Lehrstuhl für Steuerungs- und Regelungstechnik
Technische Universität München
Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss

Aspects of 3D Perception, Abstraction, and Interpretation in Autonomous Mobile Robotics

Klaas Klasing

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.sc. Samarjit Chakraborty

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss

2. Assoc. Prof. José Neira, Ph.D.
Universidad de Zaragoza, Spanien

Die Dissertation wurde am 15.04.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 16.10.2010 angenommen.

Foreword

The last three and a half years at the Institute of Automatic Control Engineering (LSR) at TU München have been an exciting and invigorating time. Filled with many insights and revelations, deadlines and milestones, failures and successes, this has been an incredible period for me to learn, experiment and share. As seems to be usual, the journey ends with a better understanding of the full extent of my ignorance. Notwithstanding, a selection of the insights gained during this time has culminated in this thesis, which represents a good part of the research I conducted. This work would not have been possible without numerous people that have helped and supported me over the course of the previous years.

First and foremost, I would like to thank my advisor Prof. Martin Buss, not only for triggering my interest in robotics during my undergraduate studies, but more importantly for providing an excellent research environment, stimulating discussions, and helpful advice during my time as a PhD student[1]. My sincere thanks also go to my co-advisor, Dr. Dirk Wollherr, for his practical comments, his ability to accommodate my often unorthodox requests, and his valuable support in the preparation of my research stay in Spain. The latter was made possible by Prof. Günther Schmidt, whom I thank for generously putting me in contact with the robotics research group in Zaragoza.

Daily life at the LSR would not have been the same without a set of great colleagues, and the important subset of great office mates. After pleasant encounters with Hasan Esen and numerous people during my temporary visit to the CoTeSys Central Robotics Lab, I finally came to settle in my old ‘habitat’ with Raphaela Groten and Georg Bätz. Thank you, Georg and Raphi, for filling these years with very helpful and motivating discussions, lots of laughter (although often about very bad jokes), and the solidary intake of food at random hours of the day. I am deeply indebted to Ulrich Unterhinninghofen for teaching me most of what I know about software and hardware development, and above all for being a very good friend. Within the Autonomous City Explorer project, my thanks go to Georgios Lidoris for making late-night debugging entertaining, and of course to Florian Rohrmüller, Quirin Mühlbauer, Andrea Bauer, Tinging Xu, Tianguang Zhang, and Stefan Sosnoswki, for the countless hours we have shared to make a robot go to the central square of Munich. My daily life was also pleesently enhanced by stimulating scientific and non-scientific discussions with Michael Bernhard, Michael Scheint, Moritz Große-Wentrup, Matthias Rungger, Omiros Kourakos, Martin Kuschel, and Thomas Schauß.

The three-months research stay in the Robotics and Realtime Group at the University of Zaragoza at the end of my PhD was an extremely productive and stimulating experience. Above all, I would like to thank Prof. José Neira for providing this opportunity and for a host of very fruitful discussions that helped shape the ideas of the object recognition framework. My sincere thanks for the warm welcome and quick integration into the group go to Ana Cristina Murillo, César Cadena, Héctor Becerra, and Luis Puig. I would also

[1] And, I almost forgot: Thank you, Martin, for the most amazing coffee machine I have ever seen ;-).

like to thank Óscar Martínez for several insightful discussions.

Special thanks go to all students who contributed to projects during my time at the LSR. Particularly, I would like to thank Markus Boxhammer, Andreas Dömel, Florian Petit, Thomas Nierhoff, and above all, Daniel Althoff, whose help with the normal vector benchmark was a central contribution to the third chapter of this thesis. Algorithms would be of little value without the hardware to verify them in experiment. Over the last years, I came to greatly appreciate the tireless support of Josef Gradl, Horst Kubick, Tobias Stoeber, Thomas Lowitz, and Wolfgang Jaschik for any kind of mechanical, electronic or computer problem encountered during experiments. As for the highly non-experimental matter of administrative issues, I would like to express my gratitude to Larissa Schmid, Waltraud Werner, and Wibke Borngesser for making my life so much easier at the LSR.

Strong support during these past years also came from my friends and family. A big 'thank you' goes out to Stephan Müßig and Sebastian Vogl for being great friends and keeping me on track while writing up the dissertation. An even bigger 'thank you' goes out to my sister, Insa Klasing, and to Mark Vandevelde, who have not only helped me with various issues of this thesis, but have also provided helpful guidance in life on many occasions. Finally, by far the biggest 'thank you' goes out to my parents for – well, for everything. Without you I would not be here. Thank you.

Munich, April 2010 Klaas Klasing

Acknowledgments

The research presented in this thesis was supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems (CoTeSys)*, the DFG project SFB-TR28 *Cognitive Automobiles*, and the EU STREP projects *Robot@CWE* and *Interactive Urban Robot*.

Abstract

This thesis presents novel insights and solutions to the problem of perceiving, abstracting, and interpreting the 3D structure of human environments with autonomous mobile robots. Within the broad research field of 3D perception, it focuses on the goal of retrieving a meaningful representation of objects from 3D laser range data for purposes of object recognition. As such, it represents an important contribution to the development of cognitive technical systems, which depend on intelligent perception as a core capability.

A chain of abstraction is developed throughout the chapters, that leads from raw point cloud data to recognized objects. Following a thorough review of state-of-the-art sensing approaches, new studies and methods for feature computation, surface-based point cloud segmentation and Bayesian part-based object recognition are presented. The proposed framework is generic and can be applied to a variety of scenarios involving structured 3D geometry. The application scenario considered throughout the thesis is the recognition of furniture objects in domestic environments, which constitutes a much-needed perception skill in household robotics. All methods and algorithms have been designed with a special focus on efficiency and applicability to real-life datasets. Specifically, the sampled 3D data may be comparatively sparse, exhibit inhomogeneous density and contain partial occlusions. The viability of both the individual abstraction steps as well as the overall framework is demonstrated in numerous simulations and experimental studies. The results show that a large number of structured objects can be learned from few examples with minimal supervision and robustly be recognized even in situations with heavy occlusions.

Zusammenfassung

Diese Dissertation stellt neue Erkenntnisse und Lösungen für das Problem der Wahrnehmung, Abstraktion und Interpretation der 3D-Struktur menschlicher Umgebungen mit autonomen mobilen Robotern vor. Zentrales Ziel der Arbeit ist die Gewinnung einer aussagekräftigen Repräsentation zur Erkennung von Objekten aus 3D-Laserdaten. In diesem Zusammenhang liefert sie einen wichtigen Beitrag zur Entwicklung kognitiver technischer Systeme, für die intelligente Perzeption eine Schlüsselfähigkeit darstellt.

Es wird eine Abstraktionskette entwickelt, die von rohen Punktdaten hin zu erkannten Objekten führt. Nach einer ausführlichen Begutachtung moderner Sensortechnologien werden neue Studien und Methoden zur Berechnung charakteristischer Punktmerkmale, der Segmentierung von Punktwolken sowie der Bayesschen teilebasierten Objekterkennung präsentiert. Das vorgestellte Rahmenwerk ist generisch und auf eine Vielzahl verschiedener Szenarien mit strukturierter 3D-Geometrie anwendbar. Das untersuchte Anwendungsszenario ist die Erkennung von Möbeln in häuslichen Umgebungen, eine essentielle Wahrnehmungsfähigkeit für Haushaltsroboter. Alle Methoden und Algorithmen wurden mit speziellem Augenmerk auf deren Effizienz und Anwendbarkeit auf realistische Datensätze konzipiert. Insbesondere können die gescannten 3D-Daten von relativ geringer und inhomogener Dichte sein und Verdeckungen beinhalten. Die Tragfähigkeit der einzelnen Abstraktionsschritte sowie des gesamten Rahmenwerks wird in zahlreichen Simulationen und Experimentalstudien demonstriert. Die Ergebnisse zeigen, dass eine große Anzahl strukturierter Objekte anhand weniger Beispiele mit minimaler Überwachung gelernt und robust wiedererkannt werden kann, sogar wenn das zu erkennende Objekt stark verdeckt wird.

Contents

Nomenclature

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
ACE	Autonomous City Explorer
ANN	Approximate Nearest Neighbors
AR	Angular resolution
CAD	Computer-aided design
CBLAS	Basic Linear Algebra Subprograms for C
CGAL	Computational Geometry Algorithms Library
CoTeSys	Cognition for Technical Systems
DoF	Degree of freedom
DR	Depth resolution
DT	Delaunay tessellation
EKF	Extended Kalman Filter
FoV	Field of view
h.o.t	Higher-order terms
IR	Infrared
kNN	k nearest neighbors
LR	Lateral resolution
LRF	Laser range finder
MAP	Maximum a-posteriori
MaR	Maximum range
MiR	Minimum range
MVBB	Minimum-volume bounding box
OOBB	Object-oriented bounding box
ORI	Omni-directional range image
PCA	Principal component analysis
PPS	Points per scan
RANSAC	Random sampling consensus
SAT	Seat, Armchair, Table (dataset)
SB	Stereo baseline
SfM	Structure from motion
SfS	Shape from shading
ST	Scan time
SVD	Singular value decomposition
ToF	Time of flight

Conventions

Scalars and are denoted by upper- and lower-case letters in italic type. *Vectors* are denoted by bold lower-case letters in italic type. *Matrices* are denoted by bold upper-case letters in italic type. *Sets* are denoted by upper-case letters in calligraphic type.

x or X	Scalar
$\boldsymbol{x}$	Vector
$\boldsymbol{X}$	Matrix
$\mathcal{X}$	Set
$\boldsymbol{X}^T$	Transpose of $\boldsymbol{X}$
$\boldsymbol{X}^{-1}$	Inverse of $\boldsymbol{X}$
$\boldsymbol{X}'$ or $\widetilde{\boldsymbol{X}}$	Transformed or modified matrix
$\hat{\boldsymbol{x}}$	Optimal, desired, or ground truth value of $\boldsymbol{x}$
$\bar{x}$	Average value of x
$\bar{\boldsymbol{x}}$	Column mean of $\boldsymbol{X}$
$f(\cdot)$	Scalar function
$\|\cdot\|_p$	p-norm
$\langle\cdot,\cdot\rangle$	Inner product

Symbols

$\mathbf{1}_n$	$n \times 1$ vector of ones
$\mathbf{1}_{m,n}$	$m \times n$ matrix of ones
a	Azimuth
$\boldsymbol{b}$	Parameter vector
$\boldsymbol{c}$	Surface coefficient vector
$\mathcal{C}$	Set of clusters
$\mathcal{C}_i$	ith cluster
d	Dimensionality of data items, $d = 3$ for spatial points
	Distance from view point
	Distance of local plane to origin
e	Elevation
$\mathcal{E}$	Edges of undirected graph $\mathcal{G}$
f_s	Scanning frequency
$\boldsymbol{f}$	Part feature vector
$g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$	State transition of EKF
$\mathcal{G}$	Undirected graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$
H	Mean curvature
h	Height of ORI
$h(\boldsymbol{x}_t)$	Measurement prediction of EKF
$\boldsymbol{H}_t$	Kalman observation matrix
$\bar{\boldsymbol{h}}_o$	Normalized histogram of object O

$\bar{\boldsymbol{h}}_{p,o}$	Mean part histogram of object O
$\bar{\boldsymbol{h}}_{r_k,o}$	Normalized relation histogram of object O
$\boldsymbol{h}_v$	View histogram
$\boldsymbol{I}_n$	$n \times n$ identity matrix
$\boldsymbol{J}$	Cost functional
K	Gaussian curvature
k	Number of nearest neighbors
k_r	Number of nearest neighbors within radius r
$\boldsymbol{K}_t$	Kalman gain matrix
$\boldsymbol{m}_i$	ith cluster mean
$\mathcal{M}$	Set of cluster means
N	Number of parts in a view
n	Number of points in a point cloud
	Number of dictionary words
n_b	Number of boundary lines
n_f	Number of part features
$n_{\min}$	Minimum number of points in segment/cluster
n_o	Number of objects
n_p	Number of part words
n_{r_k}	Number of relation words for relation of type k
n_v	Number of views
$\boldsymbol{n}_i$	Normal vector of the ith point, $\boldsymbol{n}_i = [n_{ix}\ n_{iy}\ n_{iz}]^T$
O	Object
P	Set of points in a point cloud
P_d	Detection threshold
$\boldsymbol{P}$	$n \times 3$ data matrix of a point cloud
$\bar{\boldsymbol{p}}$	Centroid of points
$\boldsymbol{p}_i$	ith point in a point cloud, $\boldsymbol{p}_i = [p_{ix}\ p_{iy}\ p_{iz}]^T$
$\boldsymbol{Q}_i$	$k \times 3$ neighborhood data matrix of ith point
$\boldsymbol{Q}_i^+$	$(k+1) \times 3$ augmented neighborhood matrix of ith point
$\boldsymbol{q}_{ij}$	jth neighbor of ith point
r	Distance threshold
$r_k(a,b)$	Relation of type k between parts a and b
$\boldsymbol{R}$	Data matrix with second order coordinate terms
S	Local plane
s_1, s_2, s_3	Singular values
$\mathcal{S}_j$	jth segment in a segmented scene
$\boldsymbol{S}$	$n \times d$ diagonal matrix with positive decreasing singular values
t	Runtime, computation time
$\bar{t}$	Average computation time per normal vector
$\hat{t}$	Desired computation time per normal vector
t_s	Scan time
$\boldsymbol{U}$	$n \times n$ orthogonal left matrix of SVD
$\boldsymbol{u}$	Unit ray direction vector

V	View
V_{oo}	Volume of object-oriented bounding box
$\boldsymbol{V}$	$d \times d$ orthogonal right matrix of SVD, contains singular vectors
$\boldsymbol{v}$	View point
$\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$	Singular vectors
$\mathcal{V}$	Vertices of undirected graph $\mathcal{G}$
w	Width of ORI
w_j	Weights used by averaging methods
$\boldsymbol{w}$	Feature weight vector
x, y, z	General point coordinates
$\boldsymbol{x}_t$, $\boldsymbol{x}_{t-1}$	Current and previous state for EKF
α_{FoV}	Angular field of view
$\alpha_{\max}$	Angular threshold
α_r	Angular resolution
γ	Overall quality of estimated normal vectors
γ_i	Quality of normal vector estimate n_i
δ	Mahalanobis distance
δ_p	Part detection threshold
δ_r	Relation detection threshold
δ_t	Measurement noise in EKF estimation
κ_1, κ_2	Principal curvatures
λ_1, λ_2, λ_3	Eigenvalues
	Variance along principal components
$\pi(\hat{t})$	Performance index
σ	Measurement noise standard deviation
σ_{ar}	Aspect ratio of points projected to the tangent plane
σ_{sv}	Surface variation
$\bar{\sigma}_{sv}$	Average surface variation
$\sigma_{sv,\max}$	Planarity threshold
$\boldsymbol{\Sigma}$	Covariance Matrix
ρ	General distance metric/measure
ρ_a	Angular distance
$\rho_{a,\min}$	Angular coplanarity threshold
ρ_c	Coplanar distance
$\rho_{c,\max}$	Coplanar merging threshold
ρ_e	Euclidean distance
$\bar{\rho}_i$	Average distance between $\boldsymbol{p}_i$ and its nearest neighbors
$\rho_{\max}$	Euclidean segmentation threshold
	Part clustering threshold
ρ_o	Orthogonal distance
$\rho_{o,\max}$	Euclidean coplanarity threshold
ρ_s	Search radius

1 Introduction

An NS5 Robot, doing some household chores
I, Robot, © 2004 Davis Entertainment

Summary

This chapter outlines the research context of this thesis. It describes the state of the art of autonomous mobile robotics and identifies several key challenges in the construction of cognitive technical systems. The research questions addressed in subsequent chapters are laid out and the contributions made by each chapter are highlighted.

Robotics research has been around for more than half a century. It has come a long way from pioneering the automation of industrial applications and has since evolved into a highly diverse and interdisciplinary field. As robots are gradually leaving factory floors and moving into human-populated spaces in which they encounter less structure and more uncertainty, there are many open research questions [134]. This thesis addresses several key research questions in the field of mobile autonomous robotics, which are outlined in this chapter. Recent developments in this field are described in Sec. 1.1, which identifies 3D perception as an especially important capability for intelligent robotic agents. Sec. 1.2 explains the layout of this thesis and highlights the contributions made in each chapter.

1.1 Developments in Autonomous Mobile Robotics

The construction of intelligent autonomous robotic agents that are able to assist humans in everyday tasks has been a long-standing goal of engineers and scientists alike [24]. Such robots would prove useful in numerous application scenarios, including fulfillment of household chores, assistance to health care personnel and factory workers etc. While the possible capabilities of such agents have been pictured down to the last detail in numerous fictional books and movies (see e.g. the image on the cover of this chapter), real-life systems are still in their infancy. The most advanced commercially available robotic systems that complete useful tasks in domestic environments today are robots for vacuum cleaning, pool cleaning, and lawn mowing [183, 187]. In most cases these robots have no idea of the exact appearance and state of their environment, but complete their tasks by means of primitive sensors and clever heuristics. The difficulty of building more advanced agents is not due to the inaptitude of researchers to improve any single hardware or software component; in fact impressive advances have been made in both domains over the past decades. Rather, the construction of agents that can interact with humans, understand context, handle uncertain information, learn from experiences, react robustly to surprise, and possess the sensorial, actuatorial and processing capabilities to do all of the above, requires a system complexity and sophistication that represents an exponential increase from today's systems.

The research cluster *Cognition for Technical Systems* (CoTeSys), within which the research of this thesis was conducted, is one of the large projects worldwide that aim to advance the state of the art of such agents. Its focus is on investigating mechanisms that turn technical systems from mere bit-crunching machines with pre-programmed routines into agents that 'know what they are doing' by equipping them with cognitive capabilities [25]. The general system architecture that describes these capabilities, called the *cognitive perception/action loop*, is shown in Fig. 1.1. As can be seen, the fundamental skill that serves as the basis for both *planning and cognitive control* as well as *learning and reasoning* is sensorial perception. In this context, perception is not limited to the mere recording of raw sensor data; rather it encapsulates all processes that serve to provide information about the state of the environment for higher level processing. Usually this involves abstracting raw data to a level at which it can be treated in a probabilistic, semantic or symbolic representation that serves as input to the planning and learning modules [25]. This thesis covers an instance of such a chain of abstraction that leads from raw sensor

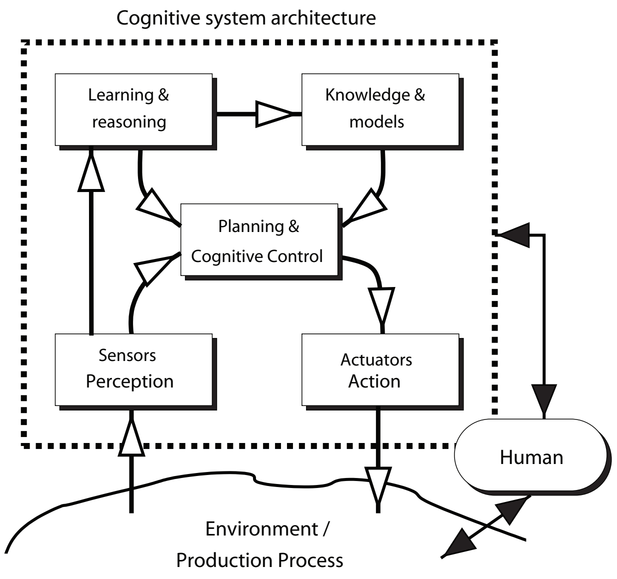

Cognitive system architecture
Learning & reasoning
Knowledge & models
Planning & Cognitive Control
Sensors Perception
Actuators Action
Human
Environment / Production Process

Figure 1.1: The cognitive perception/action loop from [25].

percepts (range measurements) to a compact representation of objects (recognized object classes represented by parts and spatial relations).

Research on cognition has evolved from early theoretical work on cognitive systems [99] into a vibrant and highly interdisciplinary topic, involving among others the fields of cognitive psychology, biology, neurology, mechatronics, control engineering, and computer science. Due to the advances in sensing, actuation and processing hardware, for the first time, the construction of non-trivial, real-life systems that possess sensing and actuation capabilities to actually perform complex tasks in human environments is within reach. While the development of *actuation modules*, such as mechanisms for locomotion, articulated actuators for manipulation and the respective control schemes, has important ramifications for the attainable level of autonomy, the *sensing modules* play an equally important if not more crucial role because other cognitive capabilities depend on robust perception, as previously outlined. Within the field of robotics, the development of sensing capabilities has been influenced significantly by the advent of affordable and accurate range sensing hardware. Fundamental problems in mobile autonomous robotics such as navigation, obstacle avoidance, localization, and mapping in indoor environments can nowadays be considered solved thanks to the availability of 2D laser range finders and the development of powerful algorithms that process this kind of data [145].

As mobile robotics research is gradually moving towards more complex scenarios, the ‘next big challenge’ is the understanding of 3D environments. In this context, ‘understanding’ entails the robust perception of 3D geometry, the abstraction of this geometry to a set of meaningful entities, and finally the interpretation of these entities to infer information about the state of the environment and to possibly perform manipulation or interaction. Such a level of environment understanding implies nothing less than putting autonomous robots on par with or at least on a comparable level to humans in terms of perception. While the motivation for attaining this level of understanding is thus very clear, by no means are the sensing methods and algorithms required to attain it. A recent survey on robotic vision identifies the understanding of 3D object shape and structure as well the categorization of objects as two big open research challenges [82].

While applicable to a large variety of scenarios, the framework presented in this thesis is focused on indoor robotics, more specifically the scenario of household robotics. This problem domain has received growing attention from research groups in both academia and industry over the past decade, because it poses the significant challenge of integrating numerous maturing research areas and has huge long-term economic potential. Some systems developed for household scenarios are:

- The *home exploring robotic butler* (HERB) robot, developed in a joint project of Intel Research and the Robotics Institute, Carnegie Mellon University [138]
- The *Home Assistant* robot, developed in a joint project of Toyota, Inc. and the Jouhou System Kougaku Laboratory, University of Tokyo [161]
- The *Assistive Kitchen* project, developed at the Intelligent Autonomous Systems Group, Technische Universität München [119]
- The humanoid robot ARMAR-III, developed at the Institute for Anthropomatics, Universtität Karlsruhe [126]
- The CoTeSys Multi-Robot Lab, developed jointly by several groups at Technische Universität München within the research cluster CoTeSys [25]
- The PR2 robot, developed by Willow Garage, Inc. [211]

The respective robots are shown in Figs. 1.2(a)-(f). The list of tasks that the robots in these projects are to perform include cleaning the room, setting the table, loading/unloading a dish washer, serving coffee etc. To this end, they are equipped with a plethora of sensors in order to tackle the problem of 3D perception. The bottom line for all of the aforementioned robots is that the ability to acquire some form of semantic understanding of the environment is a crucial prerequisite for any of the outlined tasks.

1.2 Contributions of this Thesis

This thesis is organized in six chapters, with chapters two through five presenting the main content. The chapters develop a chain of tools for the abstraction of 3D data and are best read in succession.

Chapter 2 This chapter provides a state-of-the-art overview of current sensing methods and presents the simulation and experimental setups used throughout the rest of the thesis. The overview basically summarizes today's possibilities of acquiring 3D data with a focus on the applicability of methods to mobile robotics scenarios. It was conceived for a lack of good recent surveys on the topic and represents a contribution as a stand-alone overview that can be read as a primer on 3D sensing.

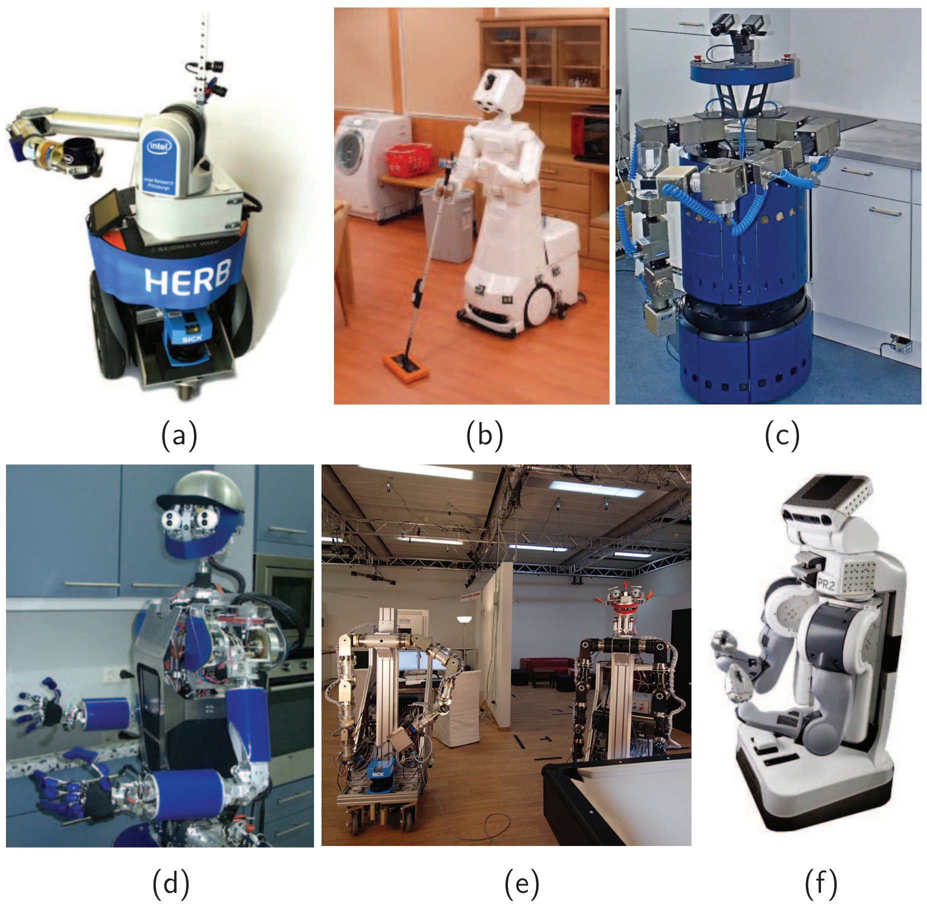


Figure 1.2: Household robots: (a) HERB, Intel/CMU [138] (b) Home Assistant Robot, Toyota/Univ. Tokyo [161] (c) AssistiveKitchen, TU München [119] (d) ARMAR-III, Univ. Karlsruhe [126] (e) CoTeSys Multi-Robot Lab, TU München [25] (f) PR2 Robot, Willow Garage [211].

Chapter 3 Building on the methods for 3D point cloud data acquisition presented in Chap. 2, this chapter introduces methods for computing numerous descriptive scalar and vectorial quantities known as *point cloud features*. The central contribution of this chapter is a rigorous analysis and comparison of methods for *normal vector estimation*. A performance evaluation of various existing normal vector estimation methods is presented, resulting in the identification of the most suitable method. Several new aspects, such as the estimation with an extended Kalman filter and the influence of sensor noise on estimation quality, are explored in simulation studies. Finally, various other descriptors, such as curvature and higher-order features, are presented with a demonstration of some of their drawbacks.

Chapter 4 This chapter presents a novel set of algorithms for the segmentation of point clouds into homogeneous surface regions. The presented algorithms use the normal vectors estimated in Chap. 3 and make up a toolchain that efficiently segments the relatively sparse point clouds obtained from an actuated 2D laser range finder on a mobile robot. They are, however, applicable to any kind of 3D range data, provided that i) it constitutes a

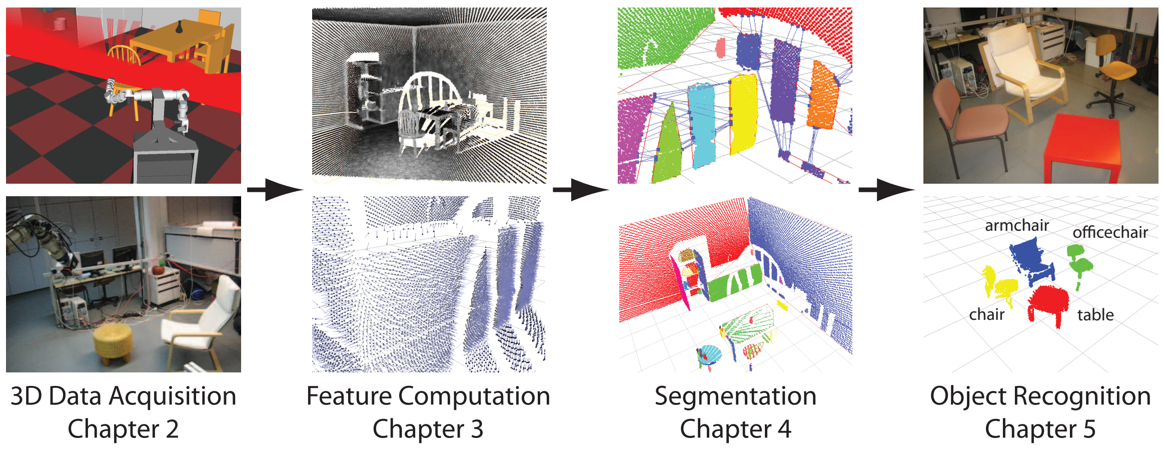


Figure 1.3: The chain of abstraction that leads from raw 3D points to recognized objects: Chap. 2 covers the acquisition of 3D data, Chap. 3 describes the computation of descriptive features, Chap. 4 details how to segment point clouds into homogeneous surfaces, and Chap. 5 explains how to use the obtained segments to perform object recognition.

surface sampling that allows for the estimation of normal vectors from neighboring points, and ii) a view point is known for every point in the set. The segmentation framework is enhanced by an efficient approach to repair oversegmentation caused by partial occlusions. Although real-life data sets are full of occlusions, they have remained largely unconsidered in research on segmentation. Therefore workable solutions such as the one presented in this chapter constitute a much-needed contribution to the design of more robust algorithms.

Chapter 5 In this chapter, the abstraction from points to segments is used for the development of a novel object recognition framework that operates with a part-based representation. A vocabulary of common parts encountered in different objects is automatically learned from a set of labeled object views. To robustify the recognition process, spatial relations between parts are introduced as additional dictionary words. From the parts detected in individual views, a compact histogram representation of objects is derived. A Bayesian formulation then allows for the classification of objects from posterior probabilities computed from the histograms. The use of a part/relation dictionary for point cloud data constitutes a new approach that differs significantly from traditional methods using local shape histograms. Both simulation and experimental results demonstrate that high recognition rates are achieved with the framework, also in instances where large portions of an object are occluded.

Fig. 1.3 illustrates the contribution of each chapter to the chain of abstraction of 3D data. Depending on the reader's interest, the chapters can also be read independently: Readers interested in an overview of current sensing hardware may refer to Chap. 2 as a stand-alone guide. Those interested in an in-depth study of methods for normal vector computation can proceed to Chap. 3. Readers familiar with point cloud features looking for an efficient way of surface segmentation may directly read Chap. 4. Finally, those interested in part-based object recognition can proceed to Chap. 5.

2 Acquisition of 3D Data

Eva, scanning her environment for life forms
WALL-E, © 2008 Disney/PIXAR

Summary

This chapter presents ways to acquire three-dimensional range data with suitable sensing hardware. Following a general overview of active and passive sensors, special focus is given to actuated laser range finders, which have become one of the most popular sensors for the acquisition of 3D data in the robotics community due to their affordability and accuracy. The simulation of this type of sensor data is discussed, and hardware setups used throughout this thesis are presented.

The desire to perceive and understand three-dimensional environments with artificial mechanisms is at least as old as the quest to build autonomous mobile robots. Historically, a great number of devices have been designed to capture the 3D structure of objects in a given environment, be it for medical applications, conservation of architectural heritage, virtual environments, urban planning, 3D navigation etc. The wide variety of applications for which such sensors are needed and used has resulted in a plethora of different sensing methods. A selection of these methods is summarized in this chapter with pointers to representative publications in each category. The approaches are presented with respect to their relevance for mobile robotic sensing. As a direct consequence, the overview covers only non-contact, markerless sensing modalities that yield a surface sampling of the objects of interest.

As a first step, similar surveys of 3D sensing are discussed in Sec. 2.1. A detailed overview of passive and active 3D sensors is presented in Secs. 2.2 and 2.3. These cover both specialized hardware systems that yield 3D data by exploiting a physical phenomenon as well as algorithmic methods that are to a certain degree independent of the underlying hardware. For the devices that are most relevant for mobile robotic sensing, characteristic specifications are provided. Sec. 2.4 explains suitable data structures and simulation methods for 3D laser range data, and Sec. 2.5 presents the hardware setups used for experiments in this thesis. Sec. 2.6 concludes the chapter with a brief discussion.

2.1 Related Work

With the rapid growth of robotics research in the 1980s, 3D sensing began to diversify and the increasing number of different sensors and concepts triggered the publication of surveys and overview papers on the topic. One of the earliest such overviews was published by Jarvis [69] in 1983, which remarkably already contained most of the sensing methods that are relevant today, with the exception of time-of-flight cameras, which did not exist at the time. The 1988 overview of Wang and Aggarwal [153] focused on active sensing methods and the reconstruction of geometric models. A little more than a decade later, the need for updated surveys on the topic yielded three such articles in the year 2000. Hebert presented a compact and comprehensive overview in [55], Rioux et al. provided a historical perspective on selected sensors in [55], and Beraldin et al. [14] published what seems to be the most comprehensive survey of active 3D sensing to date.

Following Beraldin et al. [14], 3D sensing can be taxonomized as illustrated in Fig. 2.1. The family of *contact* methods that involve physically touching (sometimes even destroying) the object in order to measure its extents, sense coordinates in a reference frame or attach markers, is not of interest for mobile robotic sensing as these methods require human intervention and a priori knowledge of the objects. The family of *non-contact* methods is suitable for mobile robots due to the fact that 3D structure can be sensed at a distance without physical contact. The methods typeset in boldface are covered in this chapter. *Stereo vision*, *shape from shading*, and *structure from motion* represent passive methods, and the rest (including *acoustic sensing*) represent active methods. Note that *marker-based* 3D measurement is considered a contact method, because even though the scanning process usually proceeds without contact, attaching the markers requires touch-

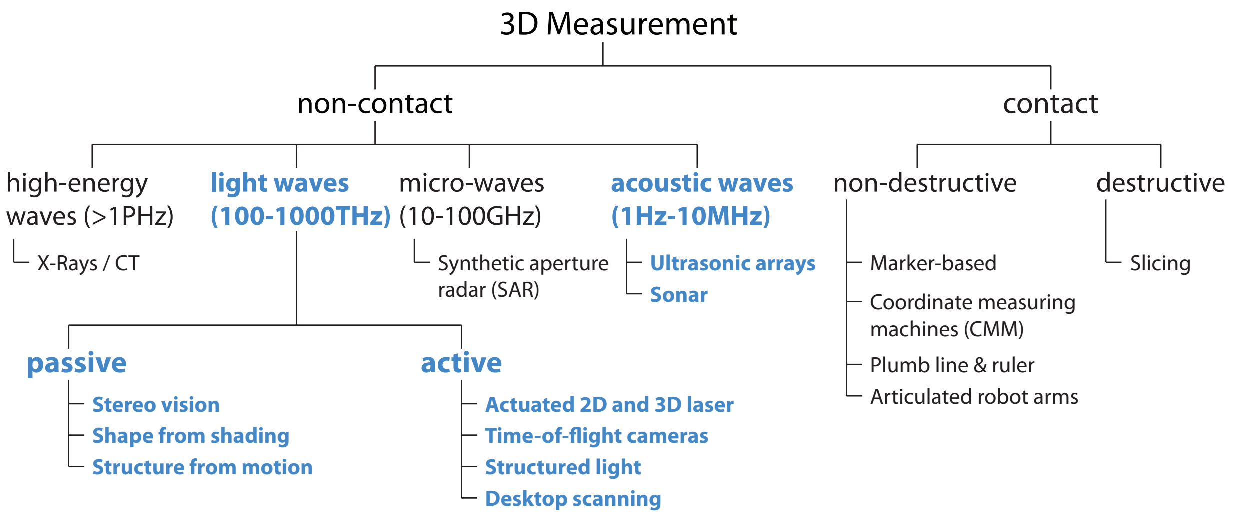


Figure 2.1: Taxonomy of 3D sensing methods, similar to that of Beraldin et al. [14]. The methods considered in this chapter (bold) fall under the category of non-contact, markerless sensing modalities that yield a surface sampling of the environment.

ing the object. Also, note that in this context, active sensing refers to the emission of energy in order to sense distance; it should not be confused with the more recent field of research that deals with the positioning of actuated sensors to maximize information gain, as surveyed in Mihaylova et al. [97].

The majority of non-contact sensing methods operate according to one of the three following principles:

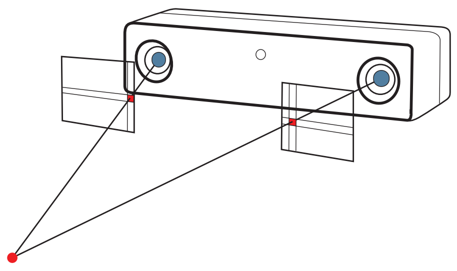
- **Triangulation**: 3D coordinates are computed from the appearance of salient points in different viewing planes. Knowledge of the geometric relation between these planes allows for direct calculation of depth using the law of cosines. Both active and passive triangulation methods exist.

- **Phase shift**: This principle is used only by active sensors. Modulated pulses of energy are emitted and superposition with the reflections allows to infer distance from the phase shift due to the wave nature of the emitted signal. The measurable distance is on the order of the wavelength and can be extended to some degree by the use of phase unwrapping algorithms.

- **Time of flight**: This principle is used only by active sensors. Pulses of energy are emitted and reflections are detected. With knowledge of the speed of the emitted waves in the surrounding medium, the elapsed time between emission and reception allows for the calculation of distance.

2.2 Passive Sensors

This section covers different types of *passive sensors*, that is, sensors which do not emit energy to measure distance, but infer depth information directly from energy present in the environment. In contrast to active sensors, this confines them exclusively to light-based sensing, mostly in the visible range.



Operating principle: Two or more cameras view a scene from a fixed stereo baseline and slightly different angles. Corresponding pixels are found in each image and depth is triangulated from pixel displacement.

Pros: Passive method, no energy emitted
Relatively fast

Cons: Poor depth resolution
Dependent on surface texture
Short range

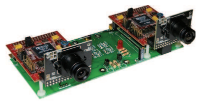
			
LR: 752x480 DR: 2mm@0.5m, 6.5cm@4m SB: 12cm and 24cm	LR: 752x480 DR: 3mm@0.5m, 10cm@3m SB: 6cm	LR: 752x480 DR: n/a SB: 10.75cm	LR: 1280x960 DR: n/a SB: 9cm
PointGrey Bumblebee XB3 [199]	Focus PCI nDepth [182]	Surveyor SVS [206]	Videre STH-MDCS [210]

Figure 2.2: Stereo vision sensors yield medium-accuracy depth values from optical triangulation. Top: Operating principle, pros and cons. Bottom: Some commercially available stereo vision systems with characteristic specifications.

