-dimensional Mobile Mapping for Underground Mines." In: *Journal of Field Robotics* 31.5, pp. 758–779.