2.2.1 Stereo Vision

As one of the most active fields in computer vision research, stereo vision methods aim to replicate the fundamental mechanism that enables humans and animals to superbly perceive their three-dimensional surroundings: images from two or more cameras which are spaced at a known distance (the stereo baseline) and observe the same scene from slightly different angles allow for the computation of distance using the principle of optical triangulation as illustrated in Fig. 2.2. The cameras need to be calibrated to establish the parameters of their geometric transformation and sometimes focal lengths, image centers and aspect ratios. Given these parameters, the epipolar constraint allows to reduce the problem of finding corresponding pixels in both images to a one-dimensional search along the horizontal image lines [9]. Once two corresponding pixels have been found, their displacement and knowledge of the stereo baseline allow for the calculation of the depth value. Stereo vision systems operate in four processing stages: 1) recording of the raw image with each camera, 2) image rectification, 3) search for corresponding pixels by means of a suitable algorithm, and 4) triangulation. While steps 1) and 2) represent mere pre-processing, step 3) is non-trivial and represents the computationally most intensive step that has been the focus of a great amount of research. An overview and performance evaluation of different stereo correspondence algorithms can be found in Scharstein and Szelisky [124]. The result of this step is the disparity map which provides inverse distances for every pixel in the common plane of both cameras. Step 4) consists of inverting these values and using the stereo baseline to calculate actual distances.

Commercially available systems usually house the cameras in a rigid and fixed casing, which alleviates most of the required calibration effort. Popular systems include the Bumblebee XB3 camera by Point Grey Research, Inc. [199], the nDepth Vision System by Focus Robotics [182], the SVS Stereo Vision System by Surveyor Corp. [206], and the STH-MDCS camera by Videre Design LLC [210]. While the hardware setup of these cameras is largely identical, with minute differences in baseline and resolution, the software processing (mainly stereo correspondence) is what determines the frame rates at which 3D data is actually obtained. To this end, some vendors provide extensive SDKs and leave the stereo matching phase to the user [199], others include special-purpose processing cards that yield out-of-the-box 3D data [182]. Typical frame rates are 1-20Hz for 640x480 depth images with 150 disparity levels.

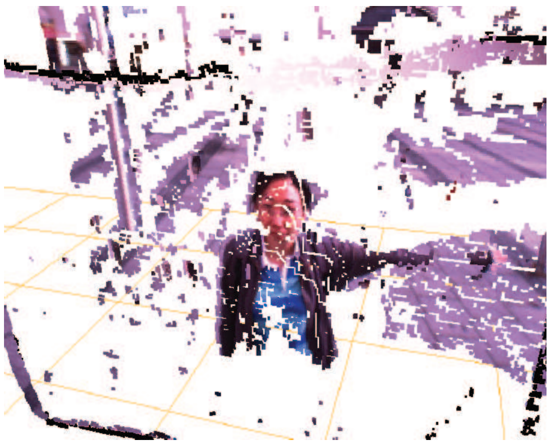


Figure 2.3: Stereo vision data comes in 'depth slices' that are spaced unevenly across the covered range. While depth accuracy may be on the order of millimeters for close-by objects, it deteriorates to up to 10 centimeters for objects that are several meters away. The point cloud shown in this figure was recorded with a Bumblebee 2 on the system described in [166].

Because stereo vision systems compute distance from pixel displacements, the resolvable depth is limited by the resolution of the camera images and the stereo baseline. In practice this means that stereo vision is only suitable for a range of a few meters, which is resolved e.g. at 150 disparity levels. Resolution can be improved by fusing information from more than two cameras, e.g. the three-camera setup provided by the Bumblebee XB3. This, however, lowers achievable frame rates. In general, depth is resolved inhomogeneously over the perceived range; that is, while objects that are less than half a meter away can be perceived with millimeter accuracy, at four meters distance accuracy may be as poor as six centimeters or worse [182, 199]. With a two-camera setup depth information is therefore essentially obtained in unevenly spaced slices as illustrated in Fig. 2.3, which shows data obtained with the two-camera Bumblebee 2 sensor. Especially for the estimation of surface features this may render data beyond a few meters unusable and severely limits the actually useful range of stereo systems.

2.2.2 Shape from Shading

This is by far the most minimalistic approach in the family of passive sensing methods. Shape from shading (SfS) aims to infer the shape of a three-dimensional object from the brightness values of a *single* two-dimensional grayscale image of that object [112]. The idea was conceived by Horn [63] in the 1970s and further developed by a growing community of researchers in the 1980s, see Brooks et al. [23]. Calculation of 3D shape is possible under certain assumptions (orthographic camera, Lambertian reflectance, single point-like light source) from the so-called brightness equation [63], a nonlinear first-order differential equation. Numerous variations and improvements of the method have been developed, summarized in Zhang et al. [164] and Prados et al. [112]. While it has provided fundamental insights into how much 3D information can be inferred from individual 2D images, SfS has mostly remained a research problem without real-life applications. Two potential usage scenarios suggested in [112] are the estimation of page distortion in book scanning applications as well as rudimentary face reconstruction. As such, SfS does not require any specific hardware other than a single camera and suitable lighting and reflectance conditions. It has no relevance for mobile robotic sensing, save that some insights can be transferred to other sensing modalities. SfS can be viewed as a special case of photometric stereo with only one illuminated view, see Sec. 2.3.7.

2.2.3 Structure from Motion

Structure from motion (SfM) methods build on the fact that the 3D information of a scene can be inferred from a sequence of 2D images taken from varying view points. In fact, humans perceive a large amount of spatial information in 3D environments by moving through them [131]. The idea of SfM was first formalized by Ullman in the 1970s [149]. In most cases, the image sequence is assumed to be continuous in time (and thus to result from movement), however, several works compute 3D structure from an unordered collection of images. SfM is closely related to stereo vision in that depth information is computed using triangulation of several views. However, in contrast to stereo, where the geometric transformation between two or more camera images obtained at the same time is known, in SfM the transformation between subsequent images must first be estimated from the image data itself. In a nutshell, this is done by extracting and tracking characteristic features from frame to frame, which allow for an online 'calibration' of the camera. With sufficiently many features both the camera motion as well as the scene structure can be computed down to a scaling factor that remains unknown because no absolute reference (such as the stereo baseline for two cameras) exists. A detailed explanation and literature overview can be found in Huang and Netravali [66] and Jebara et al. [70].

Similar to shape from shading, SfM represents an algorithmic approach that is not dependent on specific hardware. The raw data usually consists of a stream of images obtained with one camera. Commercially available software provides solutions for camera tracking, 3D mesh reconstruction, image stabilization etc. from such data and is offered by companies such as 2d3 Ltd. [177] and Vicon [209]. On the research side, many advances have been made in SfM over the last two decades. Robust self-calibration for zooming cameras was demonstrated by Pollefeys et al. [110]. Dellaert et al. presented an iterative

SfM algorithm that works without the need to find feature correspondences [33]. Snaveley et al. [137] showed that the principle of view point estimation and 3D reconstruction from extracted features can also be applied to an unordered collection of images from different cameras, e.g. user-submitted photographs on web portals such as flickr.com from which 3D models of popular tourist sights can be reconstructed.

Generally, SfM methods are computationally intensive, and processing time for an image sequence can easily be on the order of minutes. In mobile robotics they have become increasingly relevant due to the advances in processing hardware that allow for near-real-time computation of reconstructed surfaces. Examples of such GPU-assisted SfM approaches for indoor and outdoor scenarios have e.g. been presented by Bartzak et al. [11] and Koeser et al. [80]. However, the dependency of SfM on sufficient environment structure for feature tracking limits their applicability in comparison to active sensors.

2.2.4 Other Methods

Other passive sensing approaches include variations of the above-mentioned techniques with infrared light, e.g. infrared stereo vision for pedestrian detection [15] and 3D reconstruction [120]. Furthermore, there exist a number of 'niche' techniques that can be summarized as 'shape from x', including *shape from texture* [89], *shape from focus* [104], and *shape from silhouette* [29], which target special reconstruction scenarios and are not suitable as general sensing approaches for mobile robotics.

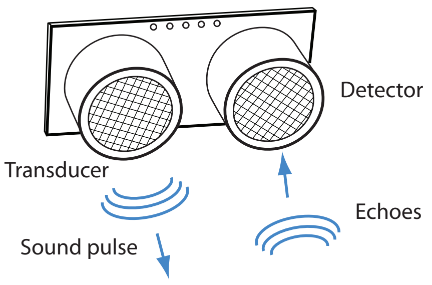
2.3 Active Sensors

This section describes the large collection of sensors that emit energy in order to measure distance. They are therefore called *active sensors*.

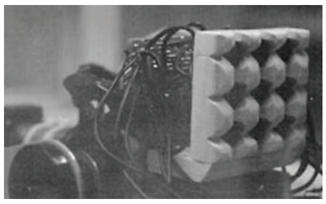
2.3.1 Acoustic Sensors

One of the earliest promising technologies for range sensing in robotics were acoustic sensors. The operating principle is illustrated in Fig. 2.4. To avoid interference with the human hearing range, acoustic range sensors usually operate at ultrasonic frequencies. Transducers convert electrical energy into ultrasonic sound waves or pulses that are emitted in the direction of interest. The echoes generated by objects in the environment are captured with suitable detectors, and distance is inferred from the elapsed time and knowledge of the speed of sound in the given environment conditions.

Low cost and – at the time – comparably high accuracy of ultrasonic range sensors led to their adoption for mobile robot navigation in the 1980s [52, 102, 128]. The technology matured in the course of the 1990s, when the first commercially available research robots such as the B12 by RWI (now iRobot Corporation), the widespread Pioneer by MobileRobots, Inc. and similar models by other companies saw the light of day. The general setup of these robots is a circular array of ultrasonic sensors that measure distance values in a 2D plane surrounding the robot to detect obstacles and perform mapping tasks [145].

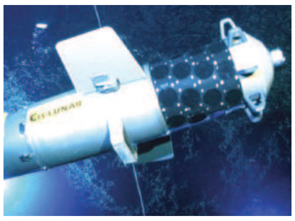


Operating principle:	Pulsed sound is emitted from one or several transducers. Reflections are detected and distance is calculated from the time of flight.
Pros:	Independent of illumination conditions, reflectance etc. Works underwater
Cons:	Low accuracy, low resolution Distorted echoes cause systematic noise Relatively low acquisition rates



Type:	Ultrasonic
Array:	8 Transducers
Acuisition Frequency:	n/a
Resolution:	n/a

University of Cantabria [88]



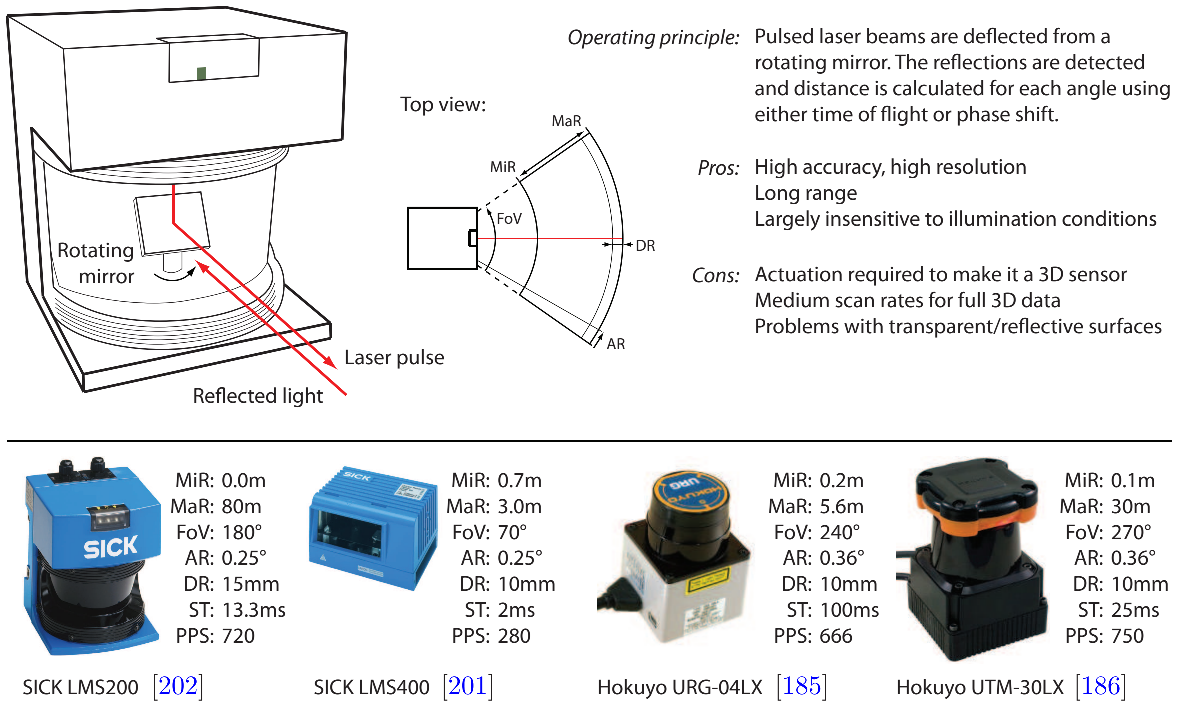
Type:	Sonar
Array:	32 Transducers
Acquisition Frequency:	4Hz
Resolution:	<10cm

Digital Wall Mapper [4]

Figure 2.4: 3D acoustic sensing is a niche field in robotics. Apart from comparatively poor accuracy and low acquisition frequencies, sensors have low resolution due to the difficulty of building dense transducer arrays. Areas of application include scenarios in which other sensors fail, e.g. mapping of underwater environments. Top: Operating principle, pros and cons. Bottom: Two of the few 3D acoustic robotic sensing setups.

Independent of this proliferation of ultrasonic sensing for 2D navigation, researchers started examining the use of acoustic devices for 3D range sensing early on. The plethora of publications that cover ultrasonic medical scanners is summarized in [113] and not considered here, as these scanners yield a volumetric sampling of stationary objects with which they are in contact and are thus not relevant for mobile robotics. Within the context of surface sampling, seminal work by Acampora and Winters [1] showed that the *acoustic signature* of 3D objects is highly discriminative and can – under modest assumptions – in principle serve as a replacement for optical sensing in recognition applications, avoiding the need to construct visual representations of the objects of interest. Such 3D object recognition from acoustic sensors was demonstrated by Watanabe et al. [156], who learned and recognized geometric primitives from low resolution data obtained with an 8x8 ultrasonic transceiver array using artificial neural networks. In a more recent work, Llata et al. [88] demonstrate shape reconstruction of primitive geometric objects with a low-resolution ultrasonic array. In general, 3D acoustic range sensing suffers from poor resolution due to large carrier wavelength and the difficulty of building dense arrays of ultrasonic sensors. A further problem in multi-robot scenarios is the interference of sound signals from several sensors. Depth accuracy remains unspecified in several works, but is generally on the order of centimeters for ultrasonic sensors.

Despite the successful proofs of concept, 3D range sensing using acoustic hardware has remained a niche field, mainly because of low resolution and poor accuracy in comparison to other sensors [48]. There are several applications where acoustic sensing is the method of choice, usually in scenarios where active and passive light-based sensing fail due to poor surface reflectivity or difficult illumination conditions. Examples for this include mapping of underwater environments with arrays of low frequency sonars [4] as well as mapping of caves with actuated ultrasonic sensors [130].



Operating principle: Pulsed laser beams are deflected from a rotating mirror. The reflections are detected and distance is calculated for each angle using either time of flight or phase shift.

Pros: High accuracy, high resolution
Long range
Largely insensitive to illumination conditions

Cons: Actuation required to make it a 3D sensor
Medium scan rates for full 3D data
Problems with transparent/reflective surfaces

	SICK LMS200 [202]	SICK LMS400 [201]	Hokuyo URG-04LX [185]	Hokuyo UTM-30LX [186]
MiR	0.0m	0.7m	0.2m	0.1m
MaR	80m	3.0m	5.6m	30m
FoV	180°	70°	240°	270°
AR	0.25°	0.25°	0.36°	0.36°
DR	15mm	10mm	10mm	10mm
ST	13.3ms	2ms	100ms	25ms
PPS	720	280	666	750

Figure 2.5: 2D Laser Range Finders provide accurate range values in the sensor plane. Top: Operating principle, pros and cons. Bottom: Specifications of the most popular laser range finders used in mobile robotics.

2.3.2 Actuated 2D Laser Range Finders

The advent of affordable and accurate laser range finders (LRFs), also referred to as LIDAR (light detection and ranging) or LADAR (laser detection and ranging) sensors, triggered tremendous developments in the field of mobile robotics starting in the 1990s. Virtually all research on fundamental robotic skills such as localization, mapping, and navigation gradually transitioned from the previously used ultrasonic or other distance sensors to laser range finders for the leap in accuracy and resolution that these provided [145].

A detailed discussion of the many different design aspects of laser-based range sensors (wavelengths, micropulse vs. high energy, receiver sensitivity etc.) is beyond the scope of this overview. The general operating principle of 2D LRFs used in mobile robotics is illustrated in Fig. 2.5. A pulsed laser beam is emitted from a laser diode and sent onto a rotating mirror that deflects it in a fan-shaped scan profile. The light that is reflected from objects in the environment is registered with highly sensitive detectors that associate the response with the angle at which each beam was sent off. Depending on the specific type of sensor, the range values are either calculated using the time-of-flight principle or the phase shift between emitted and reflected light. Distance values are calculated for every angle at which pulses were emitted, yielding a dense sampling of the environment in the sensor plane within its field of view.

2D LRFs were originally developed in the context of industrial automation applications and have found widespread use in the robotics community for over a decade. The most commonly used sensors are the LMS-200 (along with its outdoor variant LMS-291) and the LMS-400 manufactured by SICK AG [201, 202] as well as the recent and more compact URG-04LX and UTM-30LX scanners by Hokuyo Automatic Co., Ltd. [185, 186]. All but the LMS-400 use class-1 laser light invisible to the human eye.

Soon after the adoption of LRFs for 2D navigation, roboticists started building hardware setups that allowed to move or 'sweep' the sensors to obtain three-dimensional data. Up to today, the method of actuating 2D range finders represents one of the most popular approaches to acquire 3D range data within the robotics community due to its relatively low cost and high accuracy. The method requires mounting a 2D LRF on a robot actuator consisting of one or several degrees of freedom (DoF). Knowledge of the forward kinematics of the mechanism and the value of the joint angles allow for calculation of the 3D pose of the LRF, which in turn makes it possible to compute the 3D coordinates of the sensed points with respect to the robot coordinate frame. Estimation of the robot pose within the world – for mobile robots inferred from odometry and/or inertial sensor data – then allows for the transformation of the sensed data into a global coordinate system.

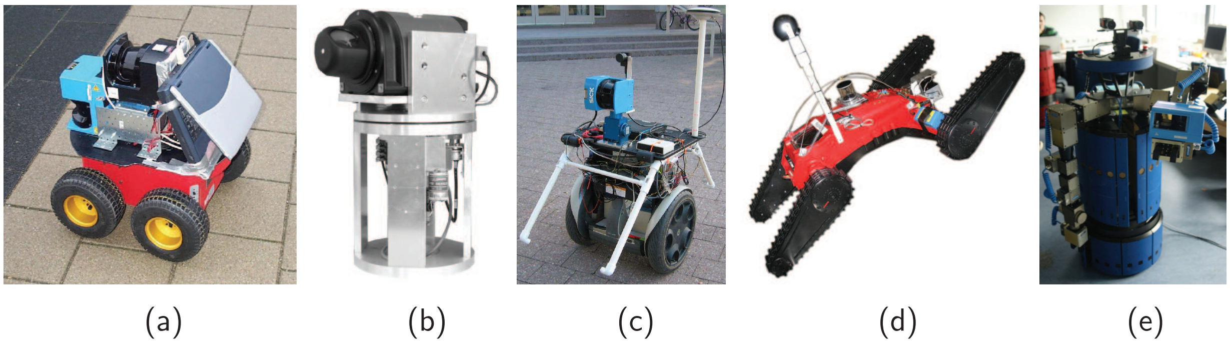
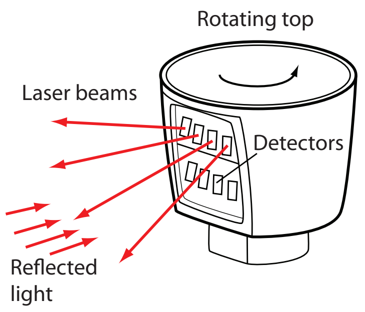


Figure 2.6: Examples of actuated 2D laser range finders: (a) SICK LMS-200 mounted directly on a Pioneer P3-DX robot [53, 144] (b) IBEO/SICK LRF mounted on a continuously rotating servo drive [158] (c) SICK LMS-200 mounted on an Amtec panning actuator attached to a tilting Segway RMP [101] (d) Hokuyo URG-04LX mounted on a pan/tilt module at the front of a Redback robot [132] (e) SICK LMS-400 mounted on a 6-DoF PowerCube arm attached to a B21 robot base

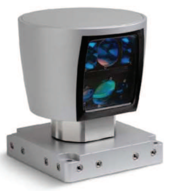
Over the past decade numerous ways of actuating 2D LRFs have been developed, some of which are illustrated in Figs. 2.6(a)-(e). One of the earliest and simplest approaches used by Thrun et al. was to mount a SICK LMS-200 directly on a Pioneer P3-DX robot [53, 144]. As the robot turns on the spot, the upward looking LRF captures a dome-shaped 3D scan. This data was used for autonomous 3D mapping of indoor environments. Later on, Wulf and Wagner developed a fast 3D scanner consisting of a 2D IBEO/SICK LRF mounted on a continuously rotating servo drive [158]. Cable wind-up usually forces rotating scanners to alternate the turning direction, which implies constant de-/acceleration and additional strain on the hardware. This problem was solved by substituting cables with slip ring connections, allowing for smooth and continuous rotational scanning. As part of the Stanford SegBot project, Montmerlo et al. mounted a SICK LMS-200 vertically on a horizontally panning Amtec actuator, which was in turn attached to a Segway Robotic Mobile Platform (RMP) [101, 105]. Due to the balancing control of the Segway, the robot was in a permanent tilting motion and its 3D pose was continuously estimated using the onboard inertial sensors. It was successfully deployed for 3D mapping of the Stanford campus along a 10km trajectory. A smaller outdoor robot was devised by Sheh et al., who mounted a Hokuyo URG-04LX on a pan/tilt unit at the front of a Redback robot [132]. The robot can flip over, change its scanning configuration and perform either vertical or horizontal sweeps with the LRF. An elaborate scanning setup was presented by Rusu et al., who used the fast, close-range SICK LMS-400 mounted on a 6-DoF articulated PowerCube robot arm attached to a B21 robot base to acquire dense scans of a kitchen environment [119]. This setup is similar to the main system used throughout this thesis, see Sec. 2.5, and allows for the greatest variety of scanning trajectories. Many more scanning setups exist, that have not been included in the selection presented in Fig. 2.6; however, the shown systems represent the most commonly used sweeping trajectories in robotics.



Operating principle: Multiple laser pulses are emitted, or one beam is deflected in several directions. Reflections are deteced and distance is calculated from the time of flight.

Pros: Complete 3D scan in each frame
High accuracy, long range
Large field of view

Cons: Very high cost
Tripod scanners not suitable for mobile operation



MiR:	0.0m
MaR:	50m
FoV:	360°x27°
AR:	0.09°x0.4°
DR:	20mm
ST:	66ms
PPS:	88.8k

Velodyne HDL-64E [208]

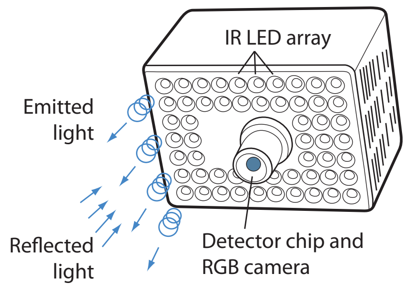
MiR:	0.1m
MaR:	134m
FoV:	360°x270°
AR:	0.003°x0.003°
DR:	6mm
ST:	~2min
PPS:	6,000k

Leica ScanStation C10 [190]

Figure 2.7: 3D Laser Range Finders provide accurate range values in a large field of view. Top: Operating principle, pros and cons. Bottom: Specifications of two very different 3D scanning systems.

2.3.3 3D Laser Range Finders

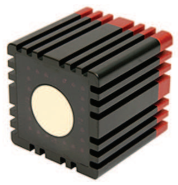
Apart from the low cost, self-constructed 3D scanning setups described in the previous section, several companies offer professional 3D LRF systems. The operating principle is the same as that of 2D LRFs; the 3D systems have, however, either more laser beams which are sent off in different three-dimensional directions, or the mirror is actuated in a more elaborate way than mere rotation in a plane. Fig. 2.7 shows two such systems for very different fields of application. The Velodyne HDL-64E was created for providing dense 3D data at high acquisition rates to vehicles for autonomous navigation [208]. It is one of the most advanced (and most expensive) 3D scanners ever built for robotics applications, surveying the environment with 64 laser beams that yield over 1.3 million points per second. A completely different but comparably expensive system is the Leica ScanStation C10 that is targeted towards civil engineering applications [190]. A single laser beam is deflected from a vertically rotating mirror on a horizontally rotating base mounted on a tripod. The scanner is slow (the acquisition takes several minutes) but has high accuracy and is intended for detailed surveying of static scenarios, such as capturing the shape of building façades. The data is further enhanced by high-resolution photographic imagery.



Operating principle:	An array of infrared LEDs emits pulsed light. Reflections are captured and distance is calculated using phase shift or shuttered light portions.
Pros:	Complete depth image in each frame Color information High acquisition rates
Cons:	Medium range High random and systematic noise Comparatively low resolution

MiR: 0.3m
MaR: 7.0m
LR: 204x204
DR: 3mm
ST: 40ms
PPS: 41.6k

PMD[vision] CamCube 2.0 [191]



MiR: 0.8m
MaR: 5.0m
LR: 176x144
DR: 10mm
ST: 20ms
PPS: 25.3k

SwissRanger 4000 [198]

MiR: 0.5m
MaR: 2.5m
LR: 320x240
DR: <20mm
ST: 16.6ms
PPS: 76.8k

ZCam [193]

MiR: 0.8m DR: 10mm
MaR: 3.5m ST: 16.6ms
LR: 640x480 PPS: 307.2k

PrimeSensor [200]

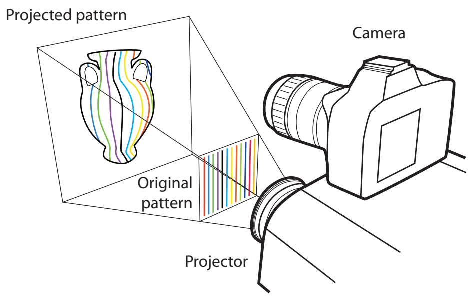
Figure 2.8: Time-of-flight cameras provide color images with associated depth values at video frame rates. Top: Operating principle, pros and cons. Bottom: Specifications of commercially available ToF cameras.

2.3.4 Time-of-flight Cameras

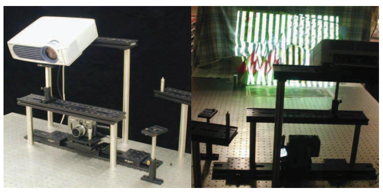
Time-of-flight (ToF) cameras are relatively recent devices whose operation at useful resolutions has become possible thanks to the advances in semiconductor and solid state technology. Current systems are able to capture low- to mid-resolution color images with associated depth values at real-time acquisition rates. Depth is measured by emitting modulated, incoherent light from arrays of infrared diodes and calculating the distance values from the detected reflections, as illustrated in Fig. 2.8. The depth data is then fused with the color image of an RGB camera. There are two families of ToF cameras with different operating principles [81]: the first infers distance from the phase shift of reflections. This principle is utilized by the majority of commercially available cameras, such as PMDTec's CamCube 2.0 [198] and Mesa Imaging's SwissRanger 4000 [191]. The second and even more recent technology uses a shuttered solid-state CCD sensor and a phase detector to split the reflected light into two portions, from which depth and intensity values can be computed separately [95]. One of the few commercially available devices using this principle was the ZCam by 3DV Systems [159, 193], which was bought out by Microsoft Corporation for its promising potential in interactive gaming applications. The release of this device, which is currently developed under the name 'project Natal' for the XBox 360 game console, is scheduled for late 2010 and could greatly impact the availability of low-cost 3D sensing technology for indoor robotics applications. Similar OEM devices are currently developed by PrimeSense Ltd. [200] and Canesta, Inc. [180].

While in principle ToF cameras yield depth resolutions on the order of one centimeter, the data exhibits inhomogeneous random and systematic noise, e.g. at object boundaries [81]. Due to the low resolution of individual images (mostly below 320x240), individual frames are therefore little suited for mapping and recognition applications. Various researchers have examined superresolution techniques which fuse information from several frames to increase both lateral and depth resolution of the 3D data [84, 129]. This, however, puts the acquisition times far out of the reach of real-time applications (up to several minutes) due to computationally costly optimization algorithms [129].

At present, 3D data acquired with ToF cameras is still of inferior quality in comparison to data acquired with other active sensing methods, such as LRFs. However, with the constant advances in semiconductor technology both lateral and depth resolution of ToF cameras can be expected to increase significantly over the coming years. The attractive acquisition rates at which colored depth data is delivered by these sensors is then likely to make them one of the predominant sensing modalities for indoor robotics scenarios.



Operating principle:	A projector casts a structured light pattern onto the scene. A high-resolution camera captures the projected pattern and depth values are calculated using triangulation between corresponding pixels of both patterns.
Pros:	Very high resolution Relatively fast Low cost
Cons:	Requires controlled illumination of the scene Low range Bulky, stationary setup


Middleburry College [123]

Heinrich-Hertz-Institute [39]

GFM TopoCam [184]

Figure 2.9: Structured light scanners yield 3D data of extremely high resolution. Top: Operating principle, pros and cons. Bottom: Selected systems used in research and industry.

2.3.5 Structured Light Scanners

In contrast to the afore-explained sensors that utilize time-of-flight or phase information of reflected light to infer distance, the sensor systems described in this section consider the visual appearance of structured patterns (also called fringe images) projected onto the objects of interest. The operating principle is illustrated in Fig. 2.9. A strong projector is used to cast a discriminative black and white, gray-scale or color pattern, often consisting of stripes with sinusoidal intensity change, onto the environment. A high-resolution digital camera captures images of the scene containing the projected stripe pattern. Having established the correspondences between the original and the projected pattern, it is then possible to compute depth values from pixel displacements using optical triangulation. The main challenge lies in correctly resolving ambiguities when searching for pixel correspondences, which is facilitated by using maximally unambiguous patterns, such as Gray codes for monochrome projectors [123] and Bruijn sequences for color projectors [163]. Ambiguity can be further reduced by consecutively projecting each pattern and its inverse. Structured light scanners are able to yield extremely high depth resolution (down to the micron range) by varying the position and spatial resolution of the projected patterns.

The structured light technique has been used for various purposes in research. Scharstein and Szeliski [123] employed a self-constructed setup with multiple patterns to retrieve accurate ground truth data for the evaluation of stereo vision systems. Zhang et al. [163] demonstrated accurate shape reconstruction of 3D scenes from a single color stripe pattern using a dynamic programming technique. Zhang and Huang [165] presented an impressive self-constructed synchronized projector-camera system that yields high resolution 3D data at 40 frames per second using sinusoidal RGB patterns and a parallelized computation pipeline. A system for capturing high-resolution 3D face models with off-the-shelf projector and camera components was demonstrated by Fechteler and Eisert in [39]. They used an adaptive color classification scheme to robustify the technique against variable illumination.

Examples of commercially available structured light sensors include the TopoCam system by GFMesstechnik GmbH [184] for the accurate measurement of industrial mechanical components as well as dental scanners such as the Maestro 3D Dental scanner by AGE Solutions S.r.l. [178] or the Rexcan DS2 by Solutionix Corp. [204] that perform high-resolution scans of teeth models. While structured light techniques have a number of advantages, such as very high resolution and (in some cases) high acquisition rates, their dependency on benign illumination conditions makes them suitable only for a very limited range of applications. In mobile robotics scenarios, environment conditions may not allow for the projection and detection of patterns or, when working in human environments, the projection of structured light in the human visible range may simply not be acceptable. Structured light scanners therefore have little relevance for mobile robotic sensing.

2.3.6 Desktop Scanning

A specialized field of 3D data acquisition is desktop scanning, which aims to reconstruct detailed textured CAD models of stationary small and mid-sized objects. Objects are sometimes placed on a rotating plate in front of the scanner to facilitate the registration of several recorded views. Other systems are capable of registering views autonomously while the user performs handheld scanning. The devices usually employ mixtures of the technologies described in the previous sections. The popular low-cost NextEngine 3D Scanner by the homonymous company uses several laser beams and a color camera to acquire extremely dense colored meshes of objects placed on a rotating plate; the acquisition of each view takes about two minutes [197]. The Artec 3D Scanner manufactured by the Artec

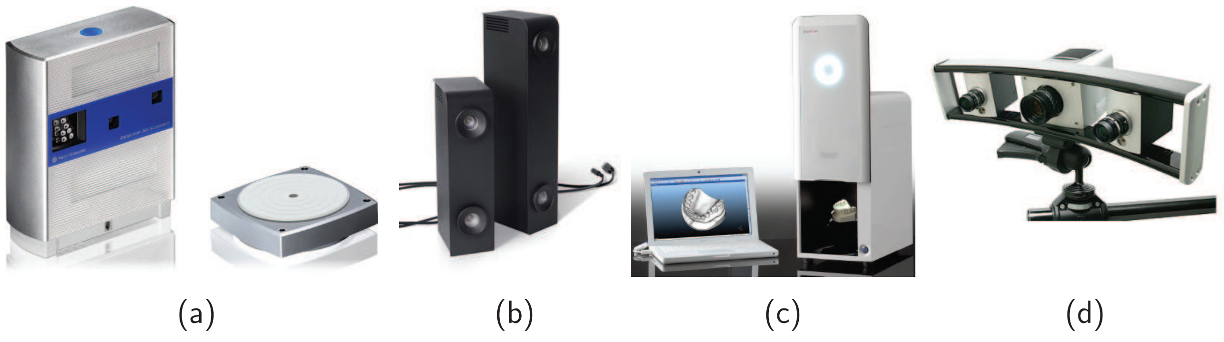


Figure 2.10: Desktop scanning devices intend to construct detailed textured meshes of close-by stationary objects. (a) The NextEngine 3D scanner uses laser beams and a camera to scan objects on a rotating plate [197]. (b) The handheld Artec 3D Scanner registers flash-bulb illuminated views into one mesh [179]. (c)-(d) The Solutionix scanners Rexcan DS2 and Rescan III scan teeth models and mechanical parts using structured light in combination with stereo vision [204, 205].

Group is a stereo vision-based handheld scanning solution that yields registered views at 15 frames per second using a flash light bulb to illuminate the scene at periodic intervals. The scanners Rexcan DS2 and Rexcan III produced by Solutionix Corp. yield detailed meshed data of teeth models and mechanical parts, respectively, using a combination of structured light and stereo vision [204, 205].

In general, desktop scanners are limited by their short range. Their focus is on the acquisition of detailed models for purposes of visualization, reverse engineering, 3D design etc. Apart from building high-resolution model libraries for object recognition applications, they therefore have little relevance for mobile robotic sensing.

2.3.7 Other Methods

The methods presented in the previous sections constitute the most commonly used active sensing approaches for acquiring 3D data. Other methods are briefly described in this section.

Similar to structured light approaches, *photometric stereo* infers the 3D shape of objects by changing the illumination of the scene in a controlled way. The idea was proposed by Woodham [157] and builds on the fact that object shape can be recovered from several images taken from the same view point under different lighting conditions. As with shape from shading methods, see Sec. 2.2.2, surface appearance is computed from the brightness equation. Original limiting assumptions such as Lambertian reflectance and known point-like light sources have since been overcome, which makes the method applicable to some real-life applications, e.g. face recognition [7] and material segmentation [56]. However, similar to structured light approaches, dependency on benign lighting conditions and on the illuminability of the scene as well as short range render the method unsuitable for mobile robotic sensing.

Further down the electromagnetic spectrum, *synthetic aperture radar* (SAR) devices are long-range sensors that operate by emitting radio waves from one or several antennas whilst moving over a stationary scene. Echo waveforms are processed to compute range images. Application areas are aerial surveying, topographic mapping, digital elevation modeling etc. [46]. SAR sensors are mostly employed on aircraft and satellites and are thus not related to mobile robotics applications.

2.4 3D Laser Range Data

This section describes some important aspects of three-dimensional range data, especially that obtained with laser range finders. While the algorithms presented throughout this thesis make no explicit assumptions about the sensor with which 3D point clouds have been acquired, they are benchmarked on simulated and real laser range data, therefore some of the aspects of this type of data merit discussion.

2.4.1 Point Clouds vs. Range Images

Since 3D range sensing has historically evolved from computer vision, it has naturally been treated in the context of images. Depending on the sensor, a 3D scan is represented by one or two images, a range image containing depth values for every pixel and an intensity image containing the associated surface brightness for every pixel. While intensity contains additional information that may be beneficial for several applications, for the scope of this thesis only the range values are of interest.

Range images constitute a fundamental data structure for various applications, such as segmentation, object detection and recognition etc. because they allow for the direct application of many of the powerful algorithms developed in the vision community over the past decades. For the majority of the previously described sensors, they constitute the ideal data structure, because the data periodically arrives in portions of 2D lattices with corresponding depth values, observed from one view point. For actuated 2D LRFs, however, they may or may not be ideal. Some of the setups described in Sec. 2.3.2 yield data from periodic sweeping motions in a confined spatial area. Such data can be represented well by range images using an appropriate projection to a viewing plane oriented relative to the LRF. Other actuation setups, such as that of Fig. 2.6(e) or the articulated robot arm used in this thesis allow for almost arbitrary scanning trajectories that may not conform to such a projection. As another example, imagine a mobile robot driving around with one or several vertically-mounted fixed 2D LRFs for the exploration of a building or an outdoor environment. The viewpoint trajectory exhibits no periodicity and the acquired data represents a continuous stream of incoming scanlines. Range images cannot accommodate this type of data.

As such, range images can be seen as a local data structure that captures depth information perceived from one view point in a compact form. Because depth is a function of pixel coordinates in the viewing plane, i.e. the local coordinates have a dependency $z = f(x, y)$, range images are a 2.5D representation. Apart from its compactness as a data structure, this property can be useful e.g. when reasoning about occlusions, see Sec. 4.3. In general, however, 3D range data is better represented by so-called *point clouds*, which are simply sets of three-dimensional points $\boldsymbol{p} = [x\ y\ z]^T$ that have no coordinate dependency. Point clouds are a global data structure that can hold any portion of 3D data, be it individual points, scanlines, scans taken from one view point or several such scans registered into one scene.

With the exception of Sec. 4.3, that uses a specific kind of range image introduced in the following section, the 3D data used throughout this thesis is stored and processed in point clouds because of their greater flexibility. Wherever relevant the view point from which individual points, scanlines or scans were recorded is retained with that data.

2.4.2 Omnidirectional Range Image

While 2.5D range images introduce a number of limitations for representing 3D information, they can be useful when assessing visibility for a set of points observed from one view point. Most researchers use image representations of homogeneous density, which can be easily constructed for simple horizontal and vertical scans (when both the field of view of the 2D LRF and the swept angle are less than 180°) using a pinhole camera projection [54]. However, in order to have *one* representation for all scan types, including rotational and omnidirectional scans, a spherical mapping can be used that is hereafter referred to as *omnidirectional range image* (ORI). It represents a 2.5D mapping from $[x\ y\ z]$ values to $[a\ e\ d(a,e)]$ values, where $a \in [-\pi, \pi[$ represents the azimuth, $e \in [-\pi/2, \pi/2[$ the elevation, and $d(a,e)$ the distance from the view point. For a given view point $\boldsymbol{v}_i$ and an ORI of size $[w, h]$ pixels, the coordinates a_i, e_i, and d_i of point $\boldsymbol{p}_i$ are computed as

$$\begin{aligned} a_i &= \left\lfloor \left(\frac{1}{2\pi} \operatorname{atan2}(p_{iy} - v_{iy}, p_{ix} - v_{ix}) + 0.5 \right) w \right\rfloor \\ e_i &= \left\lfloor \left(\frac{1}{\pi} \operatorname{atan2}(p_{zi} - v_{zi}, \sqrt{(p_{ix} - v_{ix})^2 + (p_{iy} - v_{iy})^2} + 0.5 \right) h \right\rfloor \\ d_i &= \|\boldsymbol{p}_i - \boldsymbol{v}_i\|_2 \,. \end{aligned} \tag{2.1}$$

Pixels should be initialized with a negative distance value indicating that no observation has been made. This accounts for the fact that LRFs may yield no reading on black, transparent, reflective or out-of-range surfaces and that the spacing of measured points may be larger than the spacing of pixels. The resolution should be chosen according to the angular spacing of the range sensor. Depending on the scanning setup, the coordinate system must be adapted to avoid issues with the singularity at $(x, y) = (0, 0)$. Fig. 2.11 shows the ORIs of a simulated and a real point cloud with inhomogeneous densities.

2.4.3 Simulation of Laser Range Data

For the design and assessment of algorithms that process range data, it is beneficial to have a simulation environment that replicates the data obtained from real sensors, in this case LRFs, sufficiently well. Such a simulation environment allows to evaluate algorithm performance and robustness in a way that would not be possible with real sensors. E.g. noise levels and scan densities can be varied gradually in simulation to examine their effect on algorithm performance. Obviously, simulation is not a replacement for real experiments,

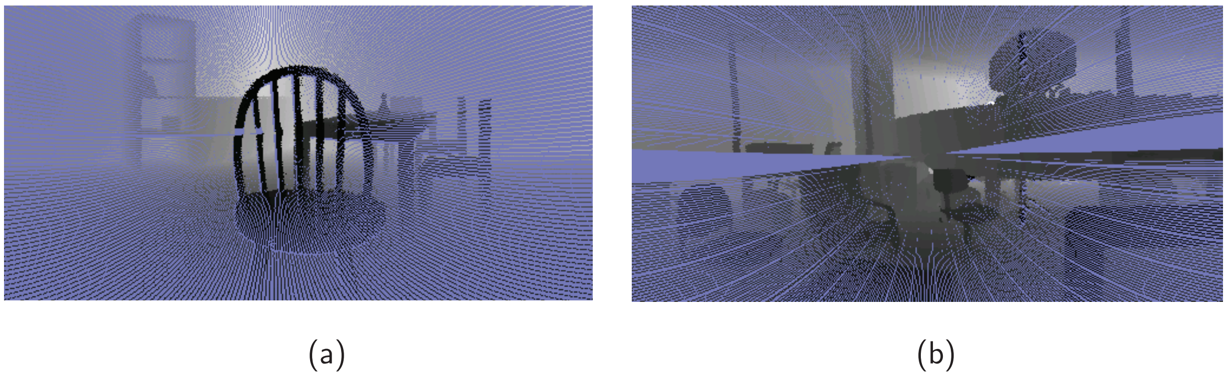


Figure 2.11: Omnidirectional range images (ORIs) contain depth information in a spherical mapping around the view point. (a) Cropped ORI of a simulated point cloud. (b) Cropped ORI of a real point cloud. Both point clouds were obtained with a rotational scan, which yields inhomogeneous density. Depth is grayscale-colored, blue pixels indicate that no observation was made.

but may provide algorithmic insights and save time and effort before new methods are tested on real data.

Fig. 2.12(a) shows an autonomous mobile robot used within the CoTeSys Multi-Robot-Lab as a household assistant. In the photograph, it utilizes a gripper as its left end effector. However, its articulated arm is identical to that introduced in the following section and can hold a compact 2D LRF to perform scans of its environment. For this robot, a detailed simulation model was created, that allows to accurately mimic the trajectories and 3D data obtained by this setup, see Fig. 2.12(b). Other simulation models include a 2D LRF on a simple tilt platform mounted on a Pioneer P3-DX robot, as shown in Fig. 2.12(c). However, because its low view point caused the acquired data to contain many occlusions, this model was only used in a series of initial studies.

The simulation of 2D laser range data is a well-understood concept that has been utilized extensively in 2D robotic mapping and navigation toolkits, such as Carmen [194] and Player/Stage [47]. Because the coherency of laser light makes for a very focused beam, and the rotating mirror has comparatively high positional accuracy, noise can be assumed to occur mainly along the laser rays [121]. The probabilistic distribution of the measured distance along the ray can be modeled as the superposition of a Gaussian around the true measurement value, a uniform distribution over the measurement range, and a peak at the maximum range [145]. The standard deviation of the Gaussian is obtained directly from the data sheet of the respective sensor, or it can be chosen manually to examine the effect of different noise levels. The peak at the maximum range results from the fact that most LRFs output maximum range values whenever a reading fails, e.g. on dark, transparent or reflective surfaces or when an object is too far away. As previously stated, this probabilistic sensor model has successfully been used for the simulation of laser data in 2D navigation and mapping scenarios. In these scenarios, the LRF is assumed to be mounted horizontally a few centimeters above ground on a mobile robot. To simulate the range readings, the true range value must be determined by intersecting the laser ray with objects in the environment, which is conceptually simple, as all calculations are carried

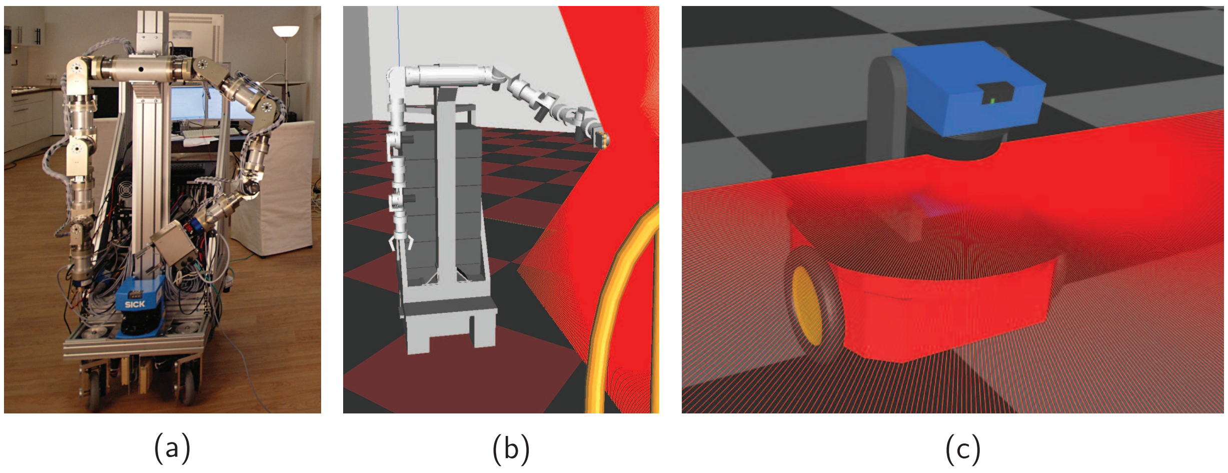


Figure 2.12: (a) One of the robots of the CoTeSys Multi-Robot Lab with a gripper as its left end effector. (b) The simulation model with a Hokuyo UTM-30LX laser range finder as its left end effector. (c) Simulation model of a Pioneer P3-DX robot with a SICK LMS-200 laser range finder on a tilt platform.

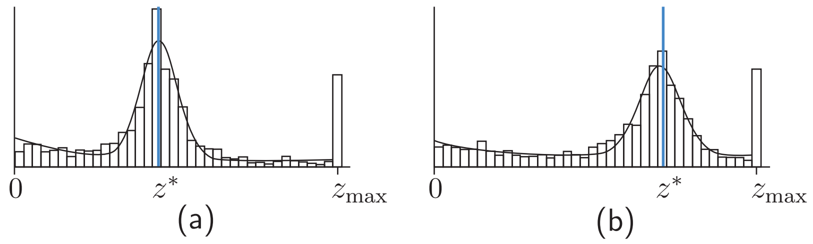


Figure 2.13: The probability of measuring the true distance with a laser range finder can be modeled as a superposition of several distributions [145]. (a)-(b) Histograms and approximated distributions for two different ranges. The distribution around the true distance z^* is Gaussian.

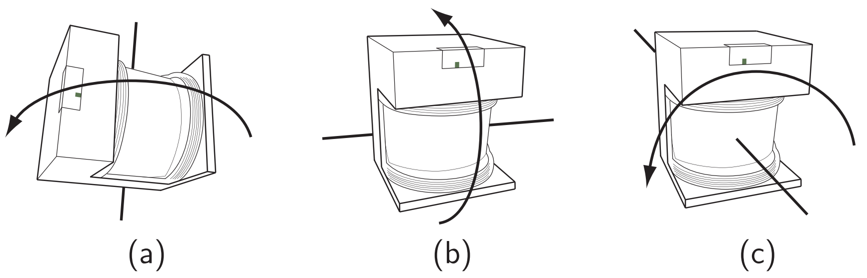


Figure 2.14: Three fundamental scan trajectories are commonly encountered in actuated 2D LRF scanning setups: (a) Horizontal scan (b) Vertical scan (c) Rotational scan

out in the two-dimensional horizontal plane. The actual range value is then calculated by distorting the true value according to the sensor distribution.

For the simulation of 3D laser range data, the same sensorial distribution along the laser ray can be used; the main challenge lies in handling the non-trivial intersection of laser rays with the three-dimensional environment. Several software packages offer the possibility of simulating a 2D LRF in a 3D dimensional world, e.g. Player/Stage in conjunction with Gazebo or commercial packages such as Webots [96] and Microsoft Robotics Developer Studio [192]. The simulation environment used in this thesis was built on top of an open source path planning library previously developed by the author [168, 188]. It already provided most of the necessary simulation capabilities for 3D environment geometry and kinematic chains of the robots, so there was no need to resort to the aforementioned solutions. The simulation environment was implemented in C++ and uses ray casting algorithms from the Bullet physics engine [181] to efficiently compute the ray intersections. The Coin3D library [189] was used for visualization. High density scans in complex environments such as the kitchen scenario shown in Fig. 2.15 were simulated effortlessly at 40Hz scanning frequency and higher on a standard desktop PC.

For both simulation and real experiments the following sensor parameters are defined:

- t_s: the total time of a 3D scan in seconds
- f_s: the scanning frequency in Hz, i.e. the rate at which scanlines are acquired
- α_r: the angular resolution in degrees, i.e. the spacing between the laser beams
- α_{FoV}: the field of view in degrees, i.e. the angle covered by the laser beams

Furthermore, three fundamental scan trajectories are defined, as illustrated in Fig. 2.14: The *horizontal scan* involves a yawing motion that causes the LRF to cover the environment with vertical scanlines from left to right or vice versa. The *vertical scan* involves a pitching motion of the actuator that causes the 2D LRF to gradually cover the environment with horizontal scanlines from top to bottom or vice versa. The *rotational scan* involves the rotation of the LRF about its axis-of-sight, measuring a cone-shaped profile. All three trajectories are commonly encountered in actuated 2D LRF scanning setups, compare Fig. 2.6. The horizontal and vertical scan essentially represent the same kind of sweeping motion in different orientations. However, due to the different resolution within scanlines

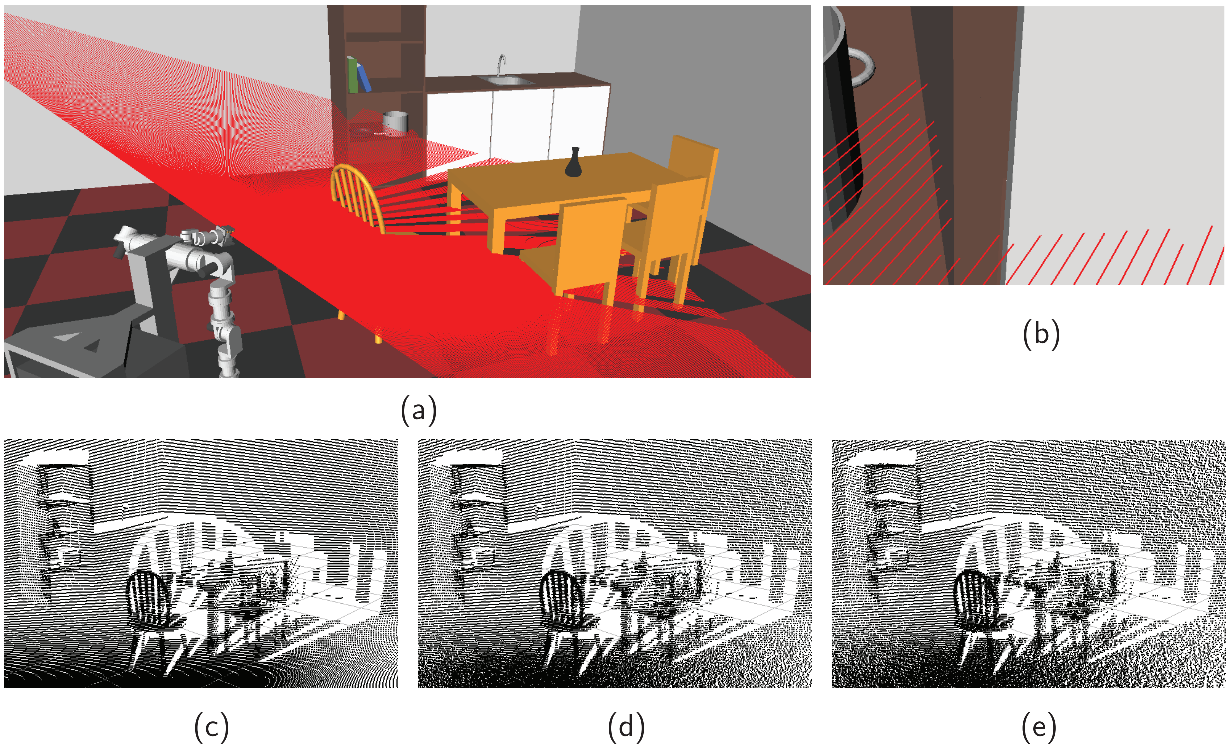


Figure 2.15: Simulation of a household robot in a kitchen environment: (a) The robot performs a vertical scan from top to bottom. (b) Gaussian noise is simulated along the laser rays. (c)-(e) The resulting point clouds with noise levels $\sigma = 0$mm (no noise), $\sigma = 15$mm, and $\sigma = 30$mm, respectively.

and between scanlines they can result in a significantly different amount of points covering the ground plane, the ceiling and the objects in a given environment. Interestingly, the rotational scan, which is most convenient from a mechanical perspective because it allows for continuous scanning, is also the one that yields data of least homogeneous density, with many points in the middle of the measurement cone and only few points on the outside.

Some of the capabilities of the simulator are illustrated in Figs. 2.15 and 2.16. Fig. 2.15(a) depicts a detailed simulation of a household robot acquiring a vertical scan of a kitchen environment. Fig. 2.15(b) shows a close-up of the Gaussian measurement noise simulated along the laser rays, and Figs. 2.15(c)-(e) show the resulting point cloud for different noise levels. Fig. 2.16 illustrates the output of the simulator for the same kitchen environment scanned with rotational scans of three different densities.

2.5 Hardware Setups Used in this Thesis

Most of the experimental data used in the following chapters has been captured with the two hardware systems shown in Fig. 2.17. The first and simpler setup uses the Autonomous City Explorer (ACE) [166] depicted in Fig. 2.17(a), a mobile robot normally used for outdoor navigation. It consists of a differential wheel platform by BlueBotics, Inc. and a number of hardware components mounted on top. 2D laser data from the bottom

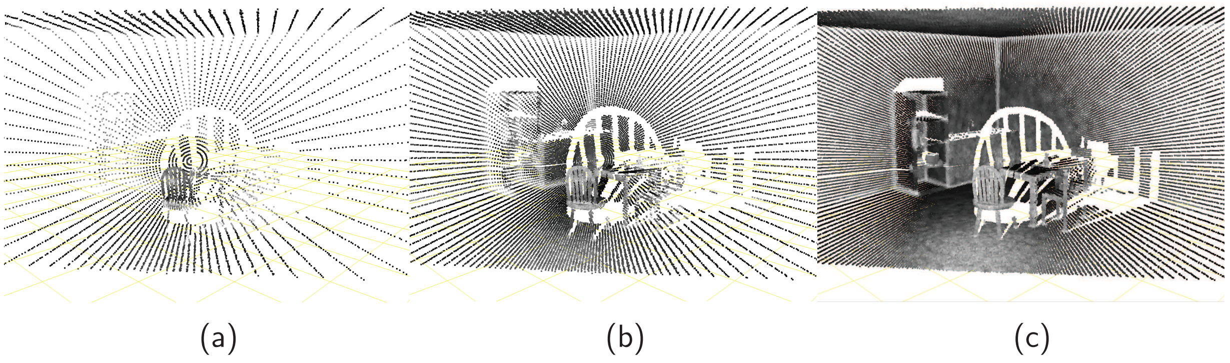


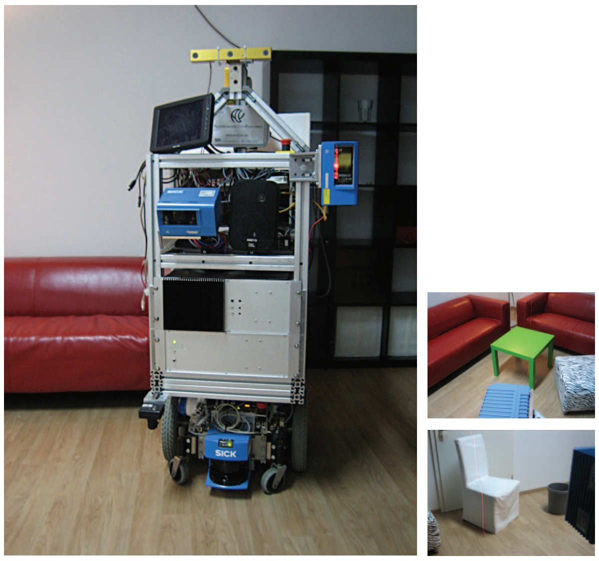
Figure 2.16: Three simulated rotational scans of different densities: (a) Low-density scan, t_s=4s, f_s=10Hz, α_r=1.0° (7,200 points). (b) Medium-density scan, t_s=4s, f_s=20Hz, α_r=0.5° (28,800 points). (c) High-density scan, t_s=4s, f_s=40Hz, α_r=0.25° (115,200 points). The last scan corresponds to the density obtained with a Hokuyo UTM-30LX. All three scans were simulated with σ=15mm and are grayscale-colored by curvature value, see Sec. 3.3.3.

SICK LMS-200 LRF is used for obstacle avoidance and mapping. In conjunction with the odometry data obtained from the wheel encoders, it serves as input to a localization module, which outputs continuous estimates of the robot pose, i.e. position and orientation in the horizontal plane. These are used to transform the laser data measured by the vertically mounted SICK LMS-400 LRF to a global coordinate frame. This scanning setup allows to capture horizontal scans as the robot turns on the spot.

The second robotic system shown in Fig. 2.17(b) consists of an articulated dual arm manipulator with seven DoF per arm [139]. A Hokuyo UTM-30LX is used as the left end effector and can be positioned and oriented in almost any 3D pose within the robot workspace. For all of the recorded data sets, only points within a field of view $\alpha_{FoV} = 180°$ were used. The robot is mounted in a fixed position at the ceiling and the end effector pose within the world is calculated solely from the forward kinematics of the linkage using the joint encoder values. With this estimate, each scanline from the LRF is transformed to a global coordinate frame. While the setup allows for almost arbitrary scanning trajectories within the kinematic and dynamic limits of the system, it was mainly used to examine the effects of the three previously explained fundamental scanning trajectories. As a future enhancement, the setup can directly be deployed on the mobile household robot shown in Fig. 2.12(a), which uses the same arms. For this scenario the end effector pose would have to be further transformed by the estimated robot pose.

2.6 Discussion

The sensing modalities that were presented in this chapter span a wide range of sensors of very different lateral and depth resolution, acquisition rates, and robustness to environment conditions. To summarize, passive methods such as *shape from shading* and *structure from motion* suffer from a number of drawbacks including high acquisition/computation time, dependence on texture and lighting conditions, unknown scaling and poor accuracy. *Stereo*


(a)

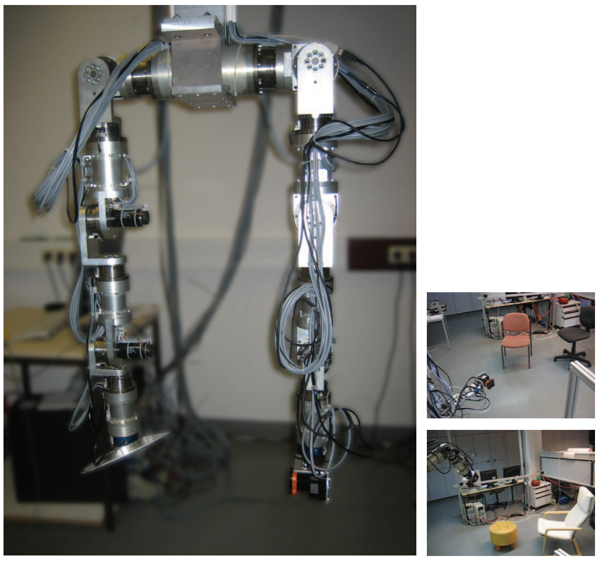

(b)

Figure 2.17: Two robots used to capture 3D data: (a) The *Autonomous City Explorer* (ACE) robot [166] with a vertically mounted SICK LMS-400. As the robot turns on the spot a horizontal scan is captured. (b) The 7 DoF dual arm manipulator [139] with a Hokuyo UTM-30LX as its left end effector. This setup allows for almost arbitrary scanning trajectories in the robot workspace.

vision remedies some of these aspects by use of two or more cameras that operate from a known, calibrated stereo baseline. Depth maps are obtained at decent frame rates; however, in comparison to other sensing modalities, stereo vision data is of inferior quality because of short range, poor and inhomogeneous depth resolution and dependence on object texture. In the family of active sensors, *acoustic sensors*, *structured light scanners*, and *desktop scanners* were ruled out as suitable sensors for mobile robots for reasons of poor accuracy, short range, poor portability, and/or dependence on illuminability of the environment. State-of-the-art *time-of-flight cameras* were identified as more promising sensors for their ability to acquire colored depth data with medium lateral and mediocre depth resolution at video frame rates; however, they suffer from high levels of random and systematic noise. As outlined, with improved resolution, ToF cameras can be expected to become increasingly relevant for mobile robotics in the future. As one of the most popular range sensors, the laser range finder provides high-resolution scans with good depth accuracy in medium acquisition times. Apart from high-performance but very expensive *3D laser range finders*, the most commonly encountered scanning setups are *actuated 2D laser range finders*, which were discussed in detail.

Different strategies and views on the acquisition of semantic environment models have been proposed and pursued in research. For the outlined reasons of accuracy and resolution, the sensing modality considered in simulation and experiment throughout the rest of this thesis are actuated 2D laser range finders. Existing works that use LRFs often process extremely dense point clouds on the order of millions of points, recorded and registered from several view points [93, 119]. This data is then processed to extract very detailed geometric models of objects in the environment. While this is possible for an initial offline learning phase, in which the robot is allowed to survey an environment uninterrupted, the acquisition of such detailed data is infeasible for everyday operation. A central argument put forth here, is that a robot that is to perform some useful task in a domestic setting must be able to repeatedly assess the state of its environment by performing a quick, relatively sparse 3D scan and interpreting it. In the case of actuated laser range finders, both the scan and the interpretation stages should take only seconds, so that the robot can make decisions in a time horizon useful for humans.

Obviously, even lower acquisition times can be provided by time-of-flight cameras and stereo vision. However, the bottom line of the overview presented in this chapter is that there currently is a trade-off between data accuracy (especially with respect to depth resolution) and acquisition time between the different sensing modalities. For the preference of accurate data, the sensing scenario considered in this thesis is therefore an actuated 2D LRF performing a quick sweep to obtain a relatively sparse point cloud. In the long run, it is expected that both sensing modalities will become equally favorable, that is, actuated laser range finders will provide shorter acquisition times and camera-based sensing will yield improved resolution.

3 Point Cloud Features

Flik, the misfit ant, climbing a dandelion
A Bug's Life, © 1998 Disney/PIXAR

Summary

This chapter describes the computation of scalar and vectorial quantities that characterize sampled surface points and point clouds as a whole. The most important of these so-called features is the surface normal vector. Numerous methods for its estimation in the presence of noise and varying point density are analyzed and compared in a performance study, followed by a description of higher-order point features that are used throughout the 3D range data literature. In addition, a number of quantities describing point sets called segment features are presented.

In the previous chapter various sensing modalities have been described that allow for the acquisition of 3D point clouds. Given such point clouds, an obvious question that arises is how to abstract and interpret this data. While humans automatically perceive spatial structure and even recognize objects when viewing a collection of sampled surface points, to a computer the recorded data is nothing but a list of x-, y-, and z-coordinates. As a natural consequence, researchers have long investigated methods to compute descriptive quantities that allow for geometric interpretation and abstraction. These quantities are commonly called *point cloud features*. This chapter describes both features for individual points, which involve characterizing the relationship between a specific point $\boldsymbol{p}_i$ and points in its local neighborhood, as well as features describing point clouds as a whole.

The chapter is structured as follows: Sec. 3.1 briefly describes work that is related to the presented material. Sec. 3.2 contains the central results of this chapter. It deals with a large variety of methods and aspects surrounding the estimation of normal vectors, which constitute one of the most important shape descriptors. In Sec. 3.3 various quantities that capture surface curvature are presented and discussed. More complex and descriptive features are described in Sec. 3.4. Sec. 3.5 presents quantities that characterize point sets, rather than individual points, and Sec. 3.6 completes the chapter with a brief discussion.

3.1 Related Work

To the best of the author's knowledge, no single article or book captures all of the methods presented in this chapter. A good amount of representative publications for each of the topics is referenced in the individual sections.

The problem of normal vector estimation examined in Sec. 3.2 has been covered in a plethora of overviews, the more comprehensive ones being McIvor and Valkenburg [94], Jin et al. [73], Dey et al. [34], and OuYang and Feng [107]. While all of these compare a number of methods, normal vector estimation approaches are neither rigorously categorized nor is their performance in terms of both quality and computation time considered, as is done here. Furthermore, methods such as the estimation with an Extended Kalman Filter, presented in Sec. 3.2.7, have to the author's knowledge not been carried out in any existing publication.

The computation of curvature values from point sets, covered in Sec. 3.3, has been the subject of several survey articles, most notably McIvor and Valkenburg [94] and Magid et al. [90]. The instability of this feature in the presence of noise, which is demonstrated in Sec. 3.3, has been noted in different contexts by other researchers, compare Trucco and Fisher [148], Powell et al. [111], and Frome et al. [42].

Both the advanced point features as well as the segment features, presented in Secs. 3.4 and 3.5, represent an aggregation of various methods to be found in the literature.

3.2 Normal Vectors

3D point clouds obtained from range sensors represent a noisy sampling of surfaces that exist in the real world. The explicit information about orientation and curvature of these surfaces is lost in the sampling process. It resides implicitly in the relationship between a sampled point and points in its neighborhood. Normal vector estimation seeks to restore this information for every sampled surface point by constructing a vector that is orthogonal to the tangent plane of that point. Normal vectors are so-called local descriptors. The majority of methods try to estimate a normal vector for every point in the point cloud. This is a reasonable procedure especially if the resulting normal vector is later on integrated into a feature space in which every data item corresponds to one point in a point cloud. An alternative not considered in this thesis is to group several points for the computation of one normal vector, as is done, e.g. for the calculation of *Big Delaunay Balls* [34].

3.2.1 Definitions and Notation

A point cloud, consisting of a set of n points $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n\}, \boldsymbol{p}_i \in \mathbb{R}^3$ shall be represented by an $n \times 3$ data matrix

$$\boldsymbol{P} = [\boldsymbol{p}_1 \ \cdots \ \boldsymbol{p}_n]^T,$$

where $\boldsymbol{p}_i = [p_{ix} \ p_{iy} \ p_{iz}]^T$ represents the 3D coordinates of the ith point. For every point $\boldsymbol{p}_i$ a normal vector $\boldsymbol{n}_i = [n_{ix} \ n_{iy} \ n_{iz}]^T$, is to be estimated from a set of k points in its neighborhood $\mathcal{Q}_i = \{\boldsymbol{q}_{i1}, \boldsymbol{q}_{i2}, ..., \boldsymbol{q}_{ik}\}, \boldsymbol{q}_{ij} \in P, \boldsymbol{q}_{ij} \neq \boldsymbol{p}_i$. Wherever necessary, normal vectors are normalized to unit length after the estimation, i.e. $\|\boldsymbol{n}\|_2 = 1$. The matrices

$$\boldsymbol{Q}_i = [\boldsymbol{q}_{i1} \ \cdots \ \boldsymbol{q}_{ik}]^T$$
$$\boldsymbol{Q}_i^+ = [\boldsymbol{p}_i \ \boldsymbol{q}_{i1} \ \cdots \ \boldsymbol{q}_{ik}]^T$$

describe the points in the vicinity of $\boldsymbol{p}_i$, where $\boldsymbol{Q}_i$ is the $k \times 3$ neighborhood matrix and $\boldsymbol{Q}_i^+$ is the $(k+1) \times 3$ augmented neighborhood matrix containing all neighbors plus the point $\boldsymbol{p}_i$ itself.

3.2.2 Nearest Neighbors

Identifying a suitable set of neighboring points is a non-trivial task. The most natural data structure that captures the notion of neighborhood is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which every vertex $\boldsymbol{v}_i \in \mathcal{V}$ corresponds to a point $\boldsymbol{p}_i \in \mathcal{P}$ and is connected to neighbors via edges $\boldsymbol{e}_{ij} \in \mathcal{E}$.

Throughout the literature, two types of graph are used for normal vector estimation:

- The k *nearest neighbor* (kNN) graph connects every vertex to its k nearest neighbors, where 'nearness' is measured by a Euclidean distance metric $\rho(\boldsymbol{v}_i, \boldsymbol{v}_j) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2$ for the scope of this thesis. The number of nearest neighbors is fixed and this may lead to overlapping triangles especially at the rim of a point cloud.

- The *Delaunay tessellation* (DT) graph connects simplices of points, such that the circumsphere of each resulting simplex does not contain any of the other points.

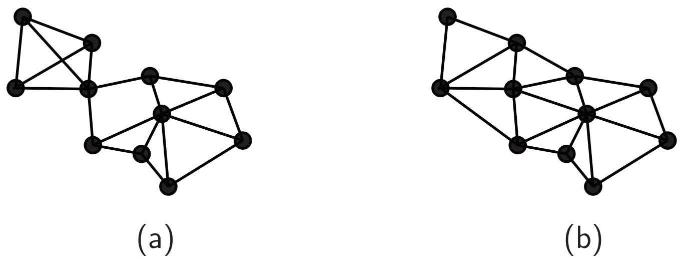


Figure 3.1: Two types of graph can be used for normal vector estimation: (a) The k nearest neighbor (kNN) graph connects every vertex to its k nearest neighbors. (b) The Delaunay tessellation (DT) graph connects neighbors to a regular mesh of simplices.

In 3D this yields a tetrahedral mesh, in 2D a triangular mesh. Each vertex has a variable number of neighbors and there are never any overlapping triangles.

Fig. 3.1 illustrates the two types of graph for a 2D point set. The methods for normal vector estimation examined in the following are benchmarked with both types of graph.

3.2.3 Estimation with Optimization-based Methods

For the majority of existing methods, normal vector estimation can be cast into an optimization problem of the form

$$\min_{\boldsymbol{n}_i} J(\boldsymbol{p}_i, \boldsymbol{Q}_i, \boldsymbol{n}_i),$$

where $J(\boldsymbol{p}_i, \boldsymbol{Q}_i, \boldsymbol{n}_i)$ is a cost functional penalizing certain criteria, e.g. the distance of points to a local plane or the angle between the tangent vectors and the normal vector. Figs. 3.2(a) and (b) visualize these two approaches. As a matter of fact, many seemingly very different methods differ only in the cost functional they optimize. Furthermore, the computational performance depends to a large degree on the optimization procedure that is used. When J can be stated as a linear expression in matrix-vector notation, the minimizer can be found directly as the result of a *singular value decomposition* (SVD) or a *principal component analysis* (PCA), see Appendix A.2. In case of a nonlinear dependence for which an analytic solution becomes infeasible, gradient techniques must be employed. All optimization-based methods examined in the following pose a convex linear optimization problem, so that gradient techniques are not relevant.

The optimization-based methods can be divided into linear and quadratic fitting approaches. In the following, three linear and two quadratic approaches are considered. It is shown that many existing methods belong to these basic categories or to variations of them.

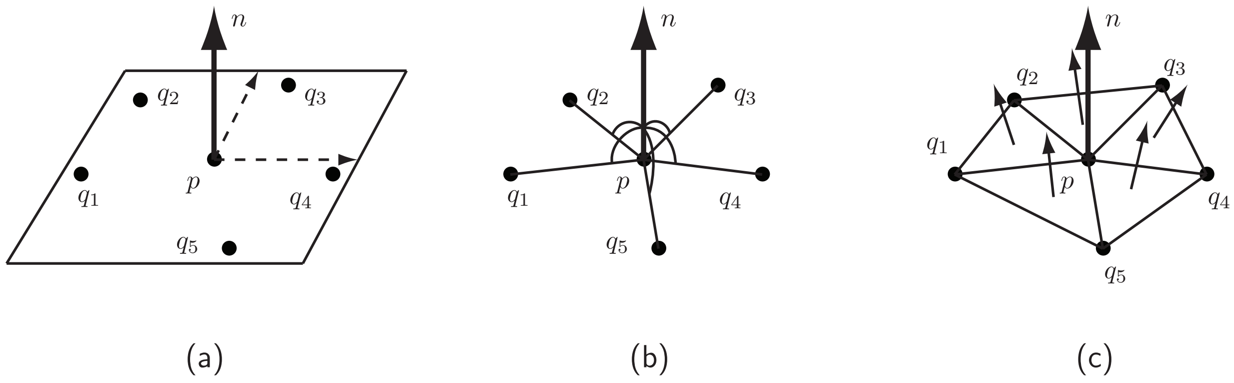


Figure 3.2: Different approaches for estimating normal vectors: (a) A plane is fitted to p and its neighbors. (b) The angle between the normal vector and the tangent vectors is maximized. (c) The normal vectors of triangles formed with pairs of neighbors are averaged.

PlaneSVD

A classical method [58, 65] is to fit a local plane $S_i = n_{ix}x + n_{iy}y + n_{iz}z + d$ to the points in $\boldsymbol{Q}_i^+$, i.e. solve

$$\min_{\boldsymbol{b}_i} \left\| \begin{bmatrix} \boldsymbol{Q}_i^+ & \mathbf{1}_{k+1} \end{bmatrix} \boldsymbol{b}_i \right\|_2^2, \tag{3.1}$$

where $\mathbf{1}_m$ represents an $m \times 1$ vector of ones and $\boldsymbol{b}_i = [\boldsymbol{n}_i^T \; d]^T$. Just as any of the following methods, this optimization problem can be solved by computing the SVD ($\boldsymbol{USV}^T$) of $\boldsymbol{Q}_i^+$. The minimizer $\hat{\boldsymbol{b}}_i$ is the vector in $\boldsymbol{V}$ that corresponds to the smallest singular value in S, see Appendix A.2, and thus the normal vector $\boldsymbol{n}_i$ is obtained directly from the SVD.

PlanePCA

Instead of minimizing the fitting error, one can minimize the variance by removing the empirical mean from the data matrix $\boldsymbol{Q}_i^+$ and then performing a SVD on the modified data matrix [62]:

$$\min_{\boldsymbol{n}_i} \left\| \left[\boldsymbol{Q}_i^+ - \bar{\boldsymbol{Q}}_i^+ \right] \boldsymbol{n}_i \right\|_2^2, \tag{3.2}$$

where $\bar{\boldsymbol{Q}}_i^+ = \mathbf{1}_{k+1} \bar{\boldsymbol{q}}_i^{+T}$ is a matrix containing the mean vector $\bar{\boldsymbol{q}}_i^+ = \frac{1}{k+1}(\boldsymbol{p}_i + \sum_{j=1}^k \boldsymbol{q}_{ij})$ in every row. This is equivalent to performing a PCA of the original matrix $\boldsymbol{Q}_i^+$ and choosing the principal component with the smallest covariance, which is why this method is dubbed *PlanePCA*. It should be noted that the subtraction of the mean is not a heuristic, but leads to the optimal solution in terms of mean square error [100].

On a side note, a PCA of the points is equivalent to the *maximum likelihood* estimation of a local plane, as described in [151] and [77]. If the points $\boldsymbol{q}_{ij}^+$ in $\boldsymbol{Q}_i^+$ are regarded as measurements of a plane (defined by $\boldsymbol{n}_i$) perturbed by independent Gaussian noise $N(0, \sigma)$

around their true position $\hat{\boldsymbol{q}}_{ij}^{+} = [\hat{q}_{ijx}^{+}\ \hat{q}_{ijy}^{+}\ \hat{q}_{ijz}^{+}]^T$ in the x, y, and z direction, then maximizing the likelihood

$$P(\boldsymbol{Q}_i^{+}|\hat{\boldsymbol{Q}}_i^{+}) = \frac{1}{2\pi\sigma^2}\prod_{j=1}^{k+1}\exp\left(-\frac{\|\boldsymbol{q}_{ij}^{+} - \hat{\boldsymbol{q}}_{ij}^{+}\|_2^2}{2\sigma^2}\right) \tag{3.3}$$

of the measurements given their true positions (defined by the plane fit) is equivalent to minimizing

$$J = \sum_{\boldsymbol{q}_{ij}\in\boldsymbol{Q}_i^{+}} \frac{\langle \boldsymbol{n}_i, \boldsymbol{q}_{ij}^{+}\rangle^2}{\|\boldsymbol{n}_i\|_2^2}, \quad \text{s.t. } \|\boldsymbol{n}_i\|_2 = 1 \tag{3.4}$$

as is shown in [77]. In [151] $\boldsymbol{n}_i$ is then obtained as the eigenvector corresponding to the smallest eigenvalue of

$$G = \frac{1}{k+1}\sum_{j=1}^{k+1}(\boldsymbol{q}_{ij}^{+} - \bar{\boldsymbol{q}}_{ij}^{+})(\boldsymbol{q}_{ij}^{+} - \bar{\boldsymbol{q}}_{ij}^{+})^T, \tag{3.5}$$

which is, however, equivalent to the minimizer found by an SVD applied to (3.2), see Appendix A.2. Thus, a PCA performed on the neighborhood $\boldsymbol{Q}_i^{+}$ is equivalent to a maximum likelihood estimation of a local plane fitted in $\boldsymbol{Q}_i^{+}$.

VectorSVD

An equally straight-forward alternative to fitting a local plane S into $\boldsymbol{Q}_i^{+}$ is to maximize the angle (minimize the inner product) between the tangent vectors from $\boldsymbol{p}_i$ to $\boldsymbol{q}_{ij}$ and the normal vector $\boldsymbol{n}_i$. A moment's thought should convince the reader that this is equivalent to fitting a local plane that is fixed in $\boldsymbol{p}_i$, or – in other words – shifting the origin to $\boldsymbol{p}_i$ and then fitting a plane $S_i = \boldsymbol{n}_{ix}x + \boldsymbol{n}_{iy}y + \boldsymbol{n}_{iz}z$ to the points in $\boldsymbol{Q}_i$:

$$\min_{\boldsymbol{n}_i} \left\| \left[\boldsymbol{Q}_i - \boldsymbol{1}_k\boldsymbol{p}_i^T\right] \boldsymbol{n}_i\right\|_2^2 \tag{3.6}$$

In analogy to the *PlanePCA* method it is tempting to define a *VectorPCA* method, that minimizes the variance of the inner product defined in *VectorSVD*. In fact this is the approach presented in [50]. However, the results produced by this method are almost identical to the ones obtained from *PlanePCA*, because their data matrices differ only by $\boldsymbol{p}_i$. When the centroid of $\boldsymbol{Q}_i$ is equal to $\boldsymbol{p}_i$ the two methods are identical. *VectorPCA* is therefore not considered in the benchmark.

	#	Method	$J(\boldsymbol{p}_i, \boldsymbol{Q}_i, \boldsymbol{n}_i)$	Matrix Size
LINEAR	1	*PlaneSVD*	$\left\Vert \left[\boldsymbol{Q}_i^+, \boldsymbol{1}_{k+1}\right] \left[\boldsymbol{n}_i^T \; d\right]^T \right\Vert_2$	$(k+1) \times 4$
	2	*PlanePCA*	$\left\Vert \left[\boldsymbol{Q}_i^+ - \boldsymbol{1}_{k+1}\bar{\boldsymbol{q}}_i^{+T}\right] \boldsymbol{n}_i \right\Vert_2$	$(k+1) \times 3$
	3	*VectorSVD*	$\left\Vert \left[\boldsymbol{Q}_i - \boldsymbol{1}_k \boldsymbol{p}_i^T\right] \boldsymbol{n}_i \right\Vert_2$	$k \times 3$
QUADR.	4	*QuadSVD*	$\Vert [\boldsymbol{R}_i \; \boldsymbol{1}_k] \, \boldsymbol{c}_i \Vert_2$	$k \times 10$
	5	*QuadTransSVD*	$\Vert [\boldsymbol{R}_i' \; \boldsymbol{1}_k] \, \boldsymbol{c}_i - \boldsymbol{q}_{iz}' \Vert_2$	$k \times 6$

Table 3.1: Cost functionals and matrix sizes of the optimization-based methods

QuadSVD

Some methods approximate not only the orientation of the tangent plane but also the curvature [98, 162] by fitting a quadric surface of the form $S = c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5xz + c_6yz + c_7x + c_8y + c_9z + c_{10}$ to the set of neighbors. The optimization problem then becomes

$$\min_{\boldsymbol{c}_i} \Vert [\boldsymbol{R}_i, \boldsymbol{1}_k] \, \boldsymbol{c}_i \Vert_2^2 \,, \tag{3.7}$$

where $\boldsymbol{R}_i$ contains the linear and quadratic data items $[q_{ix}^2 \; q_{iy}^2 \; q_{iz}^2 \; q_{ix}q_{iy} \; q_{ix}q_{iz} \; q_{iy}q_{iz} \; q_{ix} \; q_{iy} \; q_{iz}]$ in each row and $\boldsymbol{c}_i = [c_{i1} \; \cdots \; c_{i10}]^T$ is the coefficient vector, which is obtained by an SVD of $\boldsymbol{R}_i$. $\boldsymbol{n}_i$ is then obtained directly from the coefficients, see [107] for details.

A major drawback of this method is that, depending on the neighborhood graph, $\boldsymbol{Q}_i$ may not contain ten or more neighbors, which makes fitting S to $\boldsymbol{Q}_i$ an ill-posed problem. The next method represents a partial remedy for this.

QuadTransSVD

A common procedure is to transform the points to a coordinate system in which some dependence $z = f(x, y)$ can be exploited to reduce the number of coefficients to be estimated. In [150] a suitable coordinate system is identified by fitting a local plane to $\boldsymbol{Q}_i$ (with *VectorSVD*) and using the resulting normal vector as the new z-axis of a Cartesian coordinate system with origin $\boldsymbol{p}_i$. After transforming $\boldsymbol{Q}_i$ to this system, the quadric surface $S = c_1x^2 + c_2y^2 + c_3xy + c_4x + c_5y + c_6 - z$ is fitted to the set of transformed points $\boldsymbol{Q}_i'$. This is accomplished by solving

$$\min_{\boldsymbol{c}_i} \Vert [\boldsymbol{R}_i', \boldsymbol{1}_k] \, \boldsymbol{c}_i - \boldsymbol{q}_{iz}' \Vert_2 \,, \tag{3.8}$$

where $\boldsymbol{R}'_i$ contains the linear and quadratic data items $[q'^2_{ix}\ q'^2_{iy}\ q'_{ix}q'_{iy}\ q'_{ix}\ q'_{iy}]$ in each row, $\boldsymbol{c}_i = [c_{i1}, \cdots, c_{i6}]^T$ is the coefficient vector and $\boldsymbol{q}'_{iz}$ is the vector of transformed z-coordinates. The solution is again found via the SVD ($\boldsymbol{USV}^T$) of $\boldsymbol{R}'_i$ by computing

$$\boldsymbol{c}_i = \sum_{j=1}^{r} \frac{\langle \boldsymbol{u}_j, \boldsymbol{q}'_{iz} \rangle}{s_j} \boldsymbol{v}_j,$$

where $r = \operatorname{rank}(\boldsymbol{R}'_i) = 3$ and s_j is the jth singular value of $\boldsymbol{S}$, see [49]. With the coefficient vector $\boldsymbol{c}_i$ at hand, the normal vector in the transformed coordinate system is computed from the cross product

$$\boldsymbol{n}'_i = [1\ 0\ c_{i4}]^T \times [0\ 1\ c_{i5}]^T$$

and $\boldsymbol{n}_i$ is finally obtained by transforming $\boldsymbol{n}'_i$ back to the original coordinate system.

Tab. 3.1 summarizes the cost functionals and matrix sizes of the optimization-based methods.

3.2.4 Estimation with Averaging Methods

An attractive alternative to finding the normal vector by optimization is to calculate it as the weighted average of the normal vectors of the triangles formed by p_i and pairs of its neighbors. Technically, the average of a set of points can also be viewed as the solution to an optimization problem, namely as the point that minimizes the distance to all points in the set. However, averaging methods are not motivated by such an optimization, but rather by the visually intuitive idea of combining the normal vectors of neighboring triangles, see Fig. 3.2(c). They are consequently not characterized by a cost functional and are treated as a separate family of methods in this study.

Given k neighbors, in theory there are $\binom{k}{2} = \frac{1}{2}k(k-1)$ neighboring triangles. However, usually one only steps through k pairs of consecutively stored neighbors $\boldsymbol{q}_{i,j}$ and $\boldsymbol{q}_{i,j+1}$, assuming that the neighbors exhibit some sensible order. The DT graph provides an angular ordering in a local plane, but as shall be seen, even the kNN graph, where no specific ordering is obeyed, yields good results if the number of neighbors is sufficient. The basic averaging method proposed in [51] is

$$\boldsymbol{n}_i = \frac{1}{k} \sum_{j=1}^{k} w_j \frac{\left([\boldsymbol{q}_{ij} - \boldsymbol{p}_i] \times [\boldsymbol{q}_{i(j+1)} - \boldsymbol{p}_i]\right)}{\left\|[\boldsymbol{q}_{ij} - \boldsymbol{p}_i] \times [\boldsymbol{q}_{i(j+1)} - \boldsymbol{p}_i]\right\|_2}, \tag{3.9}$$

where $w_j = 1$. For notational simplicity assume here that the last-plus-one index maps onto the first neighbor again, i.e. $k + 1 := 1$. There exist numerous variations of this scheme, which are all based on using a weighted average ($w_j \neq 1$) instead of the basic average and include among others *angle-weighted*, *area-weighted*, *centroid-weighted* and *gravitational-weighted* methods. A good summary of these methods can be found in [73]. Due to the similar performance of many variants only two representative methods are considered in the following.

#	Method	Weighting Factor
6	*AreaWeighted*	$\frac{1}{2}\|[\boldsymbol{q}_{ij} - \boldsymbol{p}_i] \times [\boldsymbol{q}_{i(j+1)} - \boldsymbol{p}_i]\|_2$
7	*AngleWeighted*	$\cos^{-1}\left(\frac{\langle \boldsymbol{q}_{ij}-\boldsymbol{p}_i, \boldsymbol{q}_{i(j+1)}-\boldsymbol{p}_i\rangle}{\|\boldsymbol{q}_{ij}-\boldsymbol{p}_i\|_2 \|\boldsymbol{q}_{i(j+1)}-\boldsymbol{p}_i\|_2}\right)$

Table 3.2: Weighting factors of the averaging methods

AreaWeighted

This method represents the most straightforward algorithm of the averaging methods. The normal vector of each triangle is weighted by the magnitude of its area, which is in turn equal to half the magnitude of the normal vector, thus

$$w_j = \frac{1}{2}\|[\boldsymbol{q}_{ij} - \boldsymbol{p}_i] \times [\boldsymbol{q}_{i(j+1)} - \boldsymbol{p}_i]\|_2, \tag{3.10}$$

which cancels the normalization factor in (3.9). Therefore, the normal vector is simply computed as half the average of the (unnormalized) cross products of the adjacent triangles.

AngleWeighted

Some researchers use the angle between two consecutive tangent vectors instead of the triangle area [73]. The weight is then expressed as

$$w_j = \cos^{-1}\left(\frac{\langle \boldsymbol{q}_{ij}-\boldsymbol{p}_i, \boldsymbol{q}_{i(j+1)}-\boldsymbol{p}_i\rangle}{\|\boldsymbol{q}_{ij}-\boldsymbol{p}_i\|_2 \|\boldsymbol{q}_{i(j+1)}-\boldsymbol{p}_i\|_2}\right), \tag{3.11}$$

which penalizes acute triangle ‘shards’.

Tab. 3.2 lists the weighting functions of the averaging methods.

3.2.5 Computational Complexity

In terms of computational complexity, the five optimization-based methods differ mostly in the size of the matrix that is to be decomposed by the SVD. The SVD has $O(d^3)$, where d is number of columns. Thus all linear methods should perform approximately the same, while *QuadSVD* should be substantially slower ($d = 10$). *QuadTransSVD* has $d = 6$; however, there is an extra SVD involved for finding the transformation plane and also the overhead of the two transformations contributes to the overall cost. For the two averaging methods, no optimization is needed, thus they should perform substantially faster than the previous methods.

3.2.6 Experimental Comparison

The seven presented methods were benchmarked on a simulated and a real point cloud to determine their performance in terms of runtime and quality.

Performance Index

In order to assess the quality of the normal estimation, the quality measure γ of an estimated normal vector $\boldsymbol{n}_i$ is defined as the absolute value of its inner product with the ground truth normal vector $\hat{\boldsymbol{n}}_i$:

$$\gamma_i := \gamma(\boldsymbol{n}_i, \hat{\boldsymbol{n}}_i) = |\langle \boldsymbol{n}_i, \hat{\boldsymbol{n}}_i \rangle| \tag{3.12}$$

In the benchmark, a central question to be answered is whether the increased computational effort of some methods is justified by a higher overall quality $\gamma = \frac{1}{n}\sum_{i=1}^{n} \gamma_i$. To this end, a *performance index*

$$\pi(\hat{t}) = \frac{1}{2}\gamma + \frac{1}{2}\left(\frac{\hat{t}}{\bar{t}}\right), \tag{3.13}$$

is defined, where $\bar{t}$ is the average computation time per normal and $\hat{t}$ is the desired computation time per normal. The rationale behind this is that while quality is naturally normalized by comparison with the ground truth, there exists no such normalization for computation time. The dilemma is overcome by letting the user define a reasonable lower bound $\hat{t}$, e.g. $\hat{t} = 1\mu$s per normal vector. The ideal algorithm would yield $\gamma = 1$ in $t = n \cdot \hat{t}$ total time and thus receive a perfect performance index of $\pi(\hat{t}) = 1$. Of course, this performance index may be varied to include different weighting of the two criteria quality and time or a nonlinear penalty of higher computation times. However, for this comparison the simple index with equal weighting from (3.13) shall suffice to produce a ranking of methods.

Scenario I: Simulated Point Cloud

The first scenario of the benchmark is a simulated noisy point cloud covering the surface of a box and a cylinder shown in Fig. 3.3. Created for an early study presented in [169], the data was generated directly in MATLAB without the simulation environment introduced in Sec. 2.4.3. Notwithstanding, noise and laser rays were simulated using exactly the same simulation principles. All methods were benchmarked with different values of depth noise σ and neighborhood size k. The ground truth normal vectors in this scenario were obtained directly from the analytical description of the simulated surfaces and are visualized in Fig. 3.3.

Scenario II: Real Point Cloud

The second benchmark scenario consists of a point cloud obtained from a SICK LMS400 laser scanner mounted at a 45° angle with respect to the horizontal plane on the ACE robot, see Sec. 2.5. To be able to reconstruct the ground truth normal vectors, three simple objects (two boxes and one cylindrical recycling bin) were placed in front of the robot, see Fig. 3.4 (top), and their exact position and orientation in the scene were measured. The resulting point cloud can be seen in Fig. 3.4 (bottom).

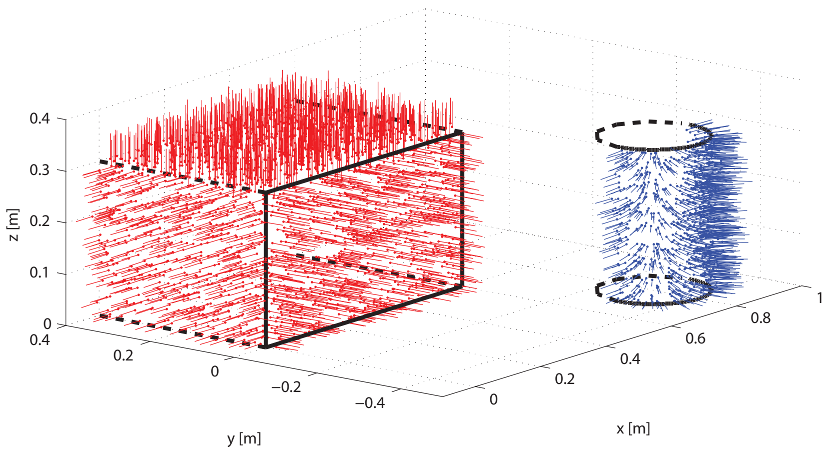


Figure 3.3: Simulated noisy data and ground truth normal vectors of a box and a cylinder.

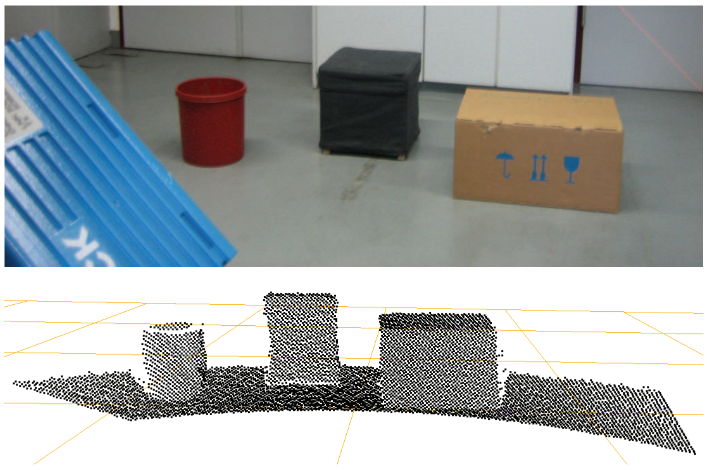


Figure 3.4: A real point cloud of simple objects was recorded for the benchmark. Top: Scan of the objects. Bottom: Resulting point cloud.

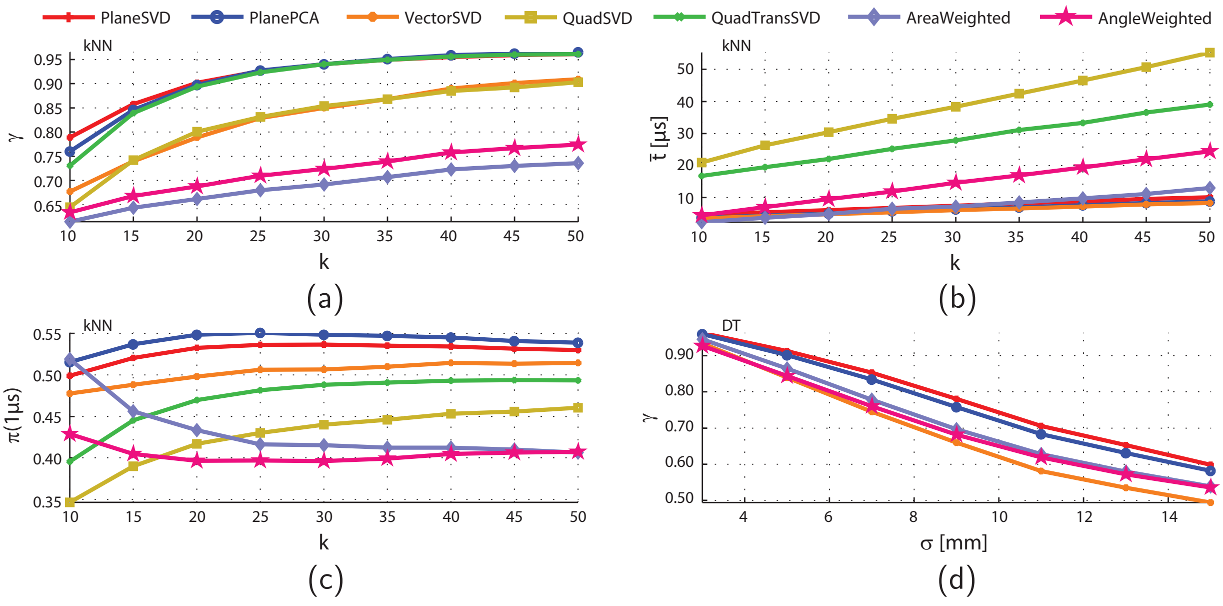


Figure 3.5: Results for Scenario I (simulated data): (a) Normal vector quality increases with growing neighborhood size k. (b) Runtime increases linearly with growing neighbor hood size k. (c) The performance index $\pi(1\mu s)$ produces a discriminative ranking of the methods. (d) The Delaunay tessellation graph connects only local neighbors, therefore normal vector quality deteriorates with growing noise levels σ.

Implementation Details

All algorithms were implemented in C and benchmarked on a Linux desktop computer with 3.0GHz and 2GB RAM. Nearest neighbors were computed using the ANN library [196], see Appendix A.1, and Delaunay tessellations were generated with the Qhull package [207]. The 3D Delaunay tessellation is cumbersome because it yields a tetrahedral mesh which must be simplified to a triangular mesh aligned with the true surface. For the benchmark, the triangular surface mesh was obtained by projecting the neighborhood $\boldsymbol{Q}_i^+$ onto a local plane and calculating the 2D Delaunay tessellation in this plane. Singular value decompositions and matrix operations were carried out with the GNU Scientific Library in conjunction with CBLAS [176].

Results

The results for Scenario I can be seen in Fig. 3.5. The simulated dataset consists of 14,200 points. For the kNN benchmark shown in Figs. 3.5(a)-(c), the neighborhood size was varied in increments of 5 between $k = 10$ and $k = 50$, while σ, the measurement noise standard deviation, was kept constant at $\sigma = 15$mm. The overall quality γ of all methods depicted in Fig. 3.5(a) is rather poor because of the high noise level σ. The presence of such noise clearly separates the methods from each other: *PlaneSVD*, *PlanePCA*, and *QuadTransSVD* take the lead with a strong increase in quality starting at medium-sized neighborhoods. The two averaging methods finish last with only slow improvement, while *QuadSVD* and *VectorSVD* show mediocre performance. The computation times (per normal) in Fig. 3.5(b) hold no surprises except that the linear optimization-based methods

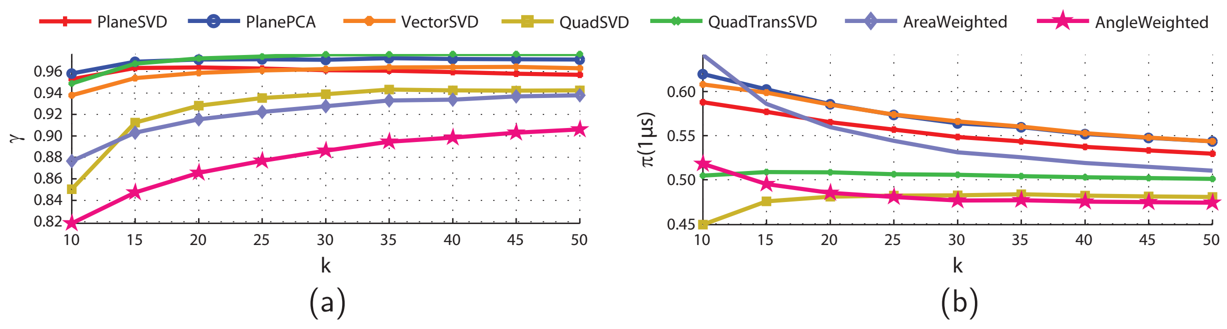


Figure 3.6: Results for Scenario II (real data): (a) Due to lower noise than in the simulation, higher quality is attained. (b) The performance index yields largely the same results as in the simulation.

Method:	1	2	3	6	7
γ:	0.9413	0.9436	0.9328	0.9180	0.8840
$\bar{t}$ [μs]	10.30	8.00	8.44	4.44	8.61

Table 3.3: Results for Scenario II (real data): Quality and computation time on DT Graph

outperform the averaging methods already for medium-sized neighborhoods. Fig. 3.5(c) shows the resulting performance index $\pi(1\mu s)$, meaning that the ideal algorithm should spend one microsecond for the computation of each normal. It highlights the strength of *PlanePCA* and *PlaneSVD*, which receive high scores for both quality and running time on a kNN graph. Fig. 3.5(d) shows the benchmark of the five applicable methods (all except the quadratic ones) for the DT graph. The variation of quality with the noise level clearly shows that all methods suffer from the locality of the DT graph. In all benchmark scenarios, the number of neighbors in the DT graph never exceeded ten, whereas the kNN graph could compensate noise for neighborhoods of size ten and above.

Scenario II contains the real range scan consisting of $7,346$ points and with a much smaller noise level σ. Fig. 3.6(a) shows that with smaller noise, *QuadTransSVD* performs even slightly better than *PlanePCA* for sufficiently large neighborhoods and the two averaging methods show constant improvement of γ with increasing neighborhood size, albeit still at a poor performance level. The ranking of running times is the exact same as for the simulated data and is thus omitted; absolute running times were halved, which corresponds to the fact that there are half the number of points in the dataset. The performance index $\pi(1\mu s)$ in Fig. 3.6(b) shows that *PlanePCA* still yields the best quality and speed for medium-sized neighborhoods for the real range data, this time sharing the lead with *VectorSVD*. Since the noise was constant (but unknown), the DT graph benchmark yielded only one set of values shown in Tab. 3.3. The running times and qualities show that with low noise the *AreaWeighted* method becomes a favorable alternative on a DT graph.

Discussion

The results demonstrate that *PlanePCA* provides the best quality and speed on a kNN graph for medium-sized and larger neighborhoods. Throughout the following chapters it

Algorithm 3.1: PlanePCA

```
   Input: Neighborhood $\boldsymbol{Q}_i^+$
   Output: Normal vector $\boldsymbol{n}_i$
1  $\widetilde{\boldsymbol{Q}}_i = \boldsymbol{Q}_i^+ - \frac{1}{k+1}\mathbf{1}_{(k+1),(k+1)}\boldsymbol{Q}_i^+$;
2  $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \mathrm{SVD}(\widetilde{\boldsymbol{Q}}_i)$;
3  $\boldsymbol{n}_i = \boldsymbol{v}_3$;
4  return $\boldsymbol{n}_i$;
```

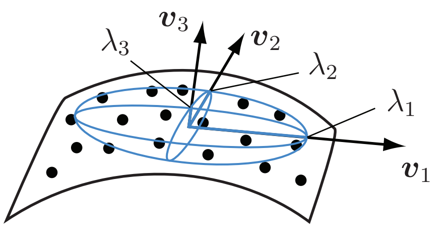


Figure 3.7: The principal component analysis yields an orthonormal basis that explains data variance. For sampled 2D surfaces most of the variance occurs along the actual surface, which is why the first two principal components $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ yield a good approximation of the tangent plane. Consequently, the normal vector is obtained as the third principal component $\boldsymbol{v}_3$, with variance λ_3.

is therefore used for normal vector estimation, using nearest neighbors retrieved from a kd-tree of the points. A pseudocode description of *PlanePCA* is provided in Alg. 3.1, where $\boldsymbol{v}_3$ denotes the singular vector with the smallest singular value. On a Delaunay tessellation graph, the *AreaWeighted* method is a favorable alternative; however, the construction of a triangular mesh may be considerably more involved than the retrieval of nearest neighbors.

In the presence of strong noise, least squares plane fits, such as the ones obtained by the linear methods, may yield poor normal vector estimates. Several researchers report improved results using a weighted covariance matrix and higher-order fits [2, 3, 93, 151]. Other alternatives include RANSAC-based fits [117] and its derivatives, such as MLESAC [146]. These methods come, however, at an increased computational cost.

The principal component analysis performed by *PlanePCA* also has a visually intuitive meaning, that shall be briefly explained here. Fig. 3.7 illustrates the principal components for an exemplary point cloud. They constitute an orthonormal coordinate frame for the points in which the largest variance of the data occurs along the first principal component, the second largest variance along the second principal component (orthogonal to the first) and so on. Intuitively it is clear, that on a sampled 2D surface in 3D space most of the data variance occurs in the tangent plane, which is spanned by the first two principal components. The third principal component represents a vector that is orthogonal to the first two and consequently represents the optimal vector (with respect to data variance) that is normal to the tangent plane. The singular values obtained as a result of the SVD are proportional (by a factor of $\sqrt{n}$) to the standard deviation along each component, see Appendix A.3.

3.2.7 Estimation with an Extended Kalman Filter

An interesting alternative to the afore-presented methods is the estimation of normal vectors with an *Extended Kalman Filter* (EKF). It does not offer computational advantages over *PlanePCA* or the like, but provides a measure for the uncertainty of the estimation taking into account the sensor's noise level σ. While estimation uncertainty is not relevant in the following chapters, it can be a highly informative quantity for applications requiring a measure of the 'goodness of estimate' for each normal vector and is therefore briefly covered.

The EKF is a nonlinear extension of the well-known linear Kalman filter [76]. Basically, each point in the neighborhood $\boldsymbol{Q}_i^+$ is regarded as a new observation, obtained with a sensor that yields data perturbed by Gaussian noise. Observations are incorporated one by one to compute the optimal estimate of the state of a dynamic system, which in this case is static and corresponds simply to the parameters of the normal vector. After several iterations (each point may be incorporated multiple times) the estimate converges to a good normal vector estimate, yielding a covariance matrix of the normal coordinates as a byproduct, which contains the uncertainty of the estimation.

Following the formulation presented in [145], the EKF state transition and measurement equations are written as

$$\boldsymbol{x}_t = g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \epsilon_t \tag{3.14}$$

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \delta_t, \tag{3.15}$$

where $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$ denote the state of the considered system before and after incorporating the next measurement. The quantities $\boldsymbol{u}_t$ and ϵ_t denote the control input to the system and the process noise, respectively. They are irrelevant in the following. δ_t denotes the measurement noise of the sensor and for the noise model presented in Sec. 2.4.3 simply corresponds to the standard deviation σ. The state consists only of the normal vector, i.e. $\boldsymbol{x}_t = [n_x \; n_y \; n_z]^T$; the position of the tangent plane is assumed to be fixed in the neighborhood centroid $\bar{\boldsymbol{q}}_i^+$. Since surface position and orientation do not change during the step-by-step incorporation of points measured at the same time, the system is static, i.e. $g(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) = \boldsymbol{x}_{t-1}$, and trivially $\boldsymbol{x}_t = \boldsymbol{x}_{t-1}$.

The only interesting aspect then concerns the measurement prediction function $h(\boldsymbol{x}_t)$, which predicts the measured depth for a given measurement ray using the current estimate of the surface normal $\boldsymbol{x}_t$ prior to incorporating the point which the ray intersects. Let $\boldsymbol{u}$ be a unit vector representing the direction in which the laser ray is emitted, then for the predicted depth $h(\boldsymbol{x}_t)$ and given a current normal estimate $\boldsymbol{x}_t$ the relation

$$\left\langle h(\boldsymbol{x}_t)\boldsymbol{u} - \bar{\boldsymbol{q}}_i^+, \boldsymbol{x_t} \right\rangle = 0 \tag{3.16}$$

holds, as illustrated by Fig. 3.8. It follows directly that

$$h(\boldsymbol{x}_t) = \frac{\langle \bar{\boldsymbol{q}}_i^+, \boldsymbol{x}_t \rangle}{\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle}. \tag{3.17}$$

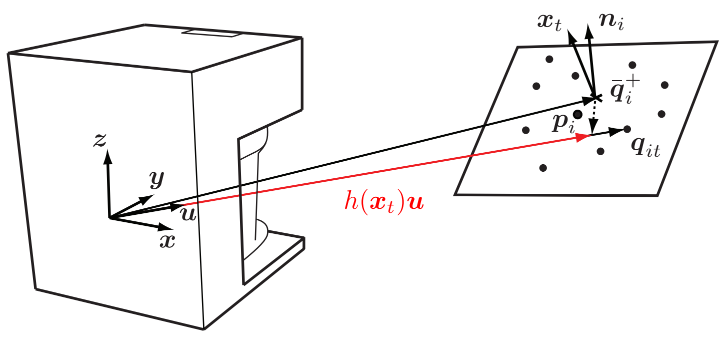


Figure 3.8: With an Extended Kalman Filter points in the neighborhood are incorporated one by one. The normal vector estimate $\boldsymbol{x}_t$ is improved at every iteration t and gradually converges to the true normal $\boldsymbol{n}_i$. $\boldsymbol{u}$ is a unit vector in the direction of the laser ray, $\bar{\boldsymbol{q}}_i^+$ the centroid of all points in the neighborhood, and $\boldsymbol{q}_{it}$ the tth neighbor of $\boldsymbol{p}_i$. The measurement function $h(\boldsymbol{x}_t)$ predicts the expected depth according to the intersection of the laser ray with the current tangent plane defined by $\boldsymbol{x}_t$.

The EKF requires the linearization of the nonlinear measurement prediction $h(\boldsymbol{x}_t)$ via a first order Taylor expansion for use in the Kalman gain update equation. The Jacobian of $h(\boldsymbol{x}_t)$ is

$$\boldsymbol{H}_t = \frac{\partial h(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} = \begin{bmatrix} \frac{\bar{q}_{ix}^+ \langle \boldsymbol{u}, \boldsymbol{x}_t \rangle - u_x \left\langle \bar{\boldsymbol{q}}_i^+, \boldsymbol{x}_t \right\rangle}{\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle^2} \\ \frac{\bar{q}_{iy}^+ \langle \boldsymbol{u}, \boldsymbol{x}_t \rangle - u_y \left\langle \bar{\boldsymbol{q}}_i^+, \boldsymbol{x}_t \right\rangle}{\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle^2} \\ \frac{\bar{q}_{iz}^+ \langle \boldsymbol{u}, \boldsymbol{x}_t \rangle - u_z \left\langle \bar{\boldsymbol{q}}_i^+, \boldsymbol{x}_t \right\rangle}{\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle^2} \end{bmatrix}. \tag{3.18}$$

The complete algorithm is shown in Alg. 3.2. The covariance matrix $\boldsymbol{\Sigma}_t$ is initialized to a scaled identity matrix with large variances of 10m^2 in each coordinate and the initial normal vector estimate to a random unit vector, since nothing is known about the orientation of the tangent surface. The loop over all points in the neighborhood may be run several times until some convergence criterion is met. If a scalar measure of the uncertainty of the normal vector estimate is desired, the trace or determinant of $\boldsymbol{\Sigma}_i$ can be returned instead of the entire covariance matrix.

In order to assess the viability of EKF estimation for normal vectors, a simple simulation was carried out, again implemented in MATLAB and again obeying the initially stated simulation principles. A 2D LRF with the characteristics of the SICK LMS-200 positioned at the origin emits laser rays in a vertical sweep, covering a horizontal FoV of α_{FoV}=180° and a vertical FoV of 60°. A planar surface of 30x30cm^2 was randomly positioned in a 2x2m^2 area 2m in front of the sensor. Its orientation was randomized uniformly over the space of quaternions, so that it could be completely incident with the laser rays, orthogonal to the rays or have any orientation in between. Fig. 3.9(a) shows an instance of the simulation. The plane was covered by 44 scan points and the normal estimated using five iterations of the loop in Alg. 3.2. The covariance matrix in Fig. 3.9(b) shows that most

Algorithm 3.2: PlaneEKF

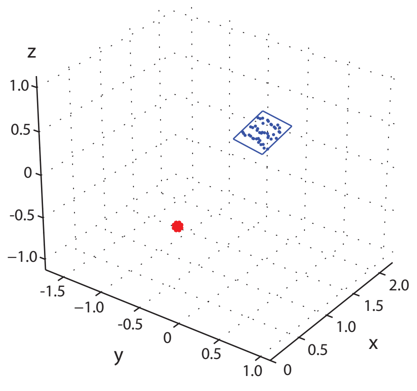
```
   Input: Neighborhood $\boldsymbol{Q}_i^+$ observed from $\mathbf{0}$
   Output: Normal vector $\boldsymbol{n}_i$
           Covariance matrix $\boldsymbol{\Sigma}_i$
 1 Initialize $\boldsymbol{\Sigma}_\mathbf{0} = 10\boldsymbol{I}_3$, $\boldsymbol{x}_0$ = random unit vector;
 2 Compute $\bar{\boldsymbol{q}}_i^+ = \frac{1}{k+1}\boldsymbol{Q}_i^{+T}\mathbf{1}_{k+1}$;
 3 for $t = 1 : k+1$ do
 4     $\boldsymbol{u} = \frac{\boldsymbol{q}_{it}}{\|\boldsymbol{q}_{it}\|_2}$;
 5     $h_t = \frac{\langle \bar{\boldsymbol{q}}_i^+, \boldsymbol{x}_{t-1} \rangle}{\langle \boldsymbol{u}, \boldsymbol{x}_{t-1} \rangle}$;
 6     $\boldsymbol{H}_t = \frac{\partial h_t}{\partial \boldsymbol{x}_{t-1}}$;
 7     $\boldsymbol{K}_t = \boldsymbol{\Sigma}_{t-1}\boldsymbol{H}_t^T(\boldsymbol{H}_t\boldsymbol{\Sigma}_{t-1}\boldsymbol{H}_t^T + \sigma^2)^{-1}$;
 8     $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \boldsymbol{K}_t(\|\boldsymbol{q}_{it}\|_2 - h(\boldsymbol{x}_{t-1}))$;
 9     $\boldsymbol{\Sigma}_t = (\boldsymbol{I}_3 - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{\Sigma}_{t-1}$;
10 end
11 $\boldsymbol{n}_i = \boldsymbol{x}_t$;
12 $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_t$;
13 return $\boldsymbol{n}_i$, $\boldsymbol{\Sigma}_i$;
```

of the uncertainty occurs in the z-direction, which is correct because the plane is almost incident with the rays. Some coupling between the coordinates can be observed in the off-diagonal elements. Fig. 3.9(c) shows a comparison of *PlanePCA* and *PlaneEKF* for ten simulation instances. Both yield virtually perfect estimates for this simple scenario, except that *PlaneEKF* sometimes yields a slightly worse estimate because of random initialization. Computationally, *PlanePCA* is much faster, as it only involves a single SVD; as initially explained, the main advantage of *PlaneEKF* lies in obtaining a measure of uncertainty. If time is not critical, *PlaneEKF* can also be initialized with the estimate of *PlanePCA* to avoid convergence to local extrema. Extensions and refinements of the EKF technique, such as the Unscented Kalman Filter (UKF) or the use of an Extended Information Filter (EIF), the dual of the EKF which avoids iterative computations, are of course possible [145] but not explored here.

3.2.8 Neighborhood Size vs. Noise vs. Quality

The previous sections have established the strength and weaknesses of several methods for normal vector estimation, resulting in the selection of *PlanePCA* on a kNN graph for the computation of normals throughout the remaining chapters. Two separate issues that are largely independent of the chosen method are addressed in this section, namely the effect of different neighborhood sizes and varying noise levels on normal vector estimation quality.

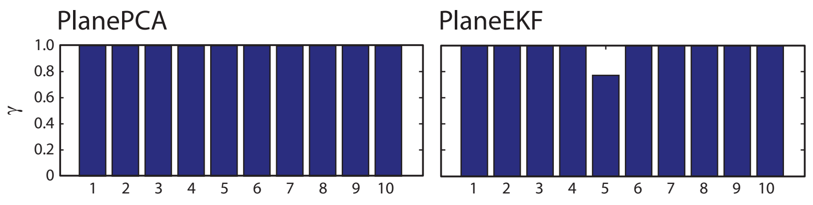
It is a known fact that k-nearest neighbors provide an excellent adaptation to variable point densitites [108, 115]. As seen in the benchmark study of Sec. 3.2.6, larger neighborhoods can smooth out and to a certain point improve estimated normal vectors. The



(a)

$$\boldsymbol{n} = \begin{bmatrix} 0.13 \\ 0.46 \\ -0.88 \end{bmatrix} \qquad \boldsymbol{\Sigma} = 10^{-3} \begin{bmatrix} 0.3 & 0.9 & -0.8 \\ 0.9 & 3.0 & -0.6 \\ -0.2 & -0.6 & 11.1 \end{bmatrix}$$

(b)



(c)

Figure 3.9: A simulation study comparing *PlanePCA* and *PlaneEKF*: (a) A plane (blue square) was randomly oriented and positioned in front of the range sensor (red dot). (b) The normal vector estimate and covariance matrix yielded by *PlaneEKF*. (c) On ten instances of the simulation the methods yield virtually identical quality. Sometimes *PlaneEKF* may converge to local extrema due to random initialization.

best value for k is, however, usually specified heuristically. In fact several publications seem to suggest that the greater k is chosen, the better the surface normal fit [16, 115]. While greater neighborhoods certainly help suppress measurement noise in the sense that they smooth out the estimated surface, they *do not* necessarily yield a better surface normal. This should become evident when considering the extreme case of fitting a plane (or higher-order surface) to the entire point cloud (the largest possible neighborhood). Thus, the question remains, at which neighborhood size k the best surface normal estimate is obtained (as compared to the ground truth normal) and how this relates to the measurement noise. Unfortunately, it is hard if not impossible to answer this question analytically, because the quality of the surface fit depends more on the exact structure of the local geometry (which is unknown) than it does on the number of neighbors. It is, however, possible to investigate this issue in simulation. In contrast to the previous benchmark, the focus of this evaluation is to examine the variation of normal vector estimation quality with respect to neighborhood size and measurement noise for a representative dataset. The simulation was carried out using the simulator introduced in Sec. 2.4.3.

Rotational scans of three different densities (7,200 points, 28,800 points, and 115,200 points) of the *Kitchen* environment shown in Fig. 2.16 were used as the benchmark dataset. Identical densities were used for three additional horizontal scans, and yet three more vertical scans. As before, the overall score for normal vector estimation quality was computed as $\gamma = \frac{1}{n}\sum_{i=1}^{n} |\langle \boldsymbol{n}_i, \hat{\boldsymbol{n}}_i \rangle|$, $\gamma \in [0, 1]$, where n is the number of points and $\hat{\boldsymbol{n}}_i$ are the ground truth normals known from the simulation model. Each of the nine scans was simulated with measurement noise standard deviations $\sigma \in \{0, 5, \ldots, 30\}$mm along the laser rays and normal vectors were computed from neighborhood sizes $k \in \{5, 10, \ldots, 60\}$ using *PlanePCA*.

The results shown in Fig. 3.10 provide several valuable insights: Firstly, very good estimates ($\gamma > 0.95$) can be attained even for sparse data, which underlines the strength

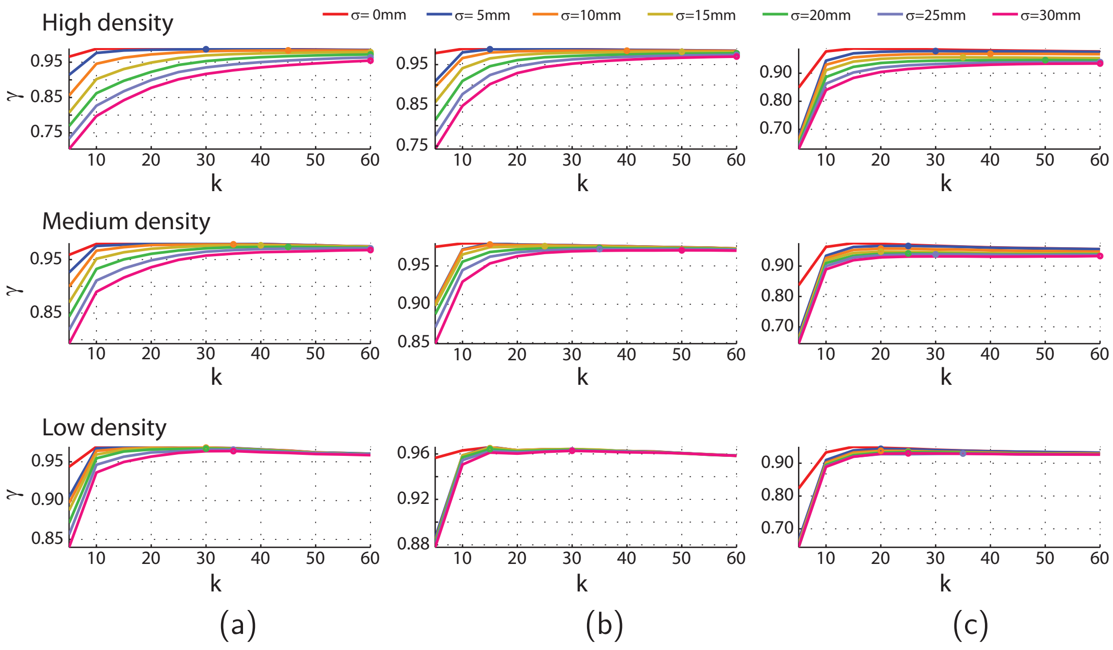


Figure 3.10: The effect of different noise levels (standard deviation σ) and neighborhood sizes (number of nearest neighbors k) on normal vector estimation quality γ: (a) Horizontal scan. (b) Vertical scan. (c) Rotational scan. The dots indicate where each curve attains its maximum.

and the adaptive nature of k-nearest neighbors. Secondly, maximum quality (indicated by a dot in each curve) is in many cases attained not for the largest examined neighborhood (k=60) but for medium-sized neighborhoods ($k \in [15, 30]$). Especially for sparse scans this means that there is no point in using larger neighborhoods to improve normal vector estimates. Finally, the effect of noise becomes negligible as density decreases. This also corresponds to intuition, as the noise has less 'leverage' in a sparse neighborhood than in a dense one.

3.2.9 Anisotropic Neighborhoods

As pointed out in Chap. 2, a popular setup for the acquisition of range data is that of a 2D LRF rotating about its axis of sight [53, 140, 158]. While convenient from a hardware perspective, it yields scans of extremely variable density, with many points in the middle of the measurement cone and only few points on the outside, see e.g. Fig. 2.16. A specific problem encountered frequently in these rotational scans, but also in vertical or horizontal scans of surfaces that are almost incident with the laser rays, is that of *anisotropic neighborhoods*. These are neighborhoods where individual scanlines are sufficiently dense, but neighboring scanlines are spaced far apart. Fig. 3.11 illustrates such a case for a vertical scan of the *Kitchen* environment. For medium k, the nearest neighbors all belong to one scanline which yields a very poor normal vector estimate (the second singular value s_2 is close to zero). The points affected by this may constitute only a tiny fraction of all points in the point cloud; however, they are especially important for subsequent processing steps

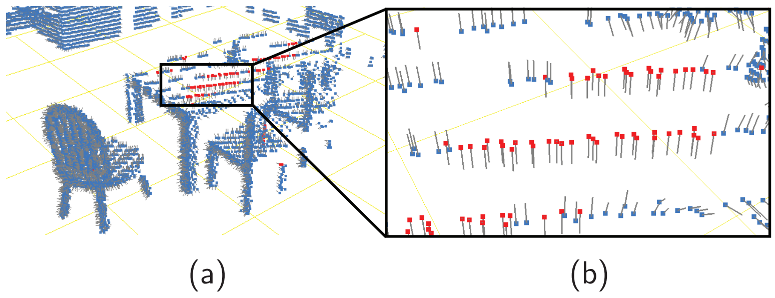


Figure 3.11: Many scans contain anisotropic point neighborhoods, where individual scanlines are sufficiently dense, but neighboring scanlines are spaced too far apart. This leads to a distortion of the normal vector estimation, because the k nearest neighbors are recruited from one scanline only: (a) Points with skewed neighborhoods from a vertical scan of the *Kitchen* dataset highlighted in red. (b) Close-up of the distorted normal vectors. The affected normals can be detected and corrected by examining the neighborhood aspect ratio σ_{ar}.

such as segmentation, as they lie on sparsely sampled surfaces. These surfaces will be discarded as noise if their normals are not estimated sufficiently well. To improve the quality of these normal vectors one could obviously set k to a very high value. This, however, would affect the estimation of all normals, possibly deteriorate the majority of estimates (as pointed out before), and introduce unnecessary computational overhead. Instead, a better alternative proposed here, is to use the ratio of the second and the first singular value obtained by the SVD of the neighborhood matrix $\boldsymbol{Q}_i^+$

$$\sigma_{ar} = \frac{s_2}{s_1}, \quad s_1 > s_2 \tag{3.19}$$

to identify neighborhoods with a degenerate aspect ratio. Recall that these singular values are proportional to the standard deviation of the data along the first and second principal component, i.e. the vectors that span the tangent plane. Singular values are to some degree preferable over the eigenvalues of the covariance matrix (which correspond to the variance) as they allow for natural interpretation of length (in mm) and are a direct outcome of the SVD, see also Appendix A.3. In the current implementation, a threshold of 10% was chosen, i.e. normal vectors were re-computed with doubled k for all points with $\sigma_{ar} < 0.1$. This method efficiently corrected the degenerate normals in all datasets used throughout this thesis, while leaving computation times virtually unchanged.

3.2.10 Consistent Orientation

A last but important aspect concerns the consistent orientation of normal vectors across the entire point cloud. While the afore-discussed estimation methods all yield vectors that are normal to the fitted tangent plane or higher order surface, these vectors may point either towards the range sensor or away from it. For unordered point clouds, consistent orientation of normal vectors can become a non-trivial problem that involves voting and propagation schemes [62, 107]. Fortunately, for point clouds obtained with range sensors,

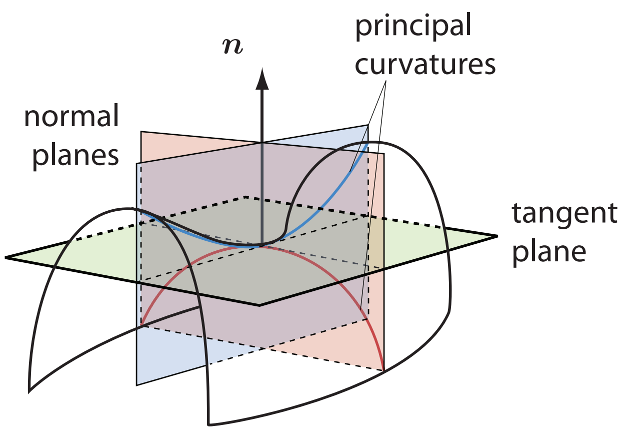


Figure 3.12: The principal curvatures of a point on a 2D surface are the minimum and maximum curvature values of the curves contained in 2D surface slices normal to the tangent plane. As shown by Euler, they occur always in normal planes which are orthogonal to each other and are oriented in the so-called principal directions.

the view point from which each point was recorded can be retained with the point cloud data, so that consistent orientation is trivially achieved by orienting the normals of all points observed from one view point to point either towards or away from that view point. Let $\boldsymbol{v}$ denote the view point from which a point cloud was recorded. By assessing the sign of the inner product of an estimated normal vector $\boldsymbol{n}_i$ with the laser ray vector $\boldsymbol{p}_i - \boldsymbol{v}$

$$\langle \boldsymbol{n}_i, \boldsymbol{p}_i - \boldsymbol{v} \rangle > 0 \Rightarrow \boldsymbol{n}_i = -\boldsymbol{n}_i \tag{3.20}$$

and inverting all normal vectors with positive inner product, consistent orientation is achieved.

3.3 Curvature

Normal vectors are a first order approximation of a sampled surface because they express the orientation of the tangent plane at each sampled point. Naturally, the computation of higher order descriptors has been of interest in various fields of research that deal with 3D point clouds. This section covers the basic computation of scalar quantities that describe surface curvature, i.e. second order descriptors for continuous surfaces.

3.3.1 Principal, Gaussian, and Mean Curvature

Historically, work on surface curvature was pioneered by Euler in his theorem of differential geometry [37] and later on by Gauss in his *Theorema Egregium* [45]. Among other things, they showed that surface curvature is an intrinsic property which is independent of the space in which the surfaces are embedded. Euler identified the so-called *principal curvatures* κ_1 and κ_2, which are the maximum and minimum values of two-dimensional curvature that occur in the space of planar slices orthogonal to the tangent plane. Fig. 3.12 illustrates these quantities. Euler showed that the normal planes in which these curvatures occur are always perpendicular to each other and are oriented in the so-called *principal directions*. For a point of an analytic, twice continuously differentiable function $z = f(x, y)$ that is situated at the origin and has its tangent plane at $z = 0$, the principal curvatures are computed from the second fundamental form [79], obtained from the Taylor expansion at $\mathbf{0}$:

$$z = f(x, y) = ax^2 + bxy + cy^2 + \text{h.o.t.} \tag{3.21}$$

as the eigenvalues of the matrix

$$\begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} = \text{eig}(\begin{bmatrix} 2a & b \\ b & 2c \end{bmatrix}). \tag{3.22}$$

Two of the most popular orientation-independent surface descriptors of curvature are *Gaussian curvature*

$$K = \kappa_1 \cdot \kappa_2 \tag{3.23}$$

and *mean curvature*

$$H = \frac{\kappa_1 + \kappa_2}{2}, \tag{3.24}$$

which are used in many different applications to characterize the fundamental shape of the local neighborhood of surface points. For an arbitrarily oriented and positioned surface, the surface description is first transformed to a principal frame with origin $\boldsymbol{p}_i$, whose z-axis is the normal vector $\boldsymbol{n}_i$ [90].

3.3.2 Estimation from Sampled Surfaces

Unfortunately, for surfaces sampled by range sensors, no analytic description $f(x, y)$ is available that would allow for the derivation of the terms a, b, and c to compute κ_1 and κ_2. Rather, the local geometry must be approximated by fitting a second or higher order surface to points in the neighborhood. Assuming it approximates the local shape reasonably well, a good curvature estimate can be computed from the surface parameters. Overviews of the large variety of methods that estimate curvature values for sampled surfaces can be found in McIvor and Valkenburg [94] and Magid et al. [90].

When using a second order fit, curvature estimation proceeds as follows: points in $\boldsymbol{Q}_i^+$ are moved so that $\boldsymbol{p}_i$ lies at the origin and then transformed to the coordinate frame defined by the principal components obtained during normal vector estimation; this yields the transformed neighborhood $\boldsymbol{Q}_i'^+$. The rotated principal quadric from (3.21) is fitted to

	$K > 0$	$K = 0$	$K < 0$
$H < 0$	Peak	Ridge	Saddle Ridge
$H = 0$	-	Flat	Minimal Surface
$H > 0$	Pit	Valley	Saddle Valley

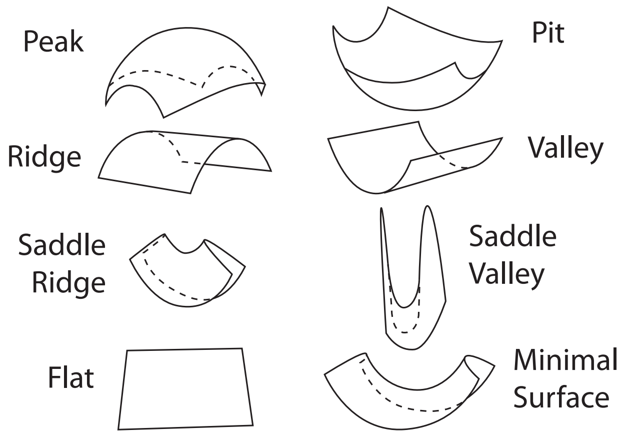


Figure 3.13: Fundamental surface types characterized by Gaussian and mean curvature, as presented in [16].

this set of points by minimizing

$$\|[\boldsymbol{R}_i, \mathbf{1}_k]\, \boldsymbol{c}_i\|_2\,, \tag{3.25}$$

where $\boldsymbol{R}_i$ contains the terms $[q'^2_{ix}\ q'_{ix}q'_{iy}\ q'^2_{iy}\ q'_{iz}]$ in each row and $\boldsymbol{c}_i = [a'\ b'\ c'\ d']^T$ is the coefficient vector, which is obtained by an SVD of $\boldsymbol{R}_i$. Using the relation $a = -a'/d$, $b = -b'/d$, and $c = -c'/d$, the curvatures are then computed as

$$K = 4ac - b^2 \tag{3.26}$$

$$H = a + c. \tag{3.27}$$

The values of K and H are difficult to visualize, because they represent two scalar quantities for each point that cannot be intuitively integrated into a coloring scheme. For various purposes researchers have, however, used the two curvature values to distinguish eight fundamental local surface types, that can be visualized nicely. Following Besl and Jain [16], K and H are used to label the shape of neighborhood $\boldsymbol{Q}_i^+$ as *Peak*, *Ridge*, *Saddle Ridge*, *Flat*, *Minimal Surface*, *Saddle Valley*, *Valley*, or *Pit* type, as illustrated in Fig. 3.13.

Visualization of the surface labels for different simulated datasets provides a good way of assessing the robustness of the local descriptor with respect to different noise levels. Figs. 3.14(a)-(c) show the labels computed from $k = 50$ neighbors for the dense point cloud from Fig. 2.16(c), simulated with noise levels of $\sigma = 0$mm (no noise), $\sigma = 15$mm, and $\sigma = 30$mm, respectively. For the noiseless dataset shown in Fig. 3.14(a) flat surfaces, fold and crease edges are all correctly and consistently labeled. However, for high noise, Fig. 3.14(c), and even for moderate noise, Fig. 3.14(b), the estimation is heavily affected, with labels oscillating wildly between *Saddle Ridge* and *Saddle Valley*. Note that the normal vectors computed before the curvature estimation were oriented consistently, see Sec. 3.2.10; so this oscillation is not due to the principal frames being aligned in different directions. Also note that with $k = 50$ the neighborhood size is already rather large.

The high sensitivity to noise could not be reduced by choosing larger neighborhoods;

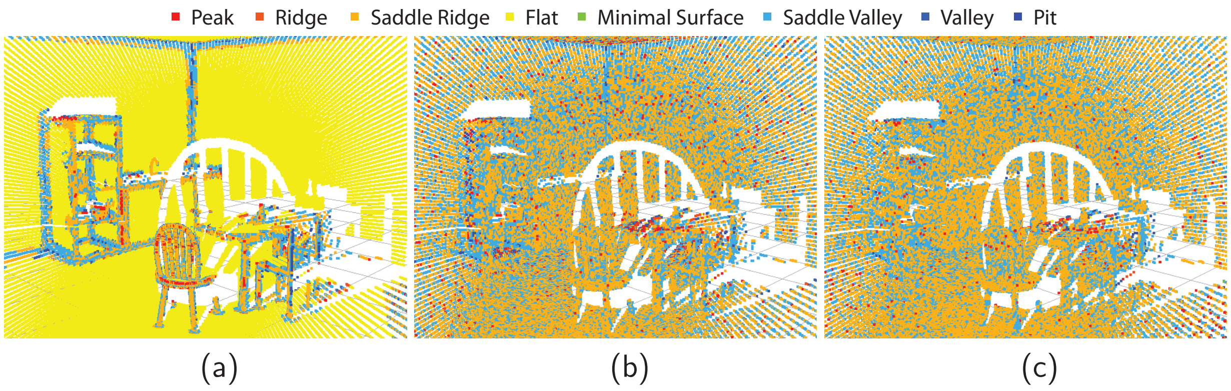


Figure 3.14: Labeling points by curvature shape types is problematic for noisy datasets: (a) Without noise ($\sigma = 0$mm) the labels are consistent. (b) With the typical noise level of LRFs ($\sigma = 15$mm) labels oscillate wildly. (c) For high noise levels ($\sigma = 30$mm) the surface labels are even more unstable.

the results remained largely identical to those depicted in Figs. 3.14(b)-(c) for $k = 100$ and even for $k = 200$. While this may be to some degree due to the properties of the dataset (many planar surfaces, sharp edges), the lack of robustness of descriptors based on Gaussian and mean curvature for noisy surface samplings has also been observed in several other studies [42, 111, 148]. A conclusion to be drawn from this is that curvature estimates can only serve as meaningful descriptors for surfaces where the sampling is dense in comparison to the depth noise. In fact, in the results presented by Besl and Jain [16], consistent descriptors are only obtained for dense scans of very simple geometric primitives. The dataset shown in Fig. 3.14 is too sparse (in comparison to the LRF noise level and the object extents) to allow for the use of curvature descriptors, as are the majority of real-life datasets obtained from similar setups.

3.3.3 Surface Variation

Rather than using detailed second-order curvature estimates that are corrupted by noise and introduce considerable computational overhead, several researchers have used information of the plane fit to infer a scalar measure of *surface normal variation.* This measure essentially intends to capture in one number, how planar or non-planar the local geometry is. In [115] it was observed that the residual error of a plane fit corresponds closely to surface variation. This is consistent with intuition, as a heavily bending surface should deviate a lot from a fitted plane. However, the residual still requires extra computation and furthermore is not normalized. A better solution was pointed out by Pauly et al. [108]. Since the PCA used for normal estimation yields the variances along the three principal components, surface variation can be computed from the ratio of the smallest variance (in the direction of the normal) and the sum of all variances. This measure has two advantages: Firstly, it comes at no additional computational cost as it is a natural byproduct of the PCA and secondly, it is normalized. Again, instead of using the eigenvalues of the covariance matrix (variances), here the use of singular values (proportional to standard deviations) is preferred. Surface variation is thus computed as

$$\sigma_{sv} = \frac{3s_3}{s_1 + s_2 + s_3}, \quad s_1 \geq s_2 \geq s_3, \tag{3.28}$$

which represents a natural coefficient $\sigma_{sv} \in [0, 1]$, where $\sigma_{sv} = 0$ indicates a perfect plane and $\sigma_{sv} = 1$ a perfect sphere. Note that the terms *surface variation* and *curvature value* are used interchangeably in the following. More so-called *shape factors* that can be computed from singular values are described in [44]. Since surface variation is a normalized scalar measure, it is well-suited for visualizing curvature using a grayscale or other color palette. As an example, the datasets shown in Fig. 2.16 are grayscale-colored by curvature value computed for $k = 30$.

3.4 Advanced Features

The quantities introduced in the previous sections represent the most fundamental point features for the description of local surface geometry. For several applications, more advanced features have been developed that aim to capture higher order information, usually by collecting neighboring points in discrete bins and counting these in histogram representations. All of them rely on a principle local coordinate frame obtained from normal vector estimation and provide greater expressive power at the cost of increased dimension and storage requirements. A selection of these advanced features is summarized in this section.

3.4.1 Spin Images

One of the most popular regional shape descriptors is the *spin image*. Spin images were conceived by Johnson et al. [74] to provide a rotationally invariant representation of the point distribution in the local neighborhood. A cylinder of radius r and height h is centered on a given surface point $\boldsymbol{p}_i$, with its axis aligned with the estimated normal vector $\boldsymbol{n}_i$. The cylinder is divided into j radial rings and k vertical slices. The resulting segments constitute a set of bins. Counting the number of points that fall into each bin yields a 2D histogram that can be represented as a $j \times k$ image, the so-called spin image. Spin images are rotationally invariant because the points are summed in radial rings. They have been widely used for purposes of object recognition [74], classification [118] and 3D registration [67], to name a few. Several variants of spin images have been proposed that count not the points but, e.g. the neighboring normal vectors [36].

3.4.2 3D Shape Contexts, Harmonic Shape Contexts

Frome et al. introduced two more regional descriptors that operate in similar fashion to spin images [42]. The *3D shape context* descriptor places a sphere at point $\boldsymbol{p}_i$ with its north pole oriented with the surface normal vector. It counts the points in wedge-formed sections defined by j azimuth and k elevation divisions, yielding a $(j+1) \times (k+1)$ histogram. Building on this descriptor, *harmonic shape descriptors* use the counted points to approximate the neighborhood by a superposition of complex spherical harmonic basis functions, similar to the way in which a Fourier series approximates a 2D function. The amplitudes of this transformation are used as the feature vector, its dimension depending only on the bandwidth of the approximation and not on the number of subdivisions j and k.

3.5 Segment Features

The features described up to now represent quantities that characterize individual points by shape properties of their local neighborhood. For methods presented in the following chapters, it will also be relevant to characterize groups of points called *segments* to allow for a more abstract representation. Within the context of this thesis, a segment refers to a collection of points that make up a homogeneous surface, i.e. a surface patch that exhibits a smooth change in curvature. Such groups of points can be considerably more complex than the local neighborhoods of points described before and therefore require different descriptors. Some such quantities are presented in this section.

3.5.1 Basic Descriptors

A number of straightforward features can be defined from the fact that rather than considering an individual point, features are now computed for a set of n points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$. These are

- $\bar{\boldsymbol{p}}$: The centroid of points in the segment
- $\bar{\boldsymbol{n}}$: The mean normal vector of points in the segment
- $\bar{\sigma}_{sv}$: The mean surface variation of points in the segment
- $s_1,\ s_2,\ s_3$: The singular values obtained from a PCA of points in the segment
- σ_{sv}: The surface variation of points in the segment
- σ_{ar}: The aspect ratio of points in the segment projected to the tangent plane
- $\boldsymbol{v}_1,\ \boldsymbol{v}_2,\ \boldsymbol{v}_3$: The principal frame obtained from a PCA of points in the segment
- $\boldsymbol{n} = \boldsymbol{v}_3$: The normal vector of a plane fitted to points in the segment
- V_{oo}: The volume of the object-oriented bounding box (OOBB), computed from the boundary data points in the principal frame

The first three features represent mean values that can simply be aggregated from averaging over the individual point features. However, one must be careful with the descriptiveness of theses quantities. While a reasonably good estimate of $\boldsymbol{n}$ can be obtained by averaging over the computed normals $\boldsymbol{n}_i$ in the segment, the surface variation σ_{sv} of the segment cannot be estimated reliably from $\bar{\sigma}_{sv}$, the average of the individual $\sigma_{sv,i}$ of the points in the segment. This is because the relative variance of a plane fit in the local neighborhood of each point may differ greatly from the relative variance of a plane fit over all points. As an example, consider a densely sampled cylinder. The $\sigma_{sv,i}$ of each point will be small, their average indicating a near planar surface; yet the segmented cylinder is by no means planar and has a large σ_{sv}. Since for the computation of the other features a PCA of points in the segment is necessary anyway, only $\bar{\boldsymbol{p}}$ should therefore be computed from averaging over the points.

3.5.2 Minimum-volume Bounding Boxes

Minimum-volume bounding boxes (MVBBs) enclose the points with the tightest possible rectangular 3D frame. If a small (bounded) approximation error is acceptable, they can be efficiently computed for arbitrary 3D point sets [10]. As a geometric construct, they are useful in collision checking and object shape decomposition, e.g. for applications in robot grasping [68]. The volume of an MVBB may be used to characterize a segment, although - as with OOBB volume - these descriptors are sensitive to individual outliers.

3.5.3 Convex Hulls

An even more detailed approximation of the enclosing shape than MVBBs are convex hulls, which represent the tightest possible polyhedral (or polygonal) cover of the point set [114]. For surface segments both the 2D convex hull of the points projected to the normal plane as well as the 3D convex hull of the points may contain valuable information. Interesting scalar quantities are the volume of the 3D convex hull (respectively the area of the 2D convex hull) as well as the number of convex hull points.

3.5.4 3D Zernike Descriptors

Proceeding in the direction of even more advanced features, by far the most complex descriptors are those that aim to reconstruct the shape of an entire segment from the superposition of basis functions. Similar to harmonic shape contexts that do the same for a local neighborhood, see Sec. 3.4.2, *3D Zernike descriptors* [106] are able to reconstruct the geometry of the original point set by a series of coefficients called Zernike moments, conceptually similar to the Fourier coefficients in function approximation. These moments then represent a sophisticated feature vector whose dimension depends on the chosen order of approximation. While their superior expressive power is demonstrated in [106] for retrieval of detailed voxelized noise-free objects, the descriptor is largely unusable for real-life point clouds obtained with noisy sensors. The computation of 3D Zernike descriptors up to fifth order for various datasets used in this thesis yielded highly dissimilar feature vectors even for simple and visually clearly similar segments. This seems to be due to the fact that the point clouds are sparse and have inhomogeneous density across similar segments. Therefore, individual points at the brink of a segment can greatly distort the computed moments.

3.6 Discussion

The various features presented throughout this chapter offer a wide variety of possibilities to compute descriptive numerical quantities that characterize point clouds. They were presented in order of increasing complexity: the normal vector, which received by far the most detailed treatment, was shown to be computationally rather cheap, followed by more expensive fits for curvature descriptors, and finally advanced features with much higher computational cost, such as spin images. The same was true for the segment features, which range from simple averages to complex vectorial quantities such as 3D Zernike descriptors. A rule of thumb that can be inferred from all of the features presented in this chapter is that the more sophisticated the descriptor, the more susceptible it is to noise and the less usable for 3D point clouds obtained with noisy range sensors. Since the computational cost increases with feature complexity, one therefore has to carefully choose features so as to find a balance between descriptive power, computational cost and feature robustness. Throughout the following chapters normal vectors will play a predominant role as both a point and a segment feature.

An important conclusion to draw from the experimental evaluation of normal vector estimation methods is that the choice of the 'best' algorithm depends mostly on the graph structure chosen by the user. If a triangular mesh such as the DT graph is available, then the *AreaWeighted* method yields the best speed at acceptable quality. As outlined before, the construction of such a graph is non-trivial. In all of the processing steps presented in subsequent chapters, the construction of meshes is not necessary. Therefore, the neighborhood graph of choice is the kNN graph, also because k nearest neighbors can efficiently be retrieved without the actual construction of the graph, see Appendix A.1. As was seen in the benchmark study, for this type of graph, *PlanePCA* is the universal method of choice because of its superior performance in terms both quality and speed. It is therefore used exclusively throughout the following chapters.

4 From Points to Segments

A school of fish, forming a big swordfish
Finding Nemo, © 2003 Disney/PIXAR

Summary

This chapter describes how to segment point clouds into homogeneous surfaces. The segments can be used as a more abstract representation for scanned objects. Following an overview of different techniques that have been used for 3D range data segmentation, a toolchain is presented that efficiently segments point clouds by grouping regions with similar normal vectors. The method is enhanced by a strategy to mend oversegmentation that is caused by partial occlusions. The efficiency of the proposed framework is demonstrated in simulation and experiment.

In the chain of abstraction that leads from a level of raw point data to interpretable objects, the previous chapter has demonstrated the computation of descriptive quantities for individual 3D points. The next step, covered in this chapter, is the grouping of points that exhibit similarity in the space of computed features. This process is commonly referred to as *clustering* or *segmentation* and leads to the description of objects in the form of *segments*. Not only does it allow for a reduction of complexity because scans are subsequently no longer processed on the point level, but it also provides a natural description for surface-sampled data obtained by range sensors.

A categorization of the plethora of existing methods for point cloud segmentation is provided in Sec. 4.1. Sec. 4.2 describes the algorithm used for efficiently retrieving a grouping of homogeneous surfaces. In Sec. 4.3 the problem of oversegmentation caused by partial occlusions is considered and a solution for repairing certain types of oversegmented surface patches is presented. Sec. 4.4 presents an evaluation in both simulation and experiment, which demonstrates the applicability and efficiency of the proposed segmentation framework. Sec. 4.5 discusses several issues and possible improvements of the approach.

4.1 Related Work

The segmentation of range data has been addressed in various contexts. In loose historical order, existing approaches can be grouped into five different categories:

Curvature-based Probably one the earliest influential works on range image segmentation is that of Besl and Jain [16], in which they suggest to segment range images by assigning surface labels computed from principal curvature estimates to each pixel and then group regions with homogeneous labels. While this works well for some of the datasets examined in their paper, it was observed in several subsequent studies [61, 111] that the method yields high rates of oversegmentation because the curvature values cannot be accurately estimated for noisy data, see also Chap. 3.

Edge-based Within the context of range images, which facilitate the direct application of computer vision segmentation algorithms, several researchers have carried out segmentation based on edge detection. This usually involves identifying fold or crease edge pixels with suitable window operators that estimate normals, curvature etc., and then grouping pixels that lie within the identified boundaries of connected edges [13, 18, 58, 122, 154, 155]. The segmentation performance is, however, highly dependent on the connectedness of boundary image pixels and therefore error-prone in the presence of noise and missing data.

Scanline-based For data obtained with actuated 2D laser range finders, a number of researchers have extracted homogeneous regions directly from the 2D scanlines [71, 72, 78, 103]. While these methods are reported to perform considerably faster than pixel-based approaches, the majority of them extract only planes. Furthermore, since they rely on a pre-defined structure in the scanlines, they are not generically applicable to unordered point clouds or variable density point clouds, such as those obtained by rotational scans.

Geometric primitives-based Apart from the many split-and-merge approaches, which divisively cluster the points by repeated plane fitting and therefore only extract planar regions, several publications consider fitting higher-order surfaces and primitives to the range data [8, 92, 127] to solve the segmentation problem. The advantage of geometric primitives-based algorithms is that they simultaneously solve the problems of surface segmentation and reconstruction. However, apart from the high computational complexity, the segmentation result is directly dependent on the number and type of the used geometric primitives. The approach proposed in [127] partially reduces this complexity by employing a clever RANSAC strategy; however, as a direct consequence the segmentation results are non-deterministic and still have high computational cost for a higher number of primitives.

Smooth region-based Several recent publications report the efficient application of some form of region growing over neighboring points that fulfill a normal vector angle criterion [115, 119, 174]. The advantage of these approaches is that they are applicable to generic point cloud data, efficiently yield repeatable segmentation results and make no explicit assumption about the shape or type of primitives. An algorithm of this type is developed in the following.

Obviously different categorizations of the cited algorithms are possible - on a fundamental level all can be classified as either agglomerative or divisive clustering algorithms using a similarity measure based on local surface features. Remarkably, a large number of the referenced works [8, 13, 16, 32, 58, 61, 71, 72, 92, 154, 155] perform segmentation on rather academic data sets that consist of a few, fully visible and densely sampled objects. For many of the earlier publications, this is explained by the lack of hard- and software resources that allowed for the acquisition and processing of more complex point clouds. However, even several more recent publications use such data sets to demonstrate the applicability of an algorithm to 'real' range data. While these data sets, typically consisting of range and intensity images of several polyhedral objects, helped to study fundamental properties of certain segmentation approaches, they have little to do with the range data encountered in real-life environments. A second shortcoming of existing publications on the topic of range data segmentation is that, with the notable exception of scanline-based algorithms, very few of them report scan or segmentation times: Only [71, 72, 78, 103] report to achieve segmentation times on the order of seconds for medium-sized range images, [127] achieves a surface reconstruction within seconds using the mentioned RANSAC approach, and [155] reports the computation of edge-boundaries within seconds. All of these rely on scans of uniform and high density. The remaining publications either report segmentation times on the order of minutes and hours or do not report segmentation times at all. As for the consideration of occlusions in range data, few works exist [13, 20, 21, 32, 147, 152], which are discussed in Sec. 4.3.1.

4.2 Surface-based Segmentation

In this section, several strategies for segmenting point clouds into homogeneous surface regions are presented. The algorithms make use of the features computed in Chap. 3. While various advanced features such as curvature and higher order descriptors have been used throughout the segmentation literature, it shall be seen that for the considered scenarios, normal vectors provide all the information needed to obtain a meaningful grouping of surfaces.

4.2.1 Segmentation Problem

For a given point cloud, the goal is to find a partitioning of n_s segments $\mathcal{S}_j$, such that

$$\begin{aligned} &\mathcal{S}_j = \left\{ \boldsymbol{p}_{j1}, \boldsymbol{p}_{j2}, ..., \boldsymbol{p}_{jn_j} \right\}, \quad n_j > 0 \; \forall j \in [1, n_s] \\ &\mathcal{S}_j \cap \mathcal{S}_k = \emptyset \; \forall j \neq k \end{aligned}$$

are disjoint regions, each containing at least one point. The objective of extracting homogeneous surfaces is formulated by requiring that the segments be collections of points with smooth curvature change. This criterion is rather liberal and allows for the extraction of almost arbitrarily shaped regions, as long as they make up smoothly varying surface patches. Specifically, the extracted surfaces are not required to conform to certain geometric primitives, such as planes, cylinders, tori etc. Mathematically, the criterion can be expressed implicitly by a constraint on the maximum angle between neighboring normal vectors, as is done in Sec. 4.2.5.

4.2.2 Pre-processing

Raw data sets should be cleaned and pruned before segmentation, in order to reduce the number of points to be processed in subsequent steps. The following may be reasonable for (indoor) point cloud data:

- Minor calibration errors can be corrected by fitting a plane to all points within a band of z-coordinates around 0 (the ground plane points) and then re-orienting all points such that the plane's normal coincides with the global z-axis.
- The ground plane is extracted by directly removing all points within a band of z-coordinates around 0.
- The ceiling may be extracted by removing all points within a band of z-coordinates around the ceiling height, which can be determined from a histogram over (corrected) z-coordinates.

These pre-processing steps exploit the characteristics of indoor data and need to be adapted for other scenarios. However, as ground plane and ceiling points often make up more than 50% of the data, their removal is important for both complexity reduction and spatial separation of the remaining (valuable) points.

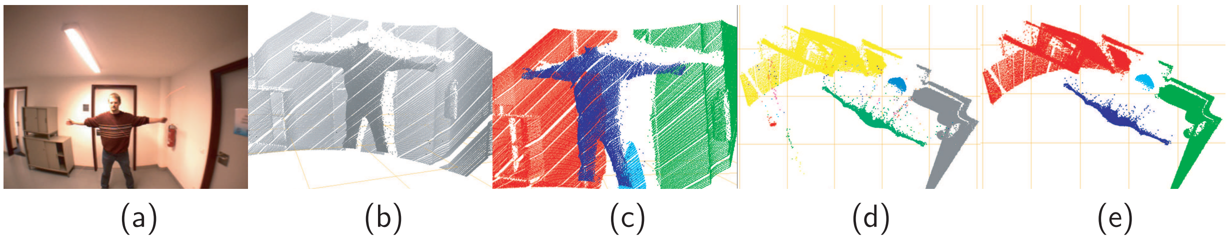


Figure 4.1: Disjoint point sets can be efficiently isolated with the segmentation scheme presented in [172]: (a) Photograph of scanned scene. (b) Resulting point cloud. (c) Segmented point cloud. (d) Top view before removing noise. (e) Top view after removing noise.

4.2.3 Basic Point Segmentation

The simplest form of segmentation consists in grouping spatially separated portions of points. While this may seem like a trivial application, it is a processing step often required to isolate individual objects after ground plane and ceiling points have been removed from a scan as described in Sec. 4.2.2. As previously explained, while humans can visually separate disjoint portions of points with ease, for the computer the points are nothing but a list of x-, y-, and z-coordinates that may even be completely unordered. Thus establishing a meaningful grouping requires a suitable distance metric and a clustering algorithm.

A simple and efficient strategy that operates only on the raw data point coordinates $[p_x\ p_y\ p_z]$ was presented in [172]. As a first step, a kd-tree of all points in the cloud is constructed; then the algorithm steps through the list of all points, connecting each point to all neighbors within a specified distance threshold r. The used distance metric is simply the Euclidean distance

$$\rho_e(\boldsymbol{p}_i, \boldsymbol{p}_j) = \left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\|_2 . \tag{4.1}$$

The neighbors within the fixed radius r can efficiently be retrieved by means of the kd-tree, see Appendix A.1, which makes the method rather fast. Conceptually, it is equivalent to region growing aproaches or graph-based methods that cut a fully connected graph at a specified distance threshold. However, in contrast to region growing no method for seed point selection is required, and in contrast to graph methods no explicit construction of the graph is necessary. The general time complexity for this clustering method involves $O(n \log(n))$ for kd-tree construction and $O(\log(n))$ for each nearest neighbor query and thus amounts to $O(kn \log(n))$. If one is willing to accept greedy behavior, which may occasionally leave two regions that should have been connected disjoint, the complexity can be significantly reduced to $O(\frac{kn}{\bar{k}_r} \log(n))$, where $\bar{k}_r$ denotes the average number of neighbors encountered within radius r [172]. The speedup is achieved by skipping all points that have already been visited as the neighbor of other points. Alg. 4.1 states the pseudocode of the basic point segmentation algorithm with the optional greedy **if**-statement in line 2. Since sensor noise tends to cause irregularly spaced outliers, these are isolated in clusters with few points. As a result, the scan can be 'cleaned up' by pruning all clusters with less than a specified $n_{\min}$ points. Fig. 4.1 shows the segmentation of a real data set obtained with a

SICK LMS-400 mounted at a 45° angle with respect to the horizontal plane on the turning ACE robot. Using r=1cm and $n_{\min}$=1,000, 150,000 points of raw data were segmented within 468ms on a standard desktop PC with 1.8GHz and 2GB RAM.

Algorithm 4.1: Basic Point Segmentation

```
Input: $n$ points $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$
       Parameters $r$ and $n_{\min}$
Output: Segments $\mathcal{S}_1, \dots, \mathcal{S}_{n_s}$
1  for $i = 1 : n$ do
2      Greedy variant: if ($\boldsymbol{p}_i$ has segment) then Continue;
3      Find $k_r$ ($\leq k$) nearest neighbors within $r$;
4      for $j = 1 : k_r$ do
5          if ($\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ belong to same segment) then Continue;
6          if ($\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ belong to different segments) then
7              Merge segments $\mathcal{S}(\boldsymbol{p}_i)$ and $\mathcal{S}(\boldsymbol{p}_j)$;
8          end
9          if (either $\boldsymbol{p}_i$ or $\boldsymbol{p}_j$ has no segment) then
10             Add it to other segment;
11         end
12         if (both $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ have no segment) then
13             Create new segment with both points;
14         end
15     end
16 end
17 Remove all segments with $|\mathcal{S}_j| < n_{\min}$ points;
18 return $\mathcal{S}_1, \dots, \mathcal{S}_{n_s}$
```

4.2.4 Basic Surface Segmentation

The algorithm described in the previous section only isolates disjoint point sets without considering surface properties. When surfaces are close to each other or connected by a chain of noisy outliers they will automatically end up in one segment, as can be seen e.g. with the cupboard in Fig. 4.1. In order to create an algorithm capable of segmenting homogeneous surfaces, more advanced features than just the point coordinates must be used, which may also require the use of different distance metrics. Naturally, the most straightforward extension of Alg. 4.1 is the use of estimated normal vectors for each point. Although similarity between normals is best expressed by the inner product, surprisingly the use of a Euclidean distance metric over a $[p_x\ p_y\ p_z\ n_x\ n_y\ n_z]$ feature space already yields good segmentation results, as was shown in [174]. Fig. 4.2(a) shows the segmentation of the basic data set with two boxes and a cylinder from Fig. 3.4 using the greedy version of Alg. 4.1 with $r = 0.14$ and $n_{\min} = 40$. Note that for the mixed feature space, the threshold r associated with the Euclidean distance metric over all features is dimensionless and has little interpretable meaning. As outlined in [174], more control over the segmentation can be gained by introducing two thresholds on two dissimilarity measures. As before, a threshold r on the Euclidean dissimilarity metric $\rho_e(\boldsymbol{p}_i, \boldsymbol{p}_j)$ specifies how far the 3D

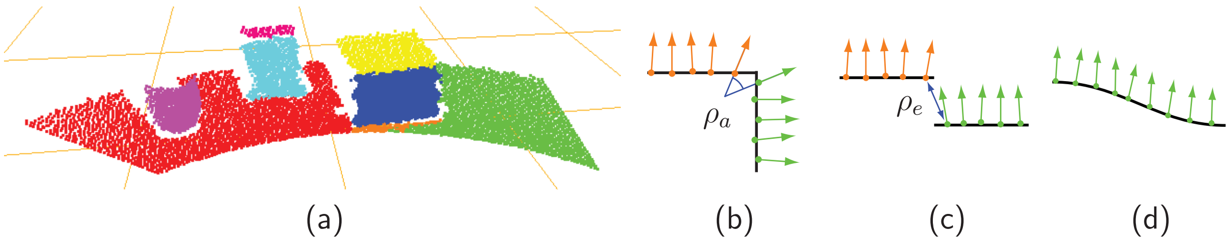


Figure 4.2: Segmentation using a feature space of point coordinates and normal vector coordinates: (a) A Euclidean distance metric over all features already yields good results, but the threshold r is dimensionless and has little interpretable meaning. (b)-(c) More control over the segmentation is obtained with two thresholds on two different dissimilarity measures, where (b) corners are detected with ρ_a, (c) steps are detected with ρ_e, and (d) smooth surfaces are not segmented.

coordinates of two points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ can be away from each other to still belong to the same segment. In addition, the threshold $\alpha_{\max}$ on the angular dissimilarity measure $\rho_a(\boldsymbol{n}_i, \boldsymbol{n}_j) = \cos^{-1}(|\langle \boldsymbol{n}_i, \boldsymbol{n}_j \rangle|)$ expresses the maximum angle that the associated normal vectors of two points are allowed to span to still belong to the same segment. Figs. 4.2(b)-(d) illustrate how both measures affect the segmentation of different surface discontinuities.

4.2.5 Refined Surface Segmentation

In the family of methods that segment regions with smooth curvature change (rather than just extracting planes), the algorithm presented in [115] makes use of both a normal vector criterion and a threshold r_{th} on surface curvature. The method uses a classical region growing technique, in which seed points are selected as points with minimal curvature below r_{th}, i.e. regions are grown from the 'smoothest' points encountered in the point cloud. Curvature is computed from the residual error of a plane fit to each point, which is unnormalized. While good segmentation performance is demonstrated on the considered datasets, the method has several significant drawbacks: Firstly, the threshold r_{th} is chosen as a fixed percentile of the sorted residual values encountered in the data, which makes the segmentation result data-dependent. Secondly, the method is susceptible to noisy outliers that can connect otherwise separate regions. As an example consider Fig. 4.3, where an individual outlier with a smooth curvature estimate would connect separate surfaces, e.g. the seat surfaces of two scanned chairs.

Some of these shortcomings could be fixed by using a threshold on the normalized surface variation σ_{sv} instead of the residual error, see Sec. 3.3.3. However, in the course of examining different variants of segmentation methods, the curvature criterion was found to be redundant to the normal vector angle criterion and therefore unnecessary. This is obvious when considering the fact that wherever curvature values are high, neighboring normal vectors must deviate a lot from each other.

The approach in [174] outlined in the previous section makes use only of a normal vector criterion and does not depend on curvature estimates. As explained, however, it uses a fixed search radius to speed up nearest neighbor queries. While this yields good segmentation results for homogeneous density scans, it leads to either over- or undersegmentation for

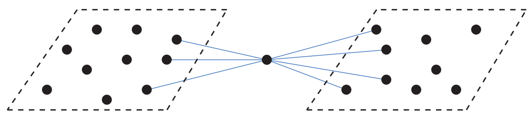


Figure 4.3: Using only continuous curvature values as a criterion for region growing produces unwanted connections between separate regions in the presence of individual outliers. In the depicted case the outlier between the two rectangular surfaces will have the same curvature value as its neighbors and will connect the two regions.

variable density data, such as that obtained by a rotational scan. A remedy for this is to store the average distance of every point $\boldsymbol{p}_i$ to its k nearest neighbors

$$\bar{\rho}_i = \frac{1}{k} \sum_{j=1}^{k} \|\boldsymbol{p}_i - \boldsymbol{q}_{ik}\|_2 \tag{4.2}$$

during normal vector estimation. This distance can be used to efficiently adapt the search radius during segmentation, while guaranteeing that on average half of the neighbors are checked for a possible connection. As a further refinement, a threshold $\rho_{\max}$ specifies a distance at which no two points should be connected, regardless of the average distance of their neighborhood. This prevents outliers from connecting separate surfaces across gaps in sparse neighborhoods, see Fig. 4.3, and thus overcomes the shortcoming of the algorithm presented in [115]. The adaptive search radius ρ_s is then computed as

$$\rho_s = \min\left(\bar{\rho}_i, \rho_{\max}\right). \tag{4.3}$$

Note that the search radius is sufficiently large even for points with anisotropic neighborhoods, because for these $\bar{\rho}_i$ has been recomputed from a larger neighborhood during normal vector estimation, see Sec. 3.2.9. It turns out that the two parameters $\alpha_{\max}$ and $\rho_{\max}$ are sufficient to produce an efficient and accurate segmentation of homogeneous surface regions. Noise will be isolated in segments with few points, which is why it is again reasonable to remove all segments with less than a specified $n_{\min}$ points. Assuming that normal vectors and average distances have been computed for a given k, the refined segmentation procedure can then be summarized as in Alg. 4.2.

4.3 Considering Occlusions

The algorithm described this far yields a segmentation of smooth surfaces. However, one of the biggest problems encountered in range scans from individual view points is that of objects occluding other objects or large surfaces. Such occlusions cause oversegmentation that cannot be remedied by the segmentation algorithm alone. If the segmented primitives are to serve as input to subsequent algorithms, e.g. for object recognition, the last step at the segmentation stage should be to attempt to explain at least basic occlusions and merge oversegmented surface patches. This section addresses existing approaches, outlines

Algorithm 4.2: Surface-based Segmentation

```
Input: $n$ points $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$
       Estimated normals $\boldsymbol{n}_1, \dots, \boldsymbol{n}_n$
       Parameters $\alpha_{\max}$, $\rho_{\max}$, and $n_{\min}$
Output: Segments $\mathcal{S}_1, \dots, \mathcal{S}_{n_s}$
1  for $i = 1 : n$ do
2      Calculate $\rho_s = \min(\bar{\rho}_i, \rho_{\max})$;
3      Find $k_r$ $(\leq k)$ nearest neighbors within $\rho_s$;
4      for $j = 1 : k_r$ do
5          if $(|\langle \boldsymbol{n}_i, \boldsymbol{n}_j \rangle| < \cos(\alpha_{\max}))$ then Continue;
6          if ($\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ belong to same segment) then Continue;
7          if ($\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ belong to different segments) then
8              Merge segments $\mathcal{S}(\boldsymbol{p}_i)$ and $\mathcal{S}(\boldsymbol{p}_j)$;
9          end
10         if (either $\boldsymbol{p}_i$ or $\boldsymbol{p}_j$ has no segment) then
11             Add it to other segment;
12         end
13         if (both $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ have no segment) then
14             Create new segment with both points;
15         end
16     end
17 end
18 Remove all clusters with $|\mathcal{S}_j| < n_{\min}$ points;
19 return $\mathcal{S}_1, \dots, \mathcal{S}_{n_s}$
```

different policies for reasoning about occlusions, and finally describes the distance measures and methodology used for mending oversegmented surface patches.

4.3.1 Existing Methods

The problem of dealing with occlusions in range data has been addressed by only a small number of researchers, partly because the majority of works focus on the segmentation of a few unoccluded objects. Of the relatively few occlusion-related publications, the majority address the problem of surface reconstruction and can be divided into model-free approaches [21, 32, 152] and those assuming some sort of knowledge of the object shape or type [13, 20, 147]. The latter generally involves the classification of points and surfaces or even the recognition of entire objects and is therefore not considered, because this is already one step beyond segmentation. [21, 152] demonstrate nice results filling holes in smooth surfaces from point samples alone; however, these methods firstly assume the existence of a relatively dense triangular mesh that encloses the hole region to be identified. Secondly, they quietly assume that any encountered hole should be filled and thus neglect reasoning about the plausibility of occlusion. Both assumptions may not be fulfilled.

Reasoning about occlusions can become arbitrarily complex as one considers the shadow of arbitrarily shaped objects in the foreground cast onto arbitrarily shaped objects in the background. As a consequence, explaining complicated instances of occlusion requires either model-specific knowledge and/or densely sampled geometry. Ambiguous cases can only be satisfactorily resolved by acquiring more data from a different view point. However, assuming a decision must be taken only with the currently available data, it should at least be possible to mend oversegmented patches that are *likely* to have resulted from shadows of objects cast onto planar surfaces. As an example consider the shadow of the kitchen chair in Fig. 2.16(c), which shatters several contiguous surfaces, including the table surface and the front panel of the sink cabinet. Without mending the patches at the segmentation stage, these surfaces may be missed or misinterpreted by subsequent algorithms.

4.3.2 Conservative vs. Liberal Merging

In the most basic approach possible one could attempt to simply merge all planar surfaces that fulfill certain criteria, such as being sufficiently close, coplanar etc. While this would certainly mend oversegmented patches, it would also produce unwanted mergers between surfaces that were previously correctly separated, e.g. the seat surfaces of two nearby chairs. The decision to merge two candidate surfaces should therefore be based on occlusion/visibility evidence in the range data. For the set of points observed from one view point, there are two possible policies for making such a decision, as illustrated in Fig. 4.4:

- *Conservative*: Two candidate surfaces are merged if and only if an object that is closer has been observed in the gap between the two.
- *Liberal*: Two candidate surfaces are merged whenever no object that is further away has been observed in the gap between the two.

Obviously, even a conservative policy still includes the possibility of a mismerger when a gap that exists in the real world is occluded, e.g. a door opening hidden behind a column. However, this is something that is impossible to explain from a single view point. One of the few approaches that interpret occlusions according to a policy is [32], which aims to reconstruct individual pixels in range images. This approach, however, quietly assumes that there always must be evidence for or against occlusion. In general this is not true, as range sensors may yield no measurements on black, transparent, or reflective surfaces or when objects are too far away. For all of the data sets segmented in the following, a conservative policy was adopted.

To assess the plausibility of occlusions, the *omnidirectional range image* (ORI) introduced in Chap. 2 is used, which projects all points perceived from a given view point onto a spherical depth map centered at that view point. The resulting image representation then allows for efficient assessment of occlusions by checking the depth values between boundary pixels using the Bresenham line algorithm [22, 32], where special care must be given to the circular nature of the azimuth a for panoramic scans. Fig. 4.5 shows the unwrapped and cropped ORI of the rotational scan from Fig. 2.16(c), which demonstrates how the foreground objects, such as the kitchen chair, align exactly with the background shadow.

4.3.3 Segment Features and Distance Measures

In order to identify candidate surfaces for merging, several of the segment features introduced in Chap. 3 must be computed for each of the segments. These are:

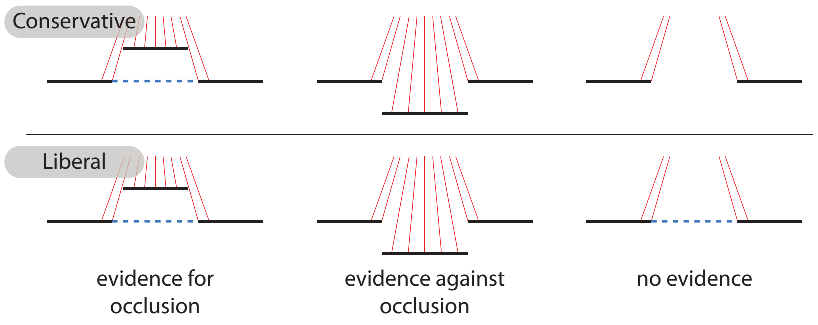


Figure 4.4: There are two possible policies for deciding whether a gap originates from an occlusion: With a conservative policy, surfaces are only merged (dashed line) if there is evidence for occlusion. In contrast, a liberal policy will merge surfaces unless there is evidence against occlusion.

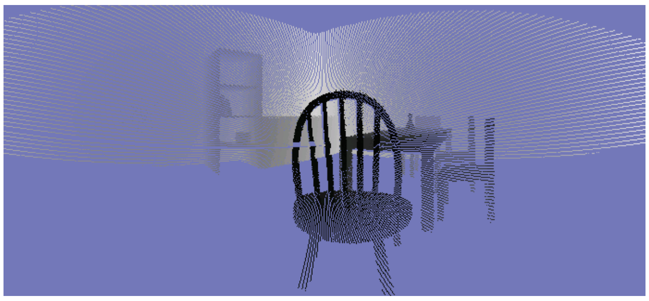


Figure 4.5: The omnidirectional range image (ORI) of the rotational scan from Fig. 2.16(c). Floor and ceiling points have been removed, depth is grayscale-colored, blue pixels indicate that no observation was made. The data points of the kitchen chair fall exactly into the gaps on the wall, which demonstrates that the ORI is well-suited to explain occlusions from a given view point.

- $\overline{\boldsymbol{p}}_j$: The centroid of all points in the segment
- $\boldsymbol{n}_j$: The normal vector of a plane fitted to all points in the segment
- $\sigma_{sv,j}$: The surface variation of all points in the segment

Given $\overline{\boldsymbol{p}}_j$ and $\boldsymbol{n}_j$ for each segment $\mathcal{S}_j$, the following distance measures between two segments $\mathcal{S}_j$ and $\mathcal{S}_k$ are defined:
Centroid distance between segments:

$$\rho(\mathcal{S}_j, \mathcal{S}_k) = \left\|\overline{\boldsymbol{p}}_j - \overline{\boldsymbol{p}}_k\right\|_2 \tag{4.4}$$

Angular distance between the segment normal vectors:

$$\rho_a(\mathcal{S}_j, \mathcal{S}_k) = \cos^{-1}(|\langle \boldsymbol{n}_j, \boldsymbol{n}_k \rangle|) \tag{4.5}$$

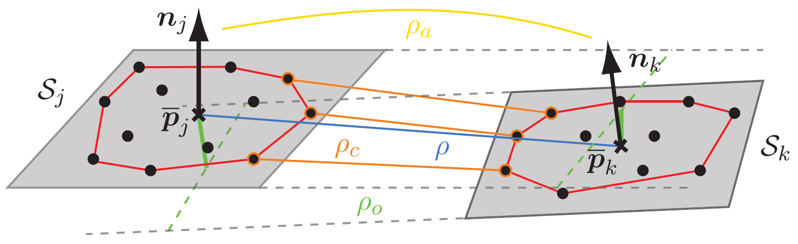


Figure 4.6: Distance measures for two neighboring segments $\mathcal{S}_j$ and $\mathcal{S}_k$: ρ is the Euclidean distance between the cluster centroids (blue). ρ_a is the absolute value of the inner product of the plane fit normal vectors (yellow). ρ_o is the orthogonal distance, i.e. the minimum of the distances from each centroid to the other plane (green). ρ_c is the coplanar distance, i.e. the minimum distance between pairs of boundary points (orange).

Orthogonal distance between planes, i.e. the minimum of the distances from each centroid to the other plane:

$$\rho_o(\mathcal{S}_j, \mathcal{S}_k) = \min\left(\left|\langle \boldsymbol{n}_j, \overline{\boldsymbol{p}}_k\rangle - \langle \boldsymbol{n}_j, \overline{\boldsymbol{p}}_j\rangle\right|, \left|\langle \boldsymbol{n}_k, \overline{\boldsymbol{p}}_j\rangle - \langle \boldsymbol{n}_k, \overline{\boldsymbol{p}}_k\rangle\right| \right) \tag{4.6}$$

Furthermore, for two segments $\mathcal{S}_j$ and $\mathcal{S}_k$ that are planar, sufficiently close and approximately coplanar, the *coplanar distance* is defined as the minimum of the distances between n_b pairs of boundary points $\left\{(\boldsymbol{p}_{j,1}, \boldsymbol{p}_{k,1}), \ldots, (\boldsymbol{p}_{j,n_b}, \boldsymbol{p}_{k,n_b})\right\}$:

$$\rho_c(\mathcal{S}_j, \mathcal{S}_k) = \min\left\{\left\|\boldsymbol{p}_{j,1} - \boldsymbol{p}_{k,1}\right\|_2, \ldots, \left\|\boldsymbol{p}_{j,n_b} - \boldsymbol{p}_{k,n_b}\right\|_2\right\} \tag{4.7}$$

Fig. 4.6 illustrates the meaning of the four distance measures.

4.3.4 Identifying Boundary Points

In order to compute the previously defined coplanar distance ρ_c and to test for occlusions along connecting lines, it is necessary to identify a set of suitable boundary points. In the context of surface reconstruction, connecting suitable pairs of boundary points is equivalent to adding edges and faces to a triangular mesh. In contrast, for the scenario considered here, the focus is on assessing the plausibility of a possible occlusion without the construction of a complete mesh. The question is then, how to efficiently find points on the boundary of a near-planar segment without computing a triangulated mesh. The convex hull of the projected points can be built without triangulation, but it will usually yield an overapproximation, meaning that many boundary points lie inside the hull and will not be selected. A better approximation could be obtained by 2D convection [28], however, this again requires a triangulation and is therefore not suitable.

Instead suitable points are chosen as follows:

- In each segment the $n_b/2$ points with minimal distance to the other segment's centroid are identified. This will yield points on the boundary close to the other segment.

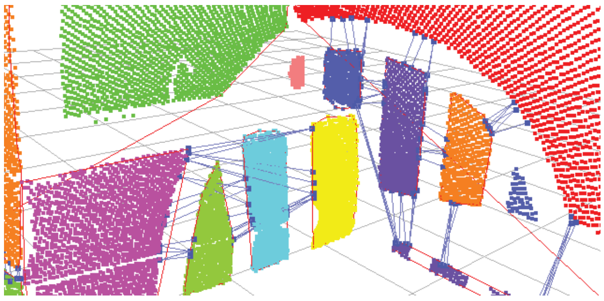


Figure 4.7: Automatically identified boundary points (square) for the oversegmented surfaces of the rotational scan from Fig. 2.16(c). The connecting lines pass through the regions that need to be checked for occlusion. The red lines show the convex hulls of all planar patches. For large patches, such as the right wall, the convex hull is clearly a bad approximation of the boundary. Note that two segments remain unconsidered because they are not planar enough.

- For each of the chosen points in one segment, the closest point in the other segment is identified, yielding a total of n_b pairs of points.

While this method does not make any guarantees in terms of geometric properties, such as a Delaunay triangulation or a convex hull, it yields direct connections between neighboring segments at essentially no additional computational cost. This is because the necessary kd-trees for accelerated retrieval of closest points have already been built during normal vector estimation. Fig. 4.7 illustrates the generated connections for the oversegmented wall and sink cabinet of the rotational scan from Fig. 2.16(c). Note that several patches of the wall actually lie partially inside the convex hull of the biggest patch; thus the proposed method yields much better pairs of boundary points than the use of convex hulls.

4.3.5 Merging the Patches

The tools defined in the previous sections provide the basis for merging surfaces that have been oversegmented because of occlusions. The algorithm described in the following is similar to the approach presented in [32]; however, the focus is not on pixel reconstruction in a range image, but on the improvement of the previously obtained segmentation result. Consequently, no exhaustive tracing of boundary lines is necessary, but rather the plausibility of occlusion is evaluated from the few boundary lines established in Sec. 4.3.4.

For a given environment, a total of four thresholds for the distance measures and segment features must be specified, which express a minimal set of constraints on the environment geometry, that aids the merging process without making any explicit model-specific assumptions. Once chosen, these parameters can stay fixed for any data set recorded in the same type of environment. The required thresholds are:

- $\sigma_{sv,\max}$: The maximum surface variation of a segment to be regarded as a plane
- $\rho_{a,\min}$: The maximum angle between the normals of two planar segments to regard them as coplanar

Algorithm 4.3: Merge Occluded Segments

```
Input: Segments $\mathcal{S}_1, \ldots, \mathcal{S}_{n_s}$
       An ORI for each view point in the scene
       Parameters $\sigma_{sv,\max}$, $\rho_{a,\min}$, $\rho_{o,\max}$, and $\rho_{c,\max}$.
Output: Updated segments $\mathcal{S}_1, \ldots, \mathcal{S}_{n_s}$
1  Calculate $\overline{\boldsymbol{p}}_i$, $\boldsymbol{n}_i$, and $\sigma_{sv,i}$ for all segments;
2  for $i = 1 : n_s$ do
3      if ($\sigma_{sv,i} > \sigma_{sv,\max}$) then Continue;
4      Find the $k_c$ segments with the nearest centroids;
5      for $j = 1 : k_c$ do
6          if ($\sigma_{sv,j} > \sigma_{sv,\max}$ or $\rho_a(\boldsymbol{S}_i, \boldsymbol{S}_i) \leq \rho_{a,\min}$ or $\rho_o(\mathcal{S}_j, \mathcal{S}_k) \geq \rho_{o,\max}$) then Continue;
7          Compute $n_b$ pairs of boundary points;
8          if ($\rho_c(\boldsymbol{S}_i, \boldsymbol{S}_n) < \rho_{c,\max}$) then
9              Merge $\mathcal{S}_i$ and $\mathcal{S}_j$;
10         else
11             for $l$=1:$n_b$ do
12                 if (Boundary points of pair $l$ belong to different ORIs) then Continue;
13                 Traverse line $l$ in ORI using Bresenham algorithm;
14                 if (majority of pixels occluded) then
15                     Count line as occluded;
16                 end
17             end
18             if (majority of lines occluded) then
19                 Merge $\mathcal{S}_i$ and $\mathcal{S}_j$;
20             end
21         end
22     end
23 end
24 return $\mathcal{S}_1, \ldots, \mathcal{S}_{n_s}$
```

- $\rho_{o,\max}$: The maximum orthogonal distance between two planar segments to regard them as coplanar

- $\rho_{c,\max}$: The maximum coplanar distance between two planar segments to regard them as belonging to the same segment

Given these thresholds and a scene possibly containing several registered point clouds, the merging process then proceeds as described in Alg. 4.3.

The rationale behind merging coplanar surfaces with sufficiently small ρ_c in line 9 is that small gaps which may originate from missing or misaligned scanlines should be merged automatically. If this is undesired, $\rho_{c,\max}$ can simply be set to 0. To evaluate whether pixels of a given connecting line indicate an occlusion, the interpolated depth between the line end points is compared to the actual depth found in the ORI. When the depth in the ORI is less than the interpolated depth this is counted as evidence for occlusion. The two majority votes for pixels and lines may be replaced by more sophisticated percentage values; however, experimental variation showed that using the simple majority (50%) works well and avoids the introduction of further parameters. In summary, Alg. 4.3 merges planar

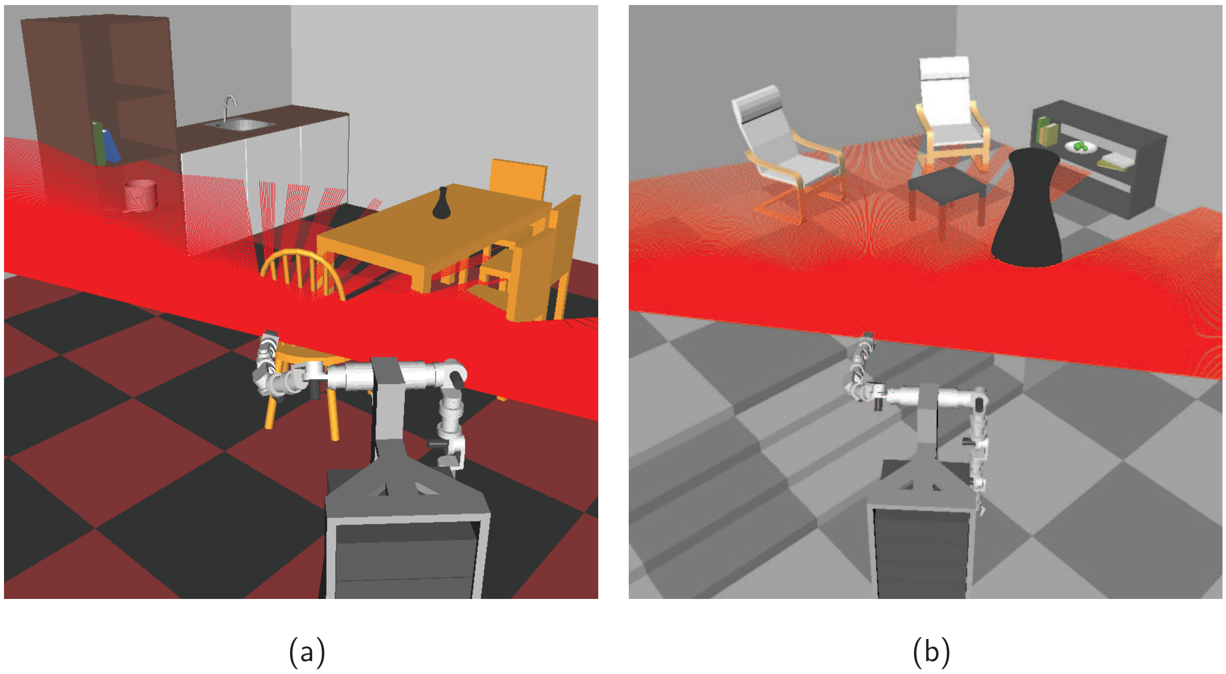


Figure 4.8: The simulation scenarios: (a) *Kitchen* environment (b) *Living Room* environment

oversegmented surface patches if there is evidence that the gap between them originates from an occlusion.

4.4 Evaluation Study

The segmentation framework presented in Secs. 4.2 and 4.3 was evaluated on several simulated and real point clouds. Implementation and performance of the framework are discussed in this section.

4.4.1 Implementation Details and Parameters

The described segmentation toolchain was implemented in C++ on a desktop PC with 2.5 GHz and 2GB RAM. As before, nearest neighbor queries were accelerated using the ANN library [196], and singular value decompositions and matrix operations were carried out with the GNU Scientific Library in conjunction with CBLAS [176]. The convex hulls shown in the results were computed with CGAL [175]. The simulated data sets were obtained using the simulator introduced in Sec. 2.4.3, and the real points clouds were acquired with the actuated Hokuyo UTM-30LX introduced in Sec. 2.5.

For *all* data sets one and the same set of parameters was used for segmentation: k=50, $\alpha_{\max}$=5°, $\rho_{\max}$=0.30m, $\sigma_{sv,\max}$=0.15, $\rho_{a,\min}$=15°, $\rho_{o,\max}$=0.05m, $\rho_{c,\max}$=0.20m, n_b=6. Segments with less than $n_{\min}$=30 points were removed from the visualization. The fact that one set of parameters yielded good segmentation results for both simulated and real range data underlines that the framework is robustly applicable to data acquired with comparable scanning characteristics.

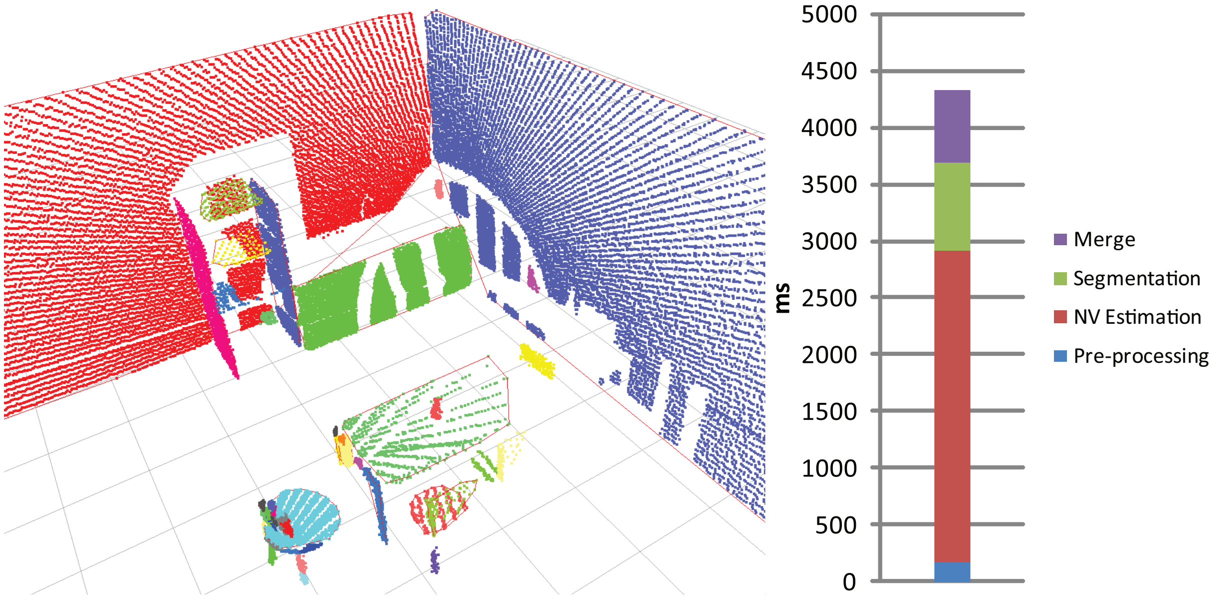


Figure 4.9: Segmentation result and runtime for the rotational scan of the *Kitchen* environment.

4.4.2 Simulation Results

Simulations of different scans were carried out in two domestic environments: the *Kitchen* environment shown in Fig. 4.8(a) contains a tall cupboard, a sink cabinet and a kitchen table with four chairs, one of them with a lattice backrest that causes disjoint areas of occlusion. The *Living Room* scenario contains a typical setup of furniture, namely two armchairs, a small table, a sideboard and a large nonconvex vase. The objects are positioned in an elevated area, that is reached by three steps, also in the field of view of the robot.

The scans were simulated with the sensor characteristics of the Hokuyo UTM-30LX, a scan angle of α_{FoV}=180°, angular resolution of α_r=0.25°, standard deviation of σ=15mm and frequency f_s=40Hz. All scans were recorded within t_s=4s, resulting in $n = t_s \cdot f_s \cdot \alpha_{FoV}/\alpha_r = 115,200$ noisy data points.

Fig. 4.9 shows the segmentation result and the time needed by each of the algorithm steps for the rotational scan of the *Kitchen* environment; the exact numbers are provided in Tab. 4.1. It can be seen that all major surfaces have been recovered correctly: the cupboard surfaces, the table surface in spite of inhomogeneous density, the right chair seat surface even though it is only covered by two scanlines, and the wall and sink cabinet despite the occlusion caused by the lattice backrest. The total processing time is slightly above the scan time of 4 seconds.

Similar results are obtained for the rotational scan of the *Living Room* data set, see Fig. 4.10. Apart from the correctly segmented wall, table and armchair surfaces, it can be seen that the vase is segmented as one object, which demonstrates the ability to extract nonplanar surfaces with smooth curvature change. The total processing time is slightly below the scan time of 4 seconds.

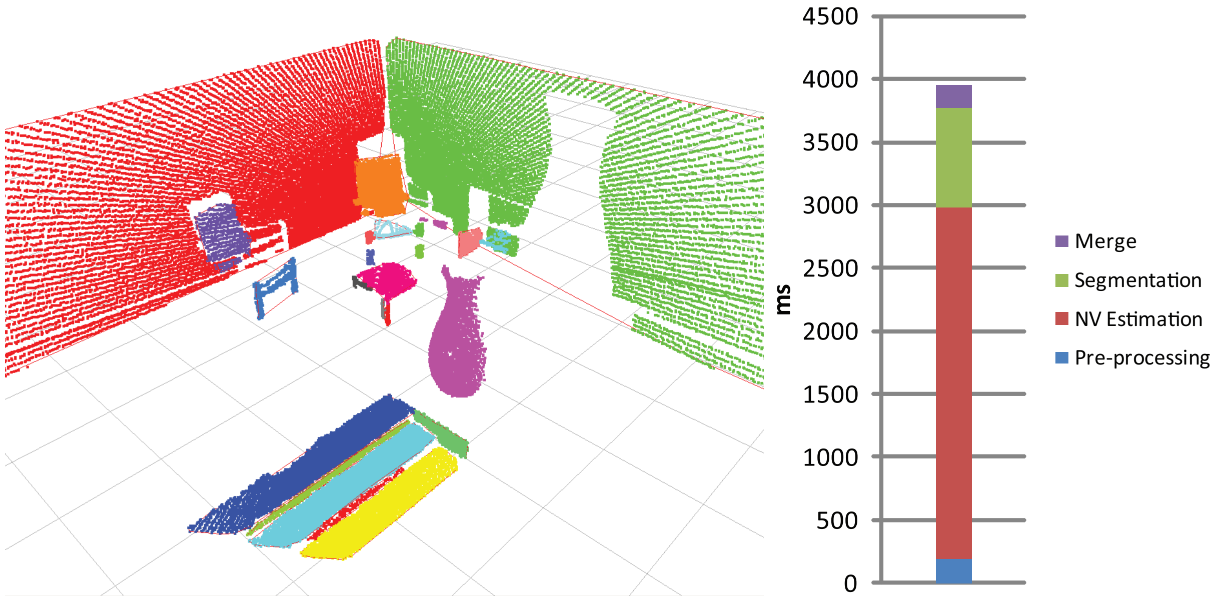


Figure 4.10: Segmentation result and runtime for the rotational scan of the *Living Room* environment.

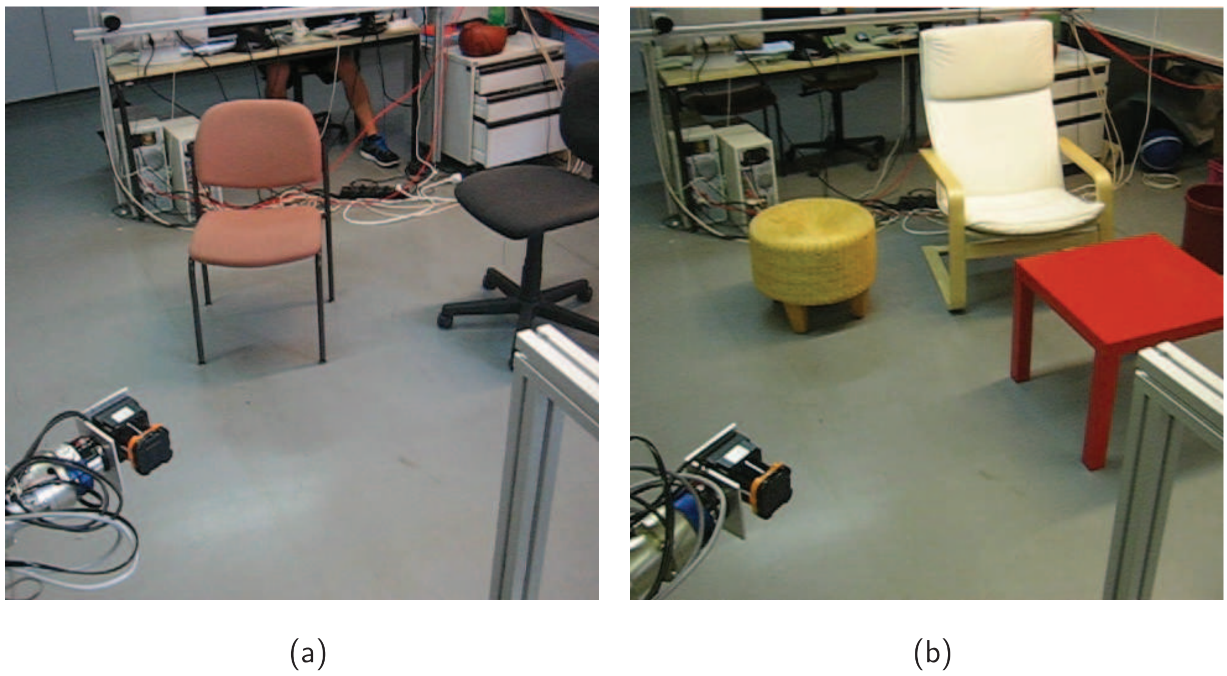


Figure 4.11: The experimental scenarios: (a) *Two Chairs* (b) *Seat, Armchair, Table* (SAT)

Dataset	n	t_s	n_{pp}	t_{pp}	t_{nve}	t_{seg}	t_m	n_m	n_s
Kitchen R	115,200	4.0s	54,828	0.17s	2.74s	0.78s	0.64s	22	59
Living Room R	115,200	4.0s	56,014	0.19s	2.80s	0.78s	0.19s	2	32
Two Chairs H	198,720	6.9s	39,203	0.39s	1.92s	0.52s	0.30s	12	91
Two Chairs V	239,040	8.2s	118,518	0.39s	5.97s	1.81s	1.53s	26	234
Two Chairs R	113,760	3.9s	53,747	0.19s	2.59s	0.72s	0.45s	27	139
SAT H	296,640	10.2s	61,448	0.55s	3.00s	0.83s	1.23s	29	145
SAT V	297,360	10.3s	146,331	0.45s	7.47s	2.27s	1.33s	115	262
SAT R	158,400	5.5s	72,158	0.25s	3.51s	0.97s	0.39s	21	180

Table 4.1: Segmentation Results: H = horizontal scan, V = vertical scan, R = rotational scan, n = number of points, t_s scan time, n_{pp} = number of points that remain after pre-processing, t_{pp} = pre-processing time, t_{nve} = normal vector estimation time, t_{seg} = segmentation time, t_m = merging time, n_m number of merged segments, n_s = final number of segments.

4.4.3 Experimental Results

With the hardware setup described in Sec. 4.4.1, two distinct scenes were scanned with different trajectories. The first consists of two simple chairs placed in an otherwise cluttered laboratory environment, see Fig. 4.11(a). For the rotational and vertical scan the arm was commanded to keep the view point invariant using an inverse kinematics scheme; for the horizontal scan it was allowed to rotate about its elbow. The trajectory for each type of scan is indicated by the arrows in the top row of Fig. 4.12, the resulting scans can be seen in the middle row, and the segmentation result in the bottom row. The point cloud sizes and processing times are provided in Tab. 4.1. For all three types of scan the major segments, including the seat surfaces, the walls, and laboratory tables have correctly been extracted from the point cloud, with several segments separated by occlusions correctly merged. Segmentation time is roughly on the order of the scan time, with the bulk of the processing time spent estimating normal vectors.

The second scenario, shown in Fig. 4.11(b) consists of a seat, an armchair and a table. Due to the similarity of results only the segmentation of the rotational scan is shown in Fig. 4.13. As can be seen, curved and planar surfaces of the furniture are extracted correctly from the point cloud and the total processing time is slightly below the scan time. The processing times and point cloud sizes of all three scans are listed in Tab. 4.1.

4.5 Discussion

Various aspects of the presented results merit further discussion. In all of the segmentation results, the bulk of the processing time is spent on the estimation of normal vectors, not on the actual segmentation. Unfortunately, the segmentation result is highly dependent on the accurate estimation of normals, which has linear runtime complexity in the number

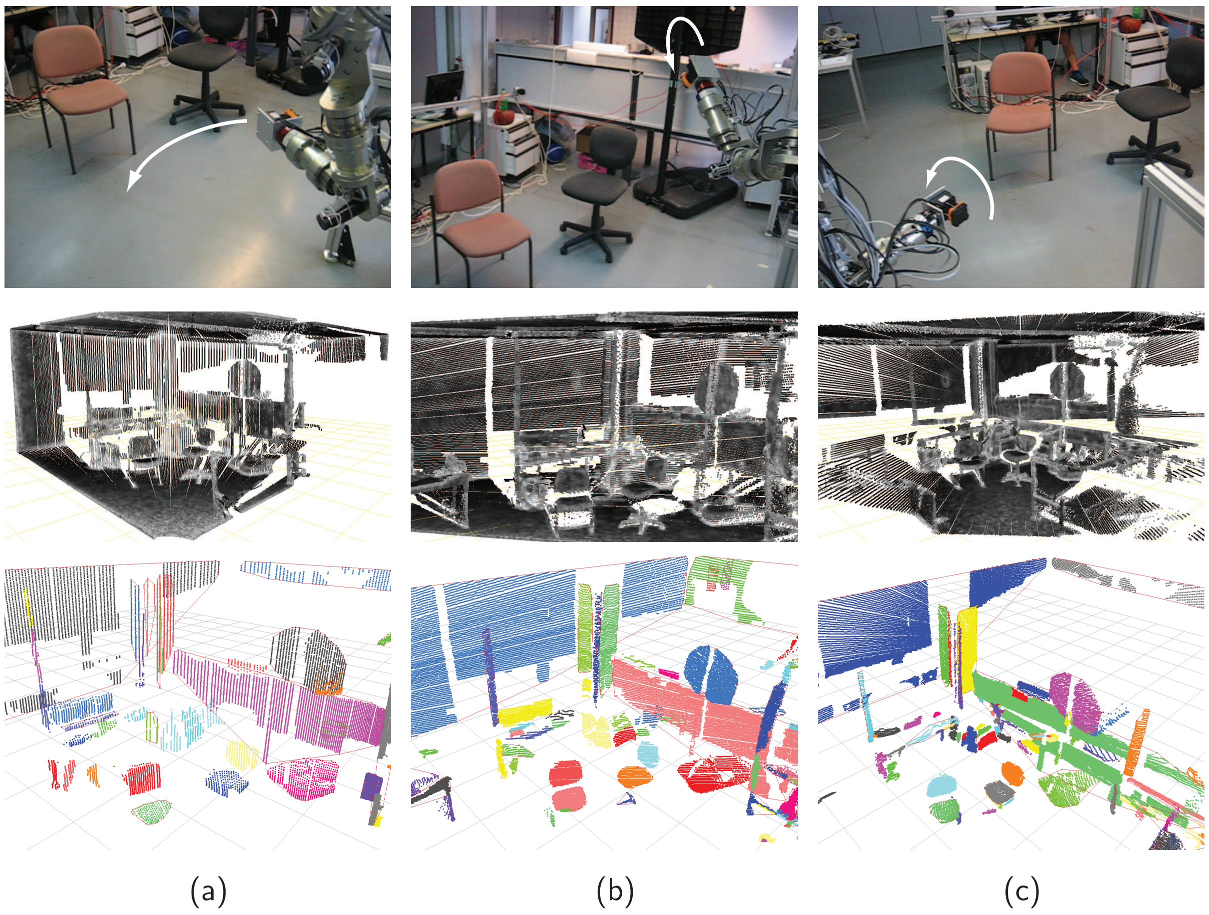


Figure 4.12: Three distinct scanning trajectories were performed with the scanning setup introduced in Sec. 2.5: (a) horizontal scan (b) vertical scan (c) rotational scan. The top row shows a photograph of each scan pose, with the scan trajectory indicated by a white arrow. The middle row shows the resulting point cloud with grayscale-colored curvature values σ_{sv} (k=30) for better visibility. The bottom row shows the segmentation result: the major surface primitives, such as the chair surfaces, the walls and laboratory tables are correctly segmented for all three types of scan in the presence of noise, variable density and occlusions (note that the wall behind the column is correctly mended into one segment).

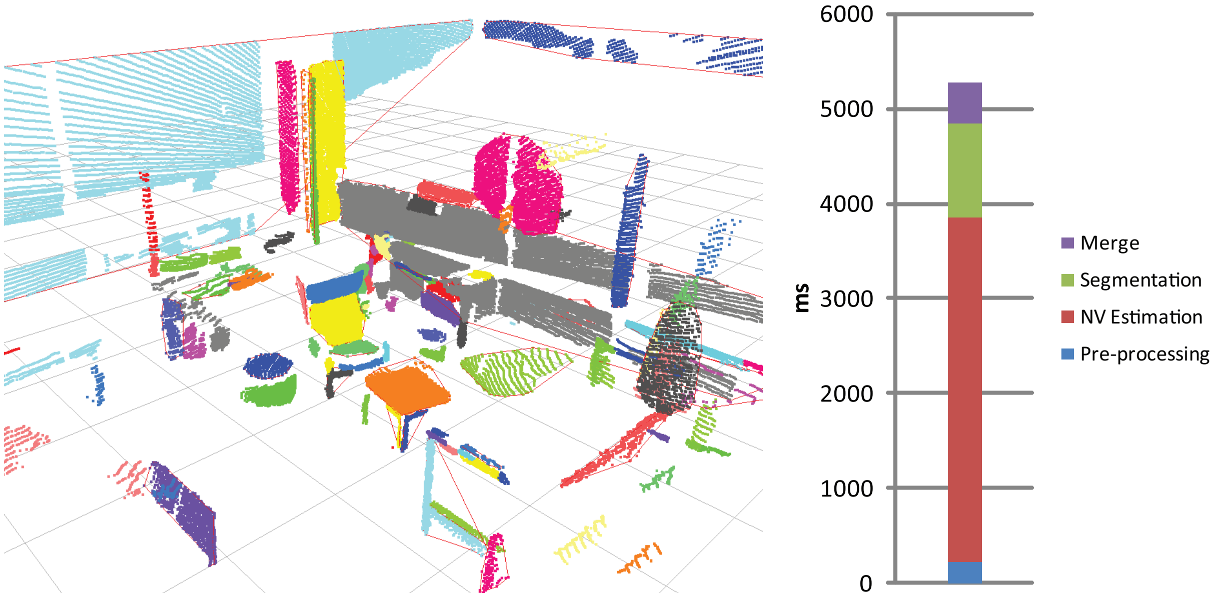


Figure 4.13: Segmentation result and runtime for the rotational scan of the *Seat, Armchair, Table* environment.

of nearest neighbors, k. As outlined in Sec. 3.2, k does not have to be very large to attain good estimates. However, even for moderate sizes of k, normal vector estimation will quickly become the computational bottle neck in the segmentation toolchain. One possible solution is to parallelize the estimation process. Since there are no interdependencies for the SVD, the normal computation is embarrassingly parallel and normals can be estimated completely independently by n_p processes each computing n/n_p normals.

The majority of the examined segmented point clouds do not exhibit sufficient density to allow for detailed surface reconstruction or the like. It should, however, be noted that the toolchain is directly applicable to dense point clouds with several million points. It is thus also useful in scenarios where processing time does not play a crucial role and both acquisition and segmentation may take more than ten or twenty seconds. For the main scenario considered in this thesis, namely the acquisition of data with a quick sensor sweep, the important fact is that the segmentation toolchain can handle relatively sparse data sets. In the context of object recognition, the extracted segments provide a meaningful abstraction of sparsely and inhomogeneously sampled objects. As an example consider the object-oriented bounding boxes (OOBBs) of the segmented surfaces in the rotational scan of the *Kitchen* environment, which can be computed directly from the SVD of the points in each segment, see Sec. 3.5. Fig. 4.14 shows OOBBs of several segments in the *Kitchen* scene and the *Seat, Armchair, Table* scene. The position, orientation and extent of the bounding boxes provide informative features e.g. for part-based classification, as shall be seen in the following chapter.

In its current form, one limitation of the proposed segmentation framework is that it requires groups of points to have been observed from a single view point to explain occlusions. This assumption holds for many scanning setups, especially those that yield range images as output. However, as explained in Chap. 2, for actuated 2D laser range finders

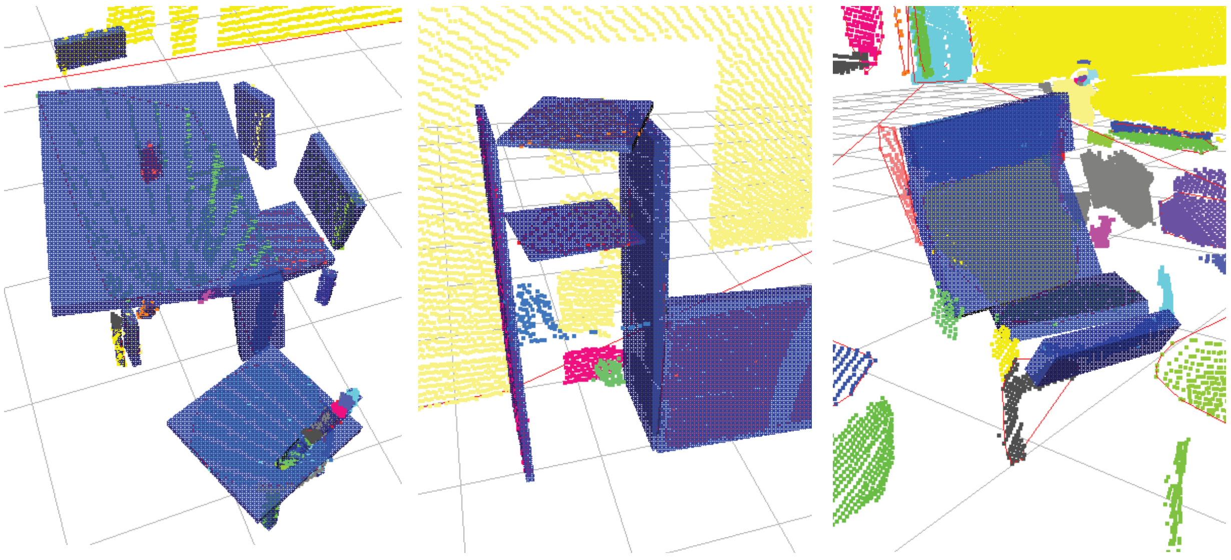

(a) (b) (c)

Figure 4.14: The position, extent, and orientation of object-oriented bounding boxes of segments provide informative features for classification: (a) Segments of the table and chairs in the *Kitchen* scene. (b) Segments of the cupboard and sink cabinet in the *Kitchen* scene. (c) Segments of the armchair in the *Seat, Armchair, Table* scene.

there are a plethora of possible trajectories that do not conform to this assumption. When each scanline is recorded from a different view point, explanation of occlusions is not supported by the current framework, but actually might still be possible from considerations of the view point trajectory.

5 Part-based Object Recognition

Trinity, learning how to fly a helicopter
The Matrix, © 1999 Warner Bros.

Summary

Building on the surface segment representation developed in the previous chapter, this chapter presents a framework for recognizing objects in 3D range data. A part-based object representation is learned from a set of simulated or real 3D scans with minimal supervision. The abstracted parts are regarded as dictionary words in the spirit of a 'bag of words' approach. To robustify the classification, spatial relations between parts are introduced as additional words. Both simulation and experimental results show that high recognition rates can be achieved with a simple maximum a posteriori classifier. Algorithmic details and possible improvements of the approach are discussed.

As motivated in Chap. 1, a central skill for intelligent behavior of autonomous mobile robots is the understanding of 3D environments, which implies the ability to recognize objects in the robot's surroundings. The algorithms presented throughout the previous chapters provide the necessary tools to abstract spatial 3D data perceived by range sensors and set the stage for the classification framework presented in this chapter. The framework provides important facilities to learn and recognize objects from few examples. Because it abstracts point clouds to a part-based representation, it learns to recognize objects on a structural level, rather then on a level of feature distributions over points. Furthermore, a probabilistic formulation allows for a trade-off between high accuracy and high precision by means of a detection threshold. In the implementation, the part primitives are learned from the segments obtained from the toolchain presented in the previous chapter, because this representation was found to work very well. The framework presented in this chapter, however, is generic, meaning that any method that is able to extract and detect parts in 3D range data can be 'plugged in' to build the vocabulary and recognize previously learned parts.

The chapter is structured as follows: In Sec. 5.1 related work from different areas is presented. Sec. 5.2 introduces the construction of a vocabulary of common parts and Sec. 5.3 explains how to use the so-obtained dictionary to detect parts and relations in new views. In Sec. 5.4, which is at the heart of the recognition framework, the detected parts are integrated into a histogram representation and the classification problem is cast into a Bayesian formulation. Sec. 5.5 describes how to retrieve the classification result from the posterior probabilities and summarizes the classification procedure. Secs. 5.6 and 5.7 demonstrate the applicabiliy of the classification framework on both simulated and experimental data, respectively. Sec. 5.8 discusses various issues of the recognition approach along with possible improvements.

5.1 Related Work

The work presented in this chapter is related to several areas. The *bag of words* approach was originally conceived and successfully applied in the machine learning community for purposes of document classification and topic modeling [19, 60]. The key idea is to neglect context and assume that any word in a text document may appear independently of other words; thus the document can be regarded as a loose collection of words, or a 'bag of words'. This assumption, which essentially states independence of features, was adopted and successfully applied to classification problems in computer vision, where words are represented by collections of visual features or image regions [135, 136, 141]. In the context of 3D range data, a recent paper applies latent Dirichlet allocation, a powerful generative probabilistic model that builds on the bag of words assumption, to the problem of discovering object classes from 3D range data in an unsupervised fashion [36]. The approach is, however, not part-based, as the vocabulary is built by discretizing local shape features obtained from modified spin images for every point. In contrast, the approach presented in this chapter uses parts and relations between them as dictionary words.

Part-based approaches to object recognition work with some sort of structural description of the objects. As opposed to mere matching of feature vectors for entire objects, they typically abstract objects to consist of a set of distinct parts with topological and/or quantitative relations between them. In the past, much effort has been invested to describe objects by graph models and perform object recognition by matching the (possibly incomplete) graph of a new view to the graphs of known objects [26, 27, 59]. E.g. a chair could be described by a seat surface, four legs and a backrest, where the legs and the backrest are spatially connected to the seat surface. While graph matching has been successfully applied in some domains, it comes with a number of issues, among which are the NP-completeness of inexact graph matching and the non-deterministic outcomes of randomized optimization performed in the matching phase [57].

Early seminal work on inferring object and part structure from surface information was conducted by Fisher [40] as well as Raja and Jain [116]. Due to the limitations of both sensing hardware and processing power at the time, both works stay confined to scenarios involving extremely simple geometric primitives. More recently, part detection from range data has been demonstrated by Gaechter et al. [43], who present the efficient agglomeration of noisy point cloud data obtained from a ToF camera to a bounding box part representation using a particle filter.

Another interesting part-based approach is the use of *probabilistic geometric grammars* [87, 133], which capture the structural variability of objects by learning rules that describe the parts and relations encountered in object examples. While this rather recent methodology yields promising recognition results and is able to learn rules from examples, it requires a pre-specified set of models and parts.

5.2 Learning Common Parts and Spatial Relations

5.2.1 Definitions

Several entities will be distinguished in the following: A *scan* denotes a surface sampling of a 3D environment obtained from one or several view points by means of a range sensor. The environment is assumed to consist of *objects* as well as *background*, where objects refer to anything that is of interest for recognition, e.g. furniture, and background denotes the rest, e.g. walls. *Segments* refer to homogeneous surfaces extracted from a scan by means of a segmentation algorithm such as that presented in Chap. 4. A *view* denotes a set of segments that contain at most one object, that is, it contains either one object or background. *Parts* denote segments that are commonly encountered across views of the same object or even across views of different objects. Wherever referenced, the global x- and y-axes are assumed to be aligned with the ground plane and the z-axis is supposed to point upward.

5.2.2 Used Part Features

As explained in Sec. 4.5, the segmented point clouds yield an abstract representation of segments, where each segment can be described by a characteristic low-dimensional feature vector, containing the features introduced in Sec. 3.5. In the current study, $n_f = 5$ part features were used:

- Mean z-coordinate $\bar{p}_z$ (in m)
- Angle between normal vector and z-axis $\rho_a(\boldsymbol{n}, [0\ 0\ 1]^T)$ (in $[0, 1]$)
- Surface variation σ_{sv} (in $[0, 1]$)
- Planar aspect ratio σ_{ar} (in $[0, 1]$)
- Object-oriented bounding box volume V_{oo} (in m^3)

Both σ_{sv} and σ_{ar} as well as the object-oriented bounding box are computed from a PCA of the points in each segment, see Sec. 3.5. It should be noted that the first two features are 'absolute', that is, they are not independent of object pitch and height, since they are defined with respect to the global coordinate frame. While this limits their expressive power for a fallen furniture object, height and pitch of part surfaces are highly discriminant for any upright object in a home environment, such as tables, chairs etc. The use of alternative features is discussed briefly in Sec. 5.8.

Features 1 and 5 are not naturally bounded and are therefore normalized by the maximum value encountered in the training set, i.e. by the maximum height and the maximum volume. The dimensionless feature vector

$$\boldsymbol{f}^T = [f_1\ f_2\ f_3\ f_4\ f_5], \quad f_i \in [0, 1] \tag{5.1}$$

is then computed for all segments of all views of all objects.

5.2.3 Clustering Parts

Given the feature vector of each part, the next step is to find groups of similar parts observed across different views. To keep the required user input minimal, it is desirable to avoid labeling at the part level; the natural consequence of this is to perform clustering in the part feature space to obtain groups of similar parts in an unsupervised fashion. In the selection of a suitable clustering algorithm the following requirements were considered: Firstly, since the number of part classes is unknown, algorithms such as k-means, EM etc., that require a fixed number of clusters, were not among the preferred choices. The use of an efficient k-means variation that guesses the number of clusters is discussed briefly in Sec. 5.8; for the considered scenarios it did not yield equally satisfying results as the method described in the following. A second requirement for the clustering algorithm was that it should operate incrementally, i.e. the incorporation of new views should ideally cause constant overhead for each new part and not require recomputing the clusters from all parts. With these two requirements in mind, the parts were clustered using a greedy, agglomerative strategy that operates using a distance threshold. The dissimilarity metric used to compare two segments described by $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ is

$$\rho(\boldsymbol{f}_i, \boldsymbol{f}_j) = \frac{1}{n_f} \left\| \langle \boldsymbol{w}, \boldsymbol{f}_i - \boldsymbol{f}_j \rangle \right\|_1, \tag{5.2}$$

Algorithm 5.1: Greedy Means Clustering

```
   Input: $n$ data items $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$
          Distance metric $\rho(\boldsymbol{p}_i, \boldsymbol{p}_j)$
          Threshold $\rho_{\max}$
   Output: Clusters $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$
           Means $\mathcal{M} = \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_m\}$
           Covariances $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m\}$
1  Initialize $\mathcal{C} = \{\{\boldsymbol{p}_1\}, \ldots, \{\boldsymbol{p}_n\}\}$, $\mathcal{M} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$;
2  for $i = 1 : |\mathcal{C}|$ do
3      for $j = i + 1 : |\mathcal{C}|$ do
4          if $(\rho(\boldsymbol{m}_i, \boldsymbol{m}_j) \le \rho_{\max})$ then
5              $\mathcal{C}_i = \mathcal{C}_i \cup \mathcal{C}_j$;
6              $\boldsymbol{m}_i = \frac{|\mathcal{C}_i|}{|\mathcal{C}_i|+|\mathcal{C}_j|}\boldsymbol{m}_i + \frac{|\mathcal{C}_j|}{|\mathcal{C}_i|+|\mathcal{C}_j|}\boldsymbol{m}_j$;
7              Remove $\mathcal{C}_j$ and $\boldsymbol{m}_j$, decrement $j$;
8          end
9      end
10 end
11 Calculate covariances $\boldsymbol{\Sigma}_i = \frac{1}{|\mathcal{C}_i|}\sum_{j=1}^{|\mathcal{C}_i|}(\boldsymbol{p}_{ij} - \boldsymbol{m}_i)(\boldsymbol{p}_{ij} - \boldsymbol{m}_i)^T$;
12 return $\mathcal{C}$,$\mathcal{M}$,$\boldsymbol{\Sigma}$
```

where $\boldsymbol{w}^T = [w_1\ w_2\ w_3\ w_4\ w_5]$, $\|\boldsymbol{w}\|_1 = 1$, is a weight vector that can be used to emphasize individual features. Since each feature is normalized, the clustering threshold $\rho_{\max}$ can be interpreted as the sum of the allowed variation of each feature (in %). E.g. a value of $\rho_{\max} = 0.1$ would allow for a 10% variation in each feature.

The clustering algorithm is described in Alg. 5.1. Essentially, it starts out with each part as a cluster and then agglomeratively merges parts whose feature vector is within the distance threshold from the current cluster mean. It is greedy because a different order of the input can result in a (slightly) different clustering. Its two main advantages are that all properties, in specific mean and covariance of the clusters, can be updated incrementally and that the thresholding parameter gives fine-grained control over the level of abstraction. This means that, in contrast to clustering algorithms that require the specification of the number of clusters, once a threshold with a suitable level of abstraction has been chosen, the algorithm can be expected to yield meaningful results when adding new objects with previously unseen parts. Note that in Alg. 5.1 the covariances are computed at the end; they can, however, be updated incrementally by maintaining the scatter matrix of each cluster [35]. When the clustering algorithm terminates, all clusters with less than $n_{\min}$ parts are pruned to obtain a compact and meaningful dictionary.

5.2.4 Extracting Part Relations

The methodology introduced so far yields a vocabulary of common parts encountered across examplary views of different objects. As initially stated, this is in analogy to approaches in 2D computer vision that use image regions or features as vocabulary words [135, 136, 141]. However, in contrast to image-based recognition, 3D range data offers a convenient

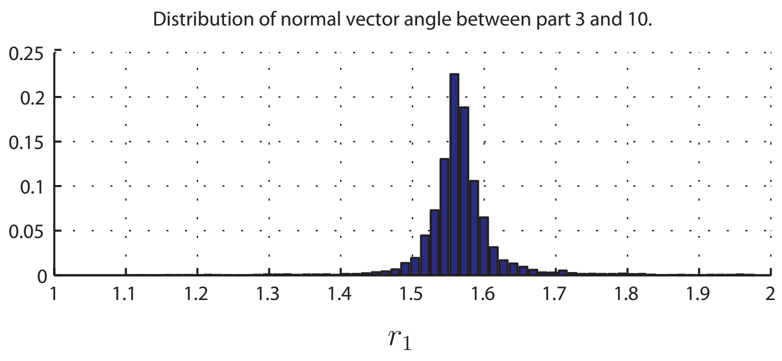


Figure 5.1: Spatial part relations are extracted as dictionary words from the histograms of relations between pairs of parts that have occurred together. This example shows the histogram of the angle between normal vectors of two characteristic parts. Clearly, there is a characteristic angle centered at $r_1 = 1.56$ which is observed in more than 20% of all cases in which parts 3 and 10 have been observed together.

advantage: since the sensed data is truly 3D and the segments are placed within a global coordinate frame, spatial relations between parts can be used as characteristic quantities describing the objects of interest. In 2D computer vision this is usually not possible as the relations between detected 2D features are subject to unknown scaling and projection in the image plane. In the following, relations between the detected parts are used as dictionary words to further robustify and disambiguate the classification. Two types of relation are considered:

- $r_1(a, b) = \cos^{-1}(\langle \boldsymbol{n}_a, \boldsymbol{n}_b \rangle)$: the angle spanned by the normal vectors of parts a and b
- $r_2(a, b) = \|\overline{\boldsymbol{p}}_a - \overline{\boldsymbol{p}}_b\|_2$: the Euclidean distance between the centroids of parts a and b

Since both types of relation are symmetric, in theory there are $n(n + 1)/2$ possible relations to consider for n part classes. This, however, would yield an extremely large dictionary with many meaningless relations that occur infrequently. To extract only 'relevant' relations, firstly only pairs of parts that were observed together in one view are considered. In other words, since the leg of a table and the backrest of a chair will never be observed together in the view of a single furniture object, it is not necessary to learn their relation as a dictionary word. Secondly, to further condense the set of considered relations, an extraction is performed as follows: The histograms of each relation are examined for each pair of parts that have occurred together in at least one view. If the peak of a given relation histogram exceeds a critical 'mass', the relation is considered relevant. For each relevant relation, sample mean $\overline{r}_k(a, b)$ and variance $\sigma_k^2(a, b)$ are computed. The extraction of relevant relations with such a simple method is feasible, because for all of the considered datasets, the histograms of both the angle between normals and the distance between centroids were observed to be unimodal. Fig. 5.1 shows an example for the normal vector relation r_1, i.e. the angle between the normal vectors of two parts.

5.3 Detecting Learned Parts and Relations

Given the extracted characteristic parts and relations, the question addressed in this section is how to classify new segments as belonging to learned part classes and the geometric relations between pairs of new segments as corresponding to a learned relation.

5.3.1 Part Detection

Part detection aims to determine whether a new segment belongs to one, several or none of the previously learned part classes. This is accomplished as follows:

Given the feature mean $\overline{\boldsymbol{f}}_i$ and covariance $\boldsymbol{\Sigma}_i$ of each cluster, and the feature vector $\boldsymbol{f}_j$ of the new segment, the Mahalanobis distance between the new segment and each cluster is computed as

$$\delta(\overline{\boldsymbol{f}}_i, \boldsymbol{f}_j) = \sqrt{\left(\overline{\boldsymbol{f}}_i - \boldsymbol{f}_j\right)^T \boldsymbol{\Sigma}_i^{-1} \left(\overline{\boldsymbol{f}}_i - \boldsymbol{f}_j\right)}, \tag{5.3}$$

see Appendix A.4. Segment j will be classified to belong to part cluster i if its Mahalanobis distance is within a threshold δ_p. Assuming Gaussianity of the features, the Mahalanobis distance of points, which contains the sum of squared feature entries, is known to follow a χ^2 distribution and the threshold can be looked up from a χ^2 table for a desired inlier percentage and n_f degrees of freedom. This means that, if multivariate normal distributions generate the d-dimensional features in each cluster, $\delta_p(p, d)$ states that part j will be regarded as belonging to cluster i if statistically it falls into the p percent of feature vectors generated by that cluster. The threshold then provides a conceptually well-founded and easily interpretable value for classifying parts.

It should be noted that, in contrast to nearest neighbor or maximum likelihood-based approaches, which have to assign a segment to exactly one part class, here the segment may belong to no part class at all. This rejection of unknown segments at the part level greatly robustifies the overall classification at the object level and can be regarded as one of the strong points of part-based approaches. It is also perfectly acceptable that parts are classified as belonging to several clusters. Essentially, the goal is to find the most discriminative setting of $\rho_{\max}$ and δ_p; too many and very narrow clusters will mean that almost no parts are detected, too few and very wide clusters will mean that the same parts are detected in all the objects.

5.3.2 Relation Detection

In analogy to the part classes, the detection of relation $r_k(a, b)$ between detected parts a and b in a new view is performed by testing whether the normalized distance

$$\delta\left(\overline{r}_k(a, b), r_k(a, b)\right) = \frac{|\overline{r}_k(a, b) - r_k(a, b)|}{\sigma_k(a, b)} \tag{5.4}$$

falls within a threshold δ_r, which can use the same inlier percentage as the threshold used for detecting parts. This measure is the one-dimensional version of the Mahalanobis distance and is motivated by the exact same reasoning as the part detection: if the distribution of relations such as normal vector angle or centroid distance is roughly Gaussian, see Fig. 5.1, the relation is detected when it falls into the p percent of samples generated by the distribution.

5.3.3 Gaussianity

The statistical justification for choosing δ_p and δ_r only holds if Gaussianity can be verified. To this end, each of the clusters obtained from the datasets presented hereafter, was examined with both the Anderson-Darling test [5] applied to the individual feature dimensions as well as a multivariate normality test applied to the entire feature vector [143] in the statistical computing language R. With a confidence level of 95%, Gaussianity could not be verified for all of the clusters. The non-Gaussianity of some clusters is, however, explicable when considering the nature of features: clusters are agglomerated by allowing for a certain percentage of variation in a given feature. If e.g. chair legs are only partially covered with scan points at different heights across different scans, this would yield segments that exhibit a Gaussian distribution in all features except feature f_1, the mean z-coordinate of points, which would have a roughly uniform distribution over a certain height range. The resulting cluster will not pass the rather strict statistical tests; however, it might still be approximated well by a Gaussian distribution with large sample variance in f_1.

At any rate, the Mahalanobis distance is independent of a specific distribution. Therefore it can be used to classify parts using a threshold δ_p, regardless of whether the underlying data is strictly Gaussian. The only drawback is that the threshold must be interpreted as a multiple of the sample variance along the principal components of a cluster; it cannot be interpreted as a bound on how many percent of points statistically lie within that threshold. However, knowing that Alg. 5.1 will yield 'well-formed' clusters, i.e. points will actually lie close to the mean of the cluster, it may actually still be helpful to choose δ_p from a χ^2 table, even though the clusters may only 'roughly' be Gaussian.

5.4 Histograms and Bayesian Formulation

The described detectors can be used to count the occurrences of parts and relations in histograms, as is common in bag of words-based approaches. For classification, the histogram of a new view can then be compared to histograms of previously seen objects, using measures such as the Kullback-Leibler divergence [83], the intersection measurement [142], the χ^2 divergence [12] or simply the scalar product of histogram vectors [136]. Different studies have reported different advantages of each of these measures; however, all of them produce only a ranking of similarity to known classes. As has been noted in [125], rather than comparing histograms using one of these unbounded measures, a preferable approach is to use a Bayesian formulation for comparing histograms. This yields normalized posterior probabilities that are - at least to some degree - interpretable in the sense of likelihood. In the following, a Bayesian formulation using histograms over detected parts and part relations is developed.

5.4.1 Histograms

The dictionary is assumed to consist of a total of n words, n_p of which correspond to parts, n_{r_1} to part relations of type 1, and n_{r_2} to part relations of type 2. The histogram of view V is a vector

$$\boldsymbol{h}_v^T = [\boldsymbol{h}_{v,p}^T \quad \boldsymbol{h}_{v,r_1}^T \quad \boldsymbol{h}_{v,r_2}^T] \tag{5.5}$$

where the n_p entries of $\boldsymbol{h}_{v,p}$ represent the number of times the respective part was detected in view V and the n_{r_k} entries of $\boldsymbol{h}_{v,r_k}$ the number of times the respective part relation was observed. Given n_v views of object O, the vector

$$\overline{\boldsymbol{h}}_{p,o} = \frac{1}{\|\sum_{i=1}^{n_v} \boldsymbol{h}_{i,p}\|_1} \sum_{i=1}^{n_v} \boldsymbol{h}_{i,p} \tag{5.6}$$

denotes the *mean part histogram* of object O, where the sum of all part histograms of that object is simply normalized by the total number of detected parts. Additionally, it is useful to define a *normalized relation histogram* for relation r_k,

$$\overline{\boldsymbol{h}}_{r_k,o} = \left[\frac{h_{r_k,1}}{n_{ab,1}} \quad \cdots \quad \frac{h_{r_k,n_{r_k}}}{n_{ab,n_{r_k}}} \right], \tag{5.7}$$

where each entry of the relation histogram is normalized by $n_{ab,i}$, the total number of times the two parts $p_{a,i}$ and $p_{b,i}$ of relation $r_{k,i}(a,b)$ were observed in all views of O. Finally, for the two relations considered in this study the overall *normalized histogram vector* computed from all views of an object O is defined as

$$\overline{\boldsymbol{h}}_o^T = [\overline{\boldsymbol{h}}_{p,o}^T \quad \overline{\boldsymbol{h}}_{r_1,o}^T \quad \overline{\boldsymbol{h}}_{r_2,o}^T]. \tag{5.8}$$

5.4.2 Calculation of the Posterior

The posterior probability of an object O given a new view V is formulated using Bayes' rule:

$$P(O|V) = \frac{P(V|O)P(O)}{P(V)}. \tag{5.9}$$

Leaving aside the part relations for a moment, the likelihood of observing the new view is computed from the likelihood of each of the observed parts $p_1, \ldots, p_{n_{p,v}}$ (assumed independent from each other) as

$$P(V|O) = P(p_1, \ldots, p_{n_{p,v}}|O) = \prod_{i=1}^{n_{p,v}} P(p_i|O) \tag{5.10}$$

and the log-likelihood as

$$\log\left(P(V|O)\right) = \sum_{i=1}^{n_{p,v}} \log\left(P(p_i|O)\right), \tag{5.11}$$

where $n_{p,v}$ is the number of all parts detected in the new view.

How does this formulation relate to the aforementioned histograms? A moment's thought should convince the reader that the part likelihoods $P(p_i|O)$ in fact correspond directly to the entries of $\overline{\boldsymbol{h}}_{p,o}$. Furthermore, (5.11) sums up the probabilities over the parts detected in a new view V, which corresponds to a multiplication by the counts contained in $\boldsymbol{h}_{v,p}$. The log-likelihood from (5.11) can therefore be computed as the scalar product

$$\log\left(P(V|O)\right) = \left\langle \boldsymbol{h}_{v,p}, \log\left(\overline{\boldsymbol{h}}_{p,o}\right)\right\rangle \tag{5.12}$$

yielding the posterior

$$P(O|V) = \frac{\exp\left(\left\langle \boldsymbol{h}_{v,p}, \log\left(\overline{\boldsymbol{h}}_{p,o}\right)\right\rangle\right) P(O)}{\sum_{l=1}^{n_o} \exp\left(\left\langle \boldsymbol{h}_{v,p}, \log\left(\overline{\boldsymbol{h}}_{p,l}\right)\right\rangle\right)}, \tag{5.13}$$

where n_o is the number of objects.

When additionally considering the part relation likelihoods, one must be careful to condition these on the parts contained in each relation. Considering only one type of part relation r_k for notational simplicity, the likelihood becomes

$$\begin{aligned} P(V|O) &= P(p_1, \dots, p_{n_{p,v}}, r_{k,1}, \dots, r_{k,n_{r_k,v}}|O) \\ &= \prod_{i=1}^{n_{p,v}} P(p_i|O) \prod_{i=1}^{n_{r_k,v}} P(r_{k,i}(a_i, b_i)|O, p_{a_i}, p_{b_i}), \end{aligned} \tag{5.14}$$

where p_{a_i} and p_{b_i} are the two parts contained in relation $r_{k,i}$. The conditional relation likelihoods $P(r_{k,i}|O, p_{a_i}, p_{b_i})$ correspond directly to the entries of $\overline{\boldsymbol{h}}_{r_k,o}$. Extending (5.12) one obtains

$$\begin{aligned} \log\left(P(V|O)\right) = &\left\langle \boldsymbol{h}_{v,p}, \log\left(\overline{\boldsymbol{h}}_{p,o}\right)\right\rangle + \left\langle \boldsymbol{h}_{r_1}, \log\left(\overline{\boldsymbol{h}}_{r_1,o}\right)\right\rangle \\ &+ \left\langle \boldsymbol{h}_{r_2}, \log\left(\overline{\boldsymbol{h}}_{r_2,o}\right)\right\rangle \end{aligned} \tag{5.15}$$

and using (5.8), the full posterior is computed as

$$P(O|V) = \frac{\exp\left(\left\langle \boldsymbol{h}_{v}, \log\left(\overline{\boldsymbol{h}}_{o}\right)\right\rangle\right) P(O)}{\sum_{l=1}^{n_o} \exp\left(\left\langle \boldsymbol{h}_{v}, \log\left(\overline{\boldsymbol{h}}_{l}\right)\right\rangle\right)}. \tag{5.16}$$

It should be noted that this posterior only provides a likelihood (with incorporated prior) normalized by the *known* object classes. More advanced methods used in other robotics domains, that attempt to factor in the general probability of observing an *unknown* object [31], usually require sampling from a model distribution. While these approaches are conceptually already rather involved for continuous feature spaces, where the unknown object often can be expressed by uniform distributions of the feature entries, it is even more complex for part-based approaches, where the unknown object could consist of any number of parts of any type. The so-called problem of 'novelty detection' in the context of part-based classification is therefore not considered further at this point.

5.4.3 Choosing the Prior

In general, there is little reason to favor one object over another prior to incorporating any evidence. In fact, a uniform prior $P(O) = 1/n_o$ yielded good classification results for all data sets used throughout this chapter. However, having tested various approaches it turned out to be beneficial to make an initial estimate about the object class using only the number of detected parts. The rationale behind this is the following: Assuming that only one characteristic part has been detected that occurs with the same frequency in two different objects, (5.16) yields the exact same posterior for both objects. However, it is well possible that one object has very few characteristic parts, while in the other usually many parts are observed. In this case it is more likely that the observed object is the first object. To factor in this prior belief a distribution $P(N|O)$ is estimated using sample mean and variance from the histogram of the number of detected parts N for each object O. Given N detected parts in a new view, the probability

$$P(O|N) = \frac{P(N|O)P(O)}{P(N)}, \tag{5.17}$$

where $P(O)$ is the uniform prior, can then serve as the prior of (5.16). In fact, in the resulting expression either equation can be regarded as the posterior and the other as the prior. This 'reweighted' posterior consistently yielded better classification results than using (5.16) alone.

5.5 Classification

This section describes the final steps required to actually retrieve a classification result from the afore-developed probabilistic formulation and summarizes the framework.

5.5.1 Extracting Views from Scans

Up to this point, a silent assumption has been that a new view V contains at most one object. As initially stated, a 3D scan of a robot's environment may, however, contain several objects of interest and lots of background data. A common approach [36] is to heuristically remove wall and ground plane points, see Sec. 4.2.2, isolate the remaining, spatially disconnected point clouds, see Sec 4.2.3, and regard them as views of potential objects. Since in this framework the entire scan is segmented into parts at the beginning, the (slightly different) task is to group close-by segments to obtain meaningful partitions and discard individual parts that most likely constitute clutter. The aim of this section is not to research the best method for partitioning; rather, experimenting with different strategies, a workable and efficient solution turned out to be to agglomerate parts whose

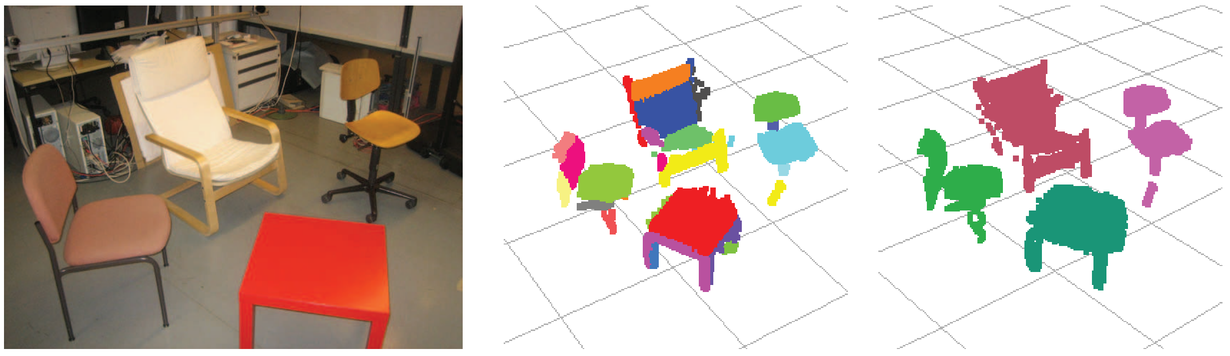


Figure 5.2: In a segmented scan, parts are grouped into object views using nearest-point distance: (a) Photograph of a scanned scene. (b) The segmented scan. (c) The views that result from grouping parts that are within a nearest-point distance of 10cm.

nearest-point distance is within a given threshold, e.g. 10cm and then remove all groups that contain only one part. Fig. 5.2 shows the grouping obtained by this method for a real 3D scan of four furniture objects.

5.5.2 Recognizing Objects

Given a number of views that have been extracted from a scan, the object recognition framework must classify each of them either as background or as one of the known object classes. To this end the posterior probability of each object is calculated for each view. In many object recognition applications, classification is performed by choosing the object class with *maximum a posteriori* (MAP) probability. While this yields the most likely candidate from the set of known objects, it excludes the possibility of having observed an unknown object. Since the posterior is a normalized probability, a more refined procedure is to choose a detection threshold P_d that can be regarded as the confidence level of the classifier. Classification is then performed by choosing the MAP object class that exceeds P_d. When no value exceeds this threshold no object is detected. To see why this is reasonable, consider an example in which only one part has been detected, e.g. a leg that could belong to any of three chairs or two tables. While one of the five furniture objects might have the highest posterior probability (around 20%) this would generally not justify making a classification decision. The threshold P_d therefore provides control over the desired confidence the classifier should have before it makes a decision.

5.5.3 Summary

The outlined steps which are at the heart of the recognition framework are briefly summarized here.

- In the training phase the framework requires a set of n_v real or simulated views of n_o objects. The views must contain one object, have a corresponding class label and are automatically segmented into parts.

- Across all views of all objects, common parts are found by an agglomerative clustering method in the segment feature space. Clusters with more than $n_{\min}$ parts are retained as *part words* of a dictionary.
- All pairs of parts that have been observed together in at least one view are examined for relevant geometric relations between them, such as the angle between normal vectors or the centroid distance. These are extracted from histograms and retained as *relation words* of a dictionary.
- Detection of parts and relations is performed by testing whether these statistically fall into a defined threshold for any of the dictionary words. Detected parts and relations are counted in a histogram for each view.
- From all views of the training data, a *normalized histogram vector* is computed for each object.
- In the recognition phase, a new scan is first segmented into parts, parts are then merged into views, and from the histogram of each view the posterior probability of having observed a known object is computed from (5.16), using (5.17) as the prior.
- The classification decision is made by choosing the object class with MAP probability that exceeds the detection threshold P_d. If no posterior exceeds this threshold the view is classified as background.

5.6 Simulation Study

The described recognition framework was first tested in an extensive simulation study. To this end, CAD models of $n_o = 12$ IKEA furniture objects, namely 6 chairs and 6 tables were created, see Fig. 5.3. Using the simulator introduced in Sec. 2.4.3, each furniture object was scanned $n_v = 40$ times by the actuated Hokuyo UTM-30LX shown in Fig. 2.12(b). The objects were placed at randomized positions (x, y) within a 5x5m area in front of the robot and randomized rotations $\psi \in [-\pi, \pi]$ about the z-axis. The point clouds were obtained from vertical scans and automatically labeled using the name of the CAD model.

5.6.1 Clustering Results

The first part of the evaluation was to assess how well the proposed clustering algorithm finds common parts across the different furniture scans. To this end, all $n_o \cdot n_v$=480 views were segmented and a total of 1,837 parts were clustered using Alg. 5.1. Features were weighted using $w = [0.30\ 0.30\ 0.13\ 0.13\ 0.13]^T$, i.e. emphasis was given to the first two features. Fig. 5.5 visualizes the part words obtained for $\rho_{\max} = 0.10$ and $n_{\min} = 4$ for a subset of views and objects. For this threshold the dictionary contains a total of $n_p = 24$ part words. As can be seen, the clustering algorithm consistently captures the structure of common parts across different views. Note that in some views only one part was observed

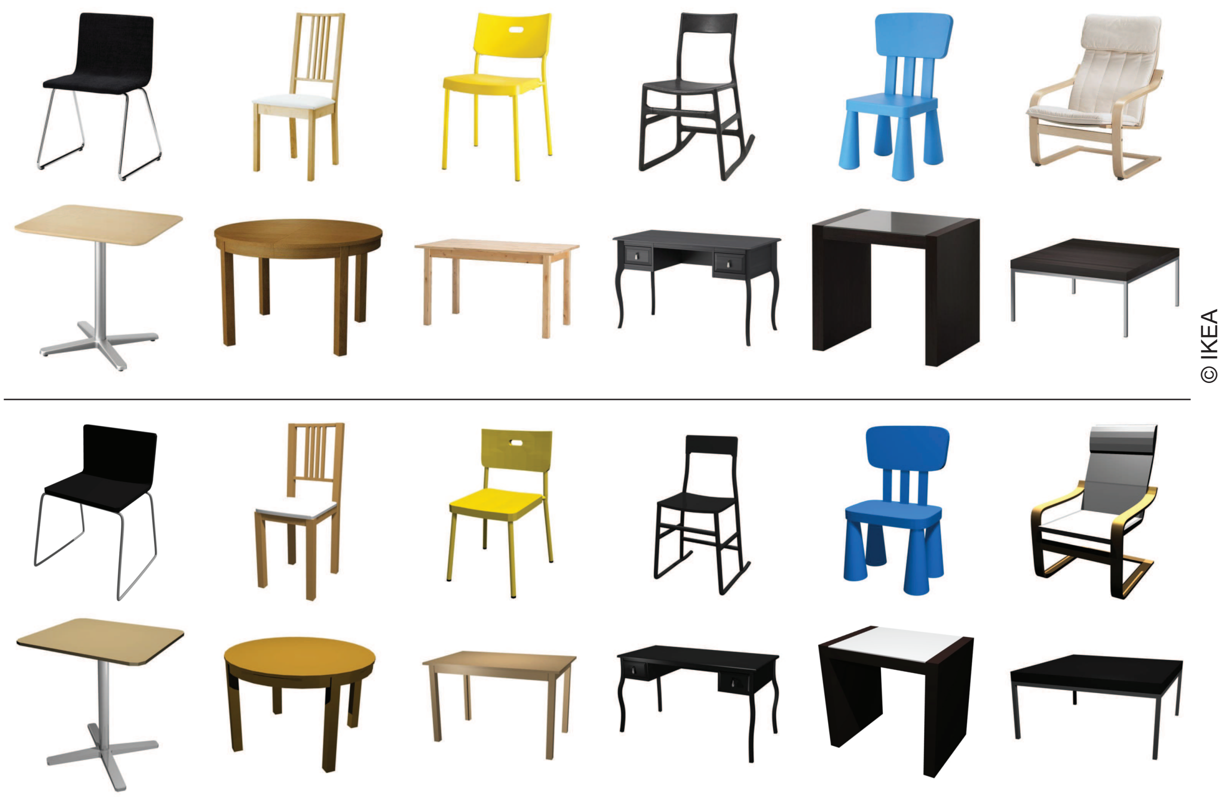


Figure 5.3: Two top rows: Photographs of real IKEA furniture selected for simulation. Two bottom rows: The corresponding simulation models.

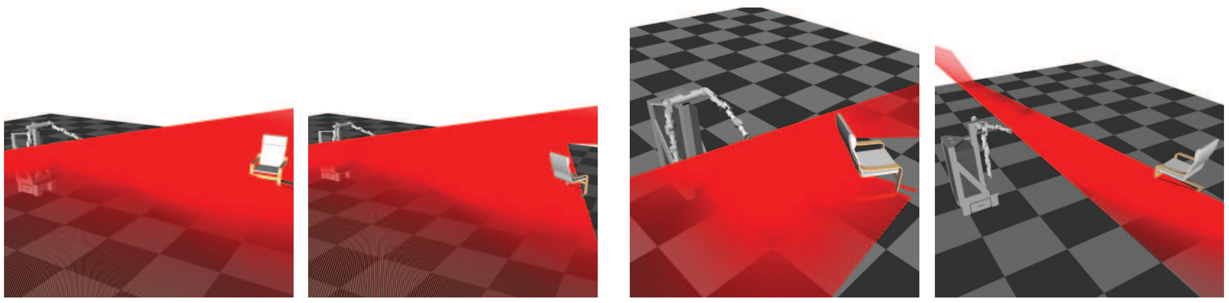


Figure 5.4: In the simulation both position (x, y) and orientation ψ of furniture objects were randomized to generate n_v different views of each object. Each view is automatically labeled with the object name, and no further labeling of parts is required.

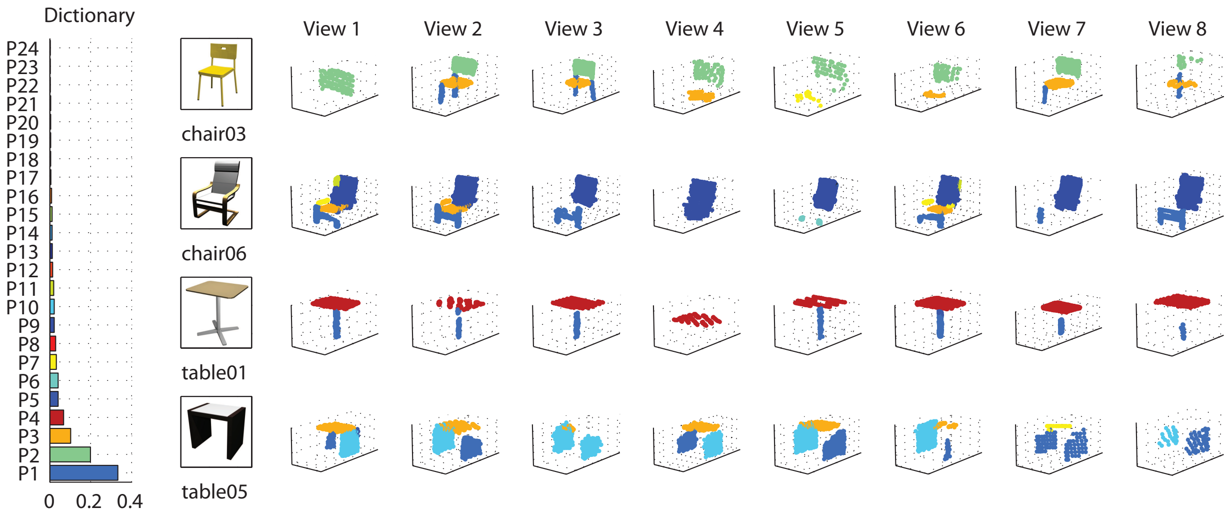


Figure 5.5: Visualization of part words for a subset of objects and views: The part dictionary has 24 words and the vertical histogram on the left shows the relative frequency of each word. The most frequent part observed across all objects are legs (light blue, observed in both tables and chairs), followed by backrests (light green, observed in several chairs). It can be seen that the chosen clustering method yields a meaningful set of part classes that consistently occur across different views of the same objects, even when only parts of the object are visible. Note that although position and orientation were randomized during simulation, here, all views have been re-oriented and point in the same direction to facilitate visual comparison.

and that segments are labeled consistently despite different point densities, which speaks to the robustness of the segment features.

5.6.2 Histogram Results

After constructing the part dictionary, relations between the observed pairs of parts were extracted as words when their histogram peak exceeded 10% for $r_1(a, b)$ and 5% for $r_2(a, b)$. This yielded $n_{r_1} = 7$ normal relation words and $n_{r_2} = 2$ distance relation words, making for an overall dictionary of $n = 33$ words. A value from the χ^2 table for 99% of inliers was used for the detection of both parts and relations, i.e. $\delta_p = 3.88$ for parts and $\delta_r = 2.58$ for relations. The corresponding normalized histograms $\overline{\boldsymbol{h}}_o$, each computed from 40 views, are shown in Fig. 5.6. These can be regarded as the 'fingerprint' of each object, revealing the characteristic distribution of parts and relations. Note that due to the different normalization of parts (5.6) and relations (5.7) relations appear to be dominant in frequency. This is because the entries actually already represent the part likelihoods $P(p_i|O)$ and relation likelihoods $P(r_{k,i}|O, p_{a_i}, p_{b_i})$. Also note that the relation likelihood is almost always 1 when the respective parts have been observed. This follows from the fact that the training set only contains objects of interest in which the relation is always present. If in a real scene two background segments should incidentally be classified as valid parts, the absence of the relevant relation will lower the probability of having observed the actual object, which robustifies the recognition.

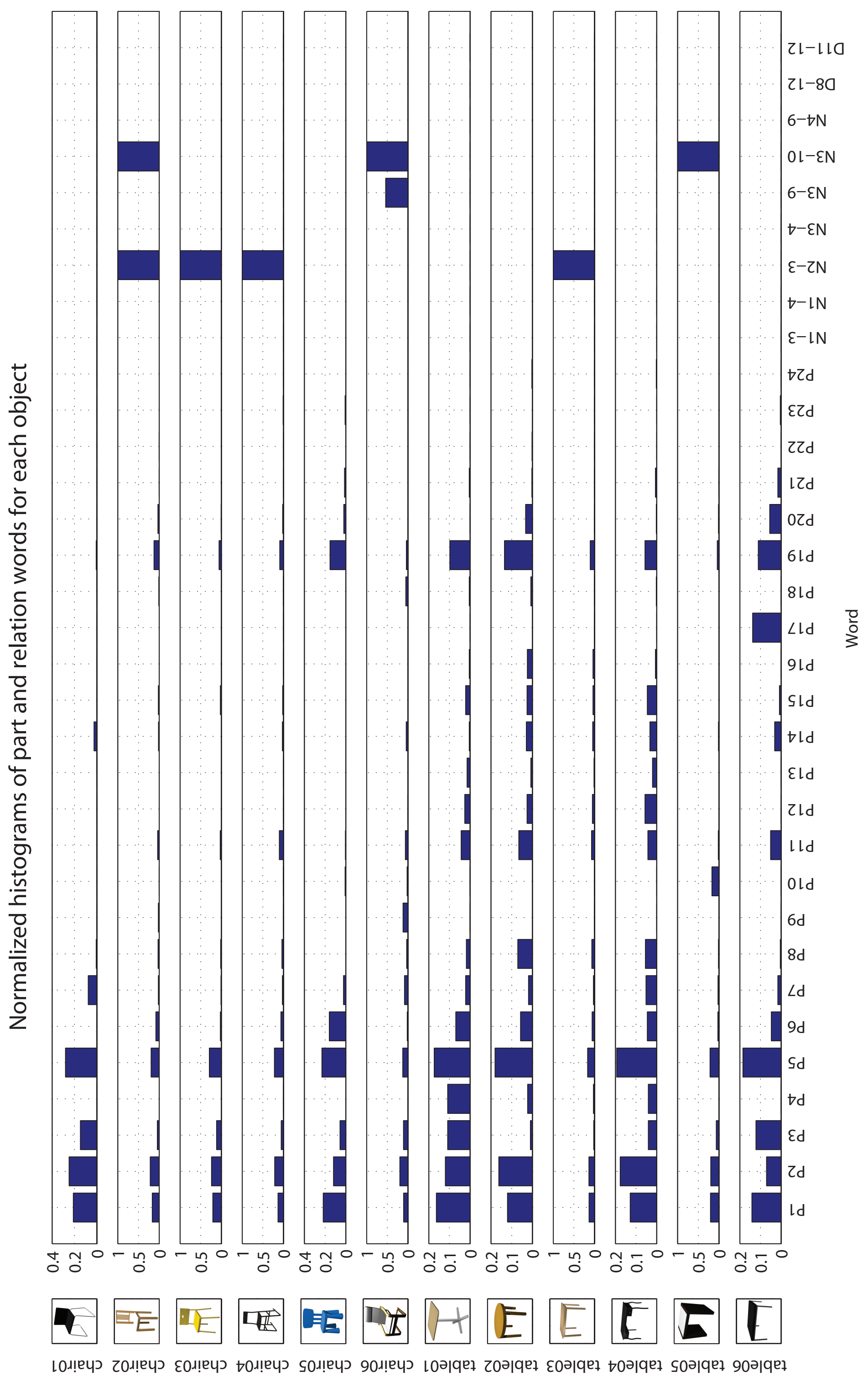


Figure 5.6: The normalized part histograms $\overline{\boldsymbol{h}}_o$ can be understood as the 'fingerprint' of each object. Every object exhibits a characteristic distribution of common parts and relations between them, which is computed from n_v labeled views of the object. **Pa** are the part words, **Na-b** the relation words $r_1(a, b)$ (angle between normal vectors), and **Da-b** the relation words $r_2(a, b)$ (distance between centroids). Note that relations are almost always detected when the two respective parts are detected. This is obvious because all training examples actually contained an object of interest and not background with coincidentally detected parts.

5.6.3 Recognition Results

To evaluate the recognition performance, first of all full posterior probabilities were calculated for each of the training examples. In addition, an independent $12 \cdot 40$ new views were generated as a test set and posteriors were computed for these views as well. Experimentation with the segmentation threshold revealed that for a smaller threshold $\rho_{\max}$ and thus larger dictionary size better classification accuracy could be achieved down to a threshold of about 3%. Fig. 5.7 shows the confusion matrices for both the training and the test set for $\rho_{\max} = 0.03$ which results in a dictionary of 110 part words, 45 normal relation words, and 307 distance relation words. Using P_d=90.0%, on the training set 97.6% classification precision (ratio of true positives and all positives) was achieved, with 23.1% false negatives, on the test set 95.5%, with 30.4% false negatives. It can be seen that confusion occurs mainly between `chair03` and `chair01`, which are structurally very similar. Note that for these results, the problem of scene segmentation was still omitted because each data set contains exactly one view. Fig. 5.8 shows how classification accuracy varies with the segmentation threshold (and consequently dictionary size). The improvement of classification performance with more dictionary words is explained by the fact that a greater vocabulary is better able to capture the variability in appearance of each object. However, one must be careful with a generalization of this trend: in the extreme case $\rho_{\max} = 0$ and $n_{\min} = 1$, where every segmented part becomes its own word, one essentially perfectly memorizes the training set. This, however, does not guarantee better performance on unseen data. Recognition then corresponds to nearest neighbor classification, where a new view is matched to the most similar previously seen part by means of its histogram. While this may yield decent classification results, one loses the ability to generalize the structure of object classes and ends up with a classification that has linear complexity in the number of training examples.

5.7 Experimental Study

Following the simulation study, the recognition framework was tested on a set of real 3D range scans. Three different chairs and one small table were scanned by the actuated Hokuyo UTM-30LX introduced in Sec. 2.5 with a 180° rotational sweep around the wrist. Their position and orientation was randomly varied across 40 different scans as shown in Fig. 5.9. The scans were relatively sparse, with each scan consisting of ca. 170,000 points, and each object view of 2,000-4,000 points. To build the training set, the scans were automatically pruned to a region of interest, segmented with the toolchain from Chap. 4 and grouped as described in Sec. 5.5.1. The only required user input was to assign an object label to each of the four views in each of the scans. Many of the scans contained partial occlusions resulting in views in which parts were only half-visible or even missing entirely.

5.7.1 Recognition Results

The first 20 scans were used as a training set and the other 20 as a test set. The training set was clustered using $\rho_{\max}$=0.04 and $n_{\min}$=4, which yielded a dictionary of 37 part words, 64

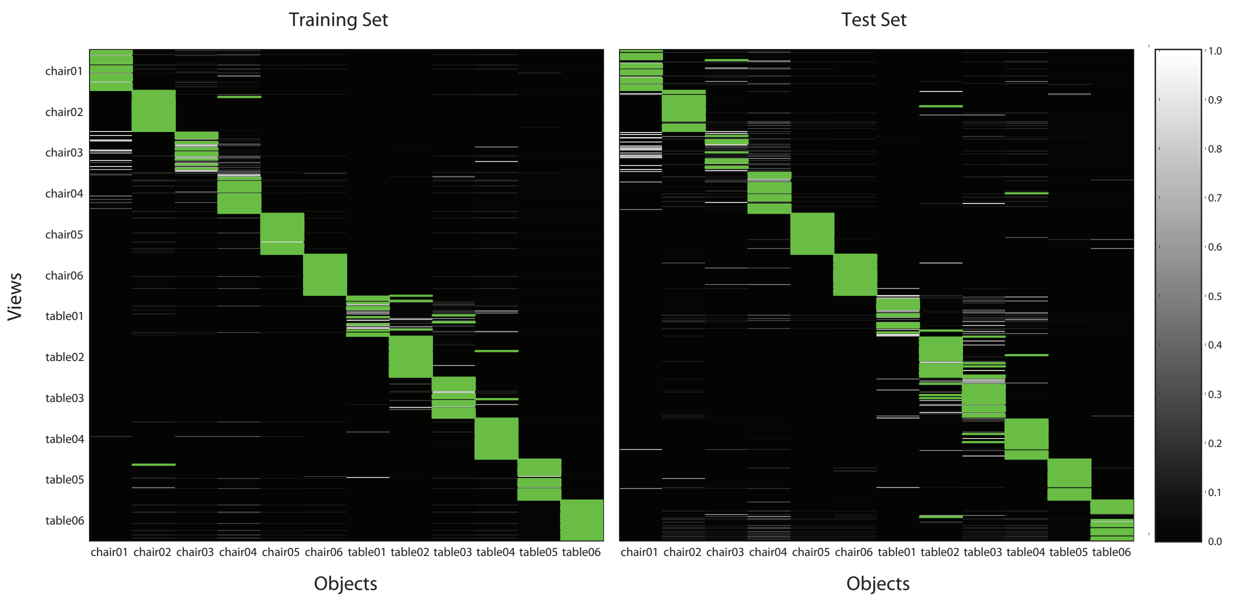


Figure 5.7: Confusion matrices for the simulated training and test set (each containing a total of 480 views): Each line shows the posterior probability of an object class given a view, the green lines highlight the object class with *maximum a posteriori* probability for a detection threshold P_d=90.0%. On the training set this classifier achieves a precision of 97.6% and an accuracy of 75.0%. Right: On the test set, the precision is 95.5% and the accuracy 65.5%.

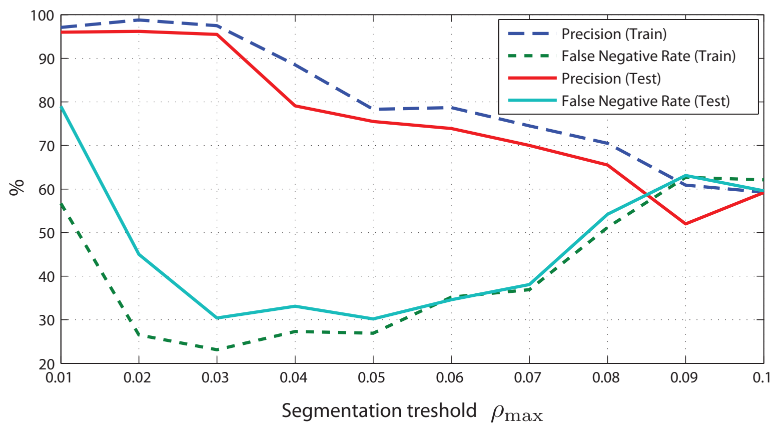


Figure 5.8: Classification accuracy vs. segmentation threshold: For both the training and the test set a larger dictionary better captures the variability of different views of the same object. The bottom two lines (green and cyan) show the percent of objects that are not detected (false negative rate), the top two lines (blue and red) show which percentage of the remaining views was classified correctly (precision). All results were obtained using a detection threshold of 90.0%.

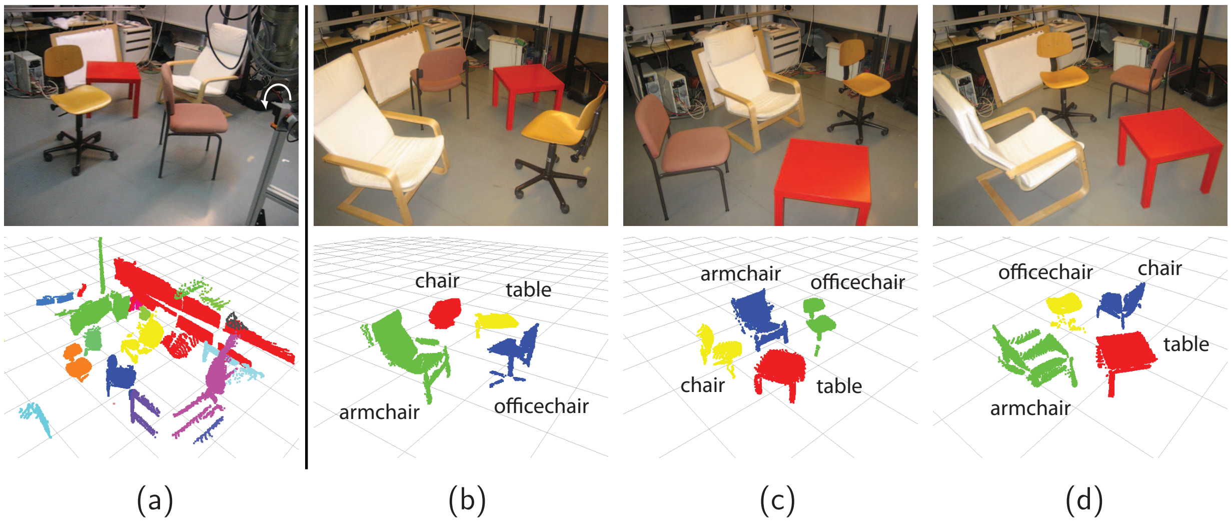


Figure 5.9: Scans of real furniture: Four furniture objects were scanned with a Hokuyo UTM-30LX laser range finder mounted on a 7 DoF articulated robot arm. (a) Top: The scanning setup, with a white arrow indicating the rotational scan performed by the robot arm. Bottom: The resulting segmented point cloud (with clutter). A total of 40 such scans was performed. (b)-(d) Top: Position and orientation of the four furniture objects are randomized across different scans. Bottom: The resulting scans. To build the training set the scans were automatically pruned to a region of interest, segmented and grouped to views; the user only had to label the four objects in each scan as shown. Note that many of the scans contain partial occlusions.

normal words, and 60 distance words. The classifier was first run on the 4·40=160 labeled views, using a detection threshold of P_d=90.0%. On the test set 100.0% precision was achieved, with 18.8% false negatives and on the training set 92.3% precision, with 35.0% false negatives.

To assess how well the part-based recognition approach suppresses background segments, the classifier was then run on the 40 entire scans, not just the labeled views. As opposed to the simulation data, where each scan consisted of exactly one view of a learned object, the real scans contained an average of 24.2 grouped views, all but four of which constituted background. Fig. 5.10 shows how *accuracy*, *precision*, and *recall* vary with the detection threshold P_d. It can be seen that *accuracy*, the ratio between correctly detected or rejected views and all views, quickly reaches 90% as P_d is raised. As the overwhelming majority of views does not contain objects of interest, this means that background is robustly rejected by the part-based recognition framework. This was also reflected in the histograms, where most background views simply had no detected parts. *Precision*, the ratio between true positives and all positives, reaches 86% for P_d >90%. *Recall*, the percentage of detected objects, drops at P_d=50% and attains a level of 43% in the area of high precision.

The influence of the part relations was investigated by again running the classifier on the training and test set with δ_r=0, i.e. this time no relations were detected. Accuracy, precision, and recall came out only 2% lower, with largely the same profile over the detection threshold P_d. Although the absence of relations thus yields a slightly worse performance, for both the training and the test set the change is virtually insignificant, which is why it has not been visualized in Fig. 5.10. The small improvement gained by the use of relations for the considered datasets is explained by the fact that the parts are already highly discriminative. As stated before, the large majority of histograms of background objects had no detected parts. For recognized objects, confusion occurred between those objects that share similar relations. Therefore, disambiguation by means of relations was only required in very few cases. Furthermore, the background views did not vary throughout different scans, i.e. while furniture was moved from scan to scan, the background surfaces of the laboratory environment remained the same. The relations can be thus be expected to have a greater impact when i) more objects with similar parts are learned, and ii) object are scanned in natural environments with changing background. Nonetheless, an important (and expected) insight gained from the current datasets is that recognition results are only improved by relations, not deteriorated.

5.8 Discussion

In general, both the simulation as well as the experimental results presented in the previous sections show that strong classifiers can be learned with the outlined approach. Keeping in mind that the views are comparatively sparse and contain partial occlusions and missing parts, a precision of over 90% in both simulated and real range data demonstrates that learned objects can reliably be recognized, even from few training examples. Also, with an accuracy of over 90% on the real scans, the classifier can robustly tell positives from negatives, i.e. it knows to distinguish known objects from background.

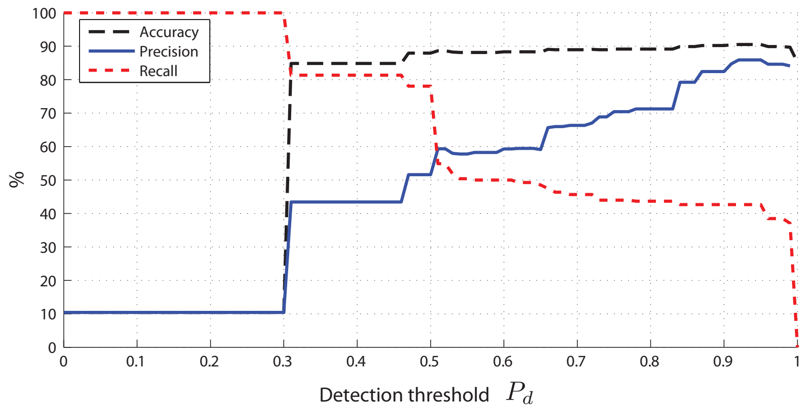


Figure 5.10: Accuracy, precision and recall: On 40 scans of real furniture, the classification *accuracy* (percentage of correctly detected or rejected views) quickly reaches 90% as the detection threshold is raised. *Precision* (percentage of correctly detected views) attains a peak value of 86% at a detection threshold of over 90% and *recall* (percentage of detected objects) drops sharply at a detection threshold of 50%.

5.8.1 Confusion and Features

Confusion occurs mainly between structurally similar objects. E.g. in the simulation false positives occur between `table01`, `table02`, and `table03`. In fact, for these objects the table surfaces are positioned at the exact same height (as with most IKEA tables) and have approximately the same bounding box volume, meaning that they are insufficiently discriminated by the chosen features. Some of this confusion may be remedied by introducing more sophisticated features. However, as with any sensor-based recognition scenario, a substantially improved performance can really only be achieved by increasing the resolution, i.e. scanning and processing point clouds with higher densities. The confusion between `chair01` and `chair03` e.g. occurs because the two most characteristic parts of both objects are the rectangular backrest and the square seat surface. Both parts as well as the geometric relations between them are structurally virtually identical, so unless the scanning resolution is increased to reveal the hole in the backrest of `chair03`, no set of geometric features has a chance of discriminating the two.

Some alternative features that lend themselves to characterizing 3D point sets have been presented in Sec. 3.5. All of them are costly to compute and their merits would have to be examined in detail. While 2D descriptors such as convex hulls of projected points, may obviously yield poor results for highly nonplanar segments, they can still be expected to be rather descriptive, because the majority of surfaces encountered in household objects are roughly planar. As outlined before, 3D descriptors such as 3D convex hulls, MVBBs, and 3D Zernike descriptors work best with a dense point sampling, which is not provided by the scenario considered here. The selection of more sophisticated features, especially position- and orientation invariant ones, therefore remains an interesting open question to further improve the classification performance.

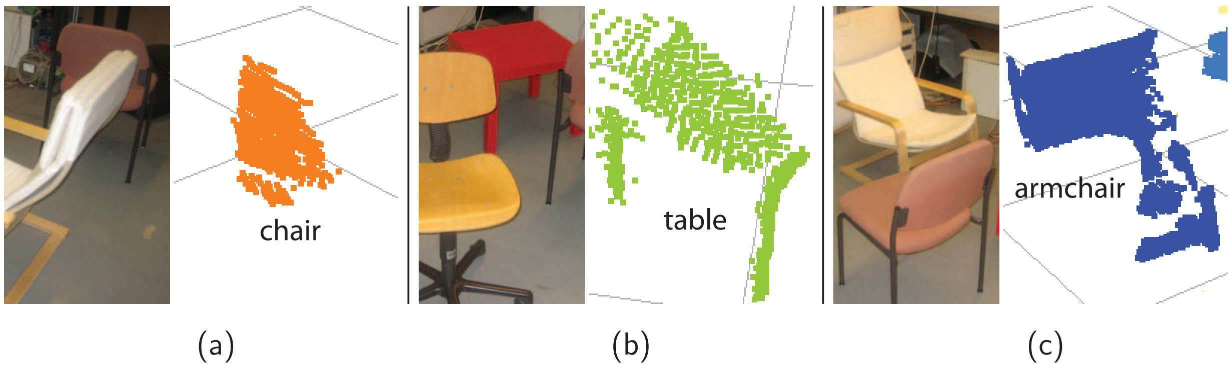


Figure 5.11: Examples of partially occluded and correctly detected objects: (a) The occluded chair is recognized because of its partially visible backrest. (b) The table is recognized even though more than half of it is occluded. (c) The armchair is recognized despite the occlusion of the largest part of its backrest.

5.8.2 Strength of the Part-based Approach

Although no direct comparison was attempted, conceptually, for various scenarios the part-based methodology offers significant advantages over approaches that attempt recognition at the point level, e.g. with local feature histograms. Especially in the presence of occlusions, the abstraction from points to segments provides the possibility of recognizing objects by characteristic parts even when more than half of the points are missing. Fig. 5.11 shows three scans in which the respective objects were detected although large sections were occluded by objects in front of them. To be fair, there are of course also scenarios in which part-based approaches can be expected to yield poor performance, e.g. when recognizing humans, which have great variability in shape and might thus be better recognizable using point-based approaches [36].

5.8.3 Clustering

As previously mentioned, several clustering methods were tried for dictionary construction. Disappointingly, x-means [109], one of the more popular k-means variants which efficiently retrieves the correct number of underlying Gaussian clusters, did not yield satisfactory results; that is, the clusters retrieved by it did not represent groupings of common parts across different views as those shown in Fig. 5.5. This is likely the case because the underlying clusters are not strictly Gaussian; also the algorithm gives little control over the desired granularity of the result (the only parameters are the minimum and maximum number of clusters desired). Other k-means variants essentially suffer from the same problem – control over the algorithm is only given by the number of clusters, which is unknown. For the previously stated reasons the method of choice was therefore the greedy algorithm presented in Alg. 5.1, the rationale being that a threshold that yields a reasonable part clustering can be expected to generalize to new objects and parts, while for other algorithms the number of clusters would have to be constantly adapted by the user.

5.8.4 Simulation vs. Reality

A desirable long-term objective is to be able to build classifiers from the simulated models that work well with real furniture, provided that the CAD models are sufficiently accurate. For scenarios such as recognizing IKEA furniture this would provide a viable way of learning a vast set of relevant objects offline. These objects could then simply be deployed to robots working in household environments. Unfortunately, one phenomenon prevented the successful use of the classifiers built for the simulated models, e.g. `chair06`, on the real data: the real scans exhibit a lot of scattered points that occur beyond surfaces that are almost incident with measurement rays; Fig. 5.11(b) shows this type of noise for the table surface. These artifacts occur in many scans and with different laser range finders. They cause the segmented surfaces and consequently the part dictionary learned from real data to be significantly different from the simulated dictionary. A mid-term goal is therefore to include this phenomenon in the simulation in order to obtain one set of classifiers that works for both worlds.

5.8.5 Improvements

As can be seen from the precision and recall curves in Fig. 5.10, when varying P_d there is a trade-off between having a very precise classifier that is conservative and detects objects in less than 50% of all cases, and having a classifier that will detect all objects but also yield many false positives. Having control over this trade-off by means of the detection threshold is a big advantage. However, the further improvement of the precision/recall rate definitely is one of the main goals for future research. Several areas to explore to this end are the previously mentioned introduction of new features and higher resolution data to reduce the confusion between positives. Also, more advanced methods for novelty detection might aid classification accuracy.

6 Conclusions

Stu, the alien apprentice, with many options to pursue
Lifted, © 2006 Disney/PIXAR

Summary

This chapter provides a summary of the presented material and some concluding remarks. Strengths and weaknesses of the presented approaches are discussed and possible directions for future research are outlined.

The previous chapters have presented a chain of abstraction that leads from raw point cloud data to recognized objects. As such it constitutes an instance of a perception module required for intelligent behavior of cognitive mobile autonomous robots. The developed methods and representations naturally offer numerous possibilities for improvements and enhancements, some of which are outlined in this chapter. Following a summary of the thesis in Sec. 6.1, possible directions for future research are discussed in Sec. 6.2, and closing remarks are provided in Sec. 6.3.

6.1 Summary

The contents of Chaps. 2-5 are briefly summarized in this section.

Chapter 2 Methods for the acquisition of 3D data were presented in Chap. 2. Available sensing modalities that are suitable for mobile robotics generally include stereo vision as a passive method as well as 2D and 3D laser range finders and time-of-flight cameras as active sensors. Although it is steadily gaining in popularity and robustness, currently stereo vision is still impaired in its accuracy and usability by its short range, poor depth resolution and dependence on object texture. Time-of-flight cameras suffer from similar problems: mediocre depth resolution and high random and systematic noise currently render the 3D data obtained by these sensors inferior to other active methods. However, as pointed out, resolution can be expected to increase significantly over the coming years, thus making the direct acquisition of colored depth data with a ToF camera one of the most attractive sensing modalities in the near future. As of now, laser range finders are among the most popular 3D acquisition methods, because of their high resolution and good depth accuracy. Apart from expensive 3D scanning setups, the method favored by most robotics researchers is the actuation of a 2D laser range finder. While this renders acquisition times relatively slow because one needs to wait for a sweep of scanlines to be completed, the obtained point clouds are directly usable for normal vector estimation, segmentation etc. For these reasons, the central sensing scenario outlined in Chap. 2 is that of an autonomous mobile robot performing a quick horizontal, vertical or rotational sweep with a 2D laser range finder to assess the state of its environment from a relatively sparse point cloud. The simulation of such data as well as the hardware systems used to verify the efficiency of the methods in experiment were presented.

Chapter 3 Chap. 3 addressed the problem of computing features that describe sampled surface geometry, with normal vectors occupying the largest part of the chapter. A comparison study of numerous different methods for normal vector estimation revealed that i) most methods proposed in literature can be cast into a simple optimization problem and differ only in the optimized cost functional, and ii) for use on a kNN graph, *PlanePCA* is the most favorable method, as it yields the best normal vector estimation quality and speed. Also, as explained, the principal component analysis allows for the natural interpretation of the normal vector being the direction of minimal surface variance. Several important issues not previously addressed in literature, such as the improvement of normal vector estimates in anisotropic neighborhoods, the effect of noise and neighborhood size on estimation quality as well as normal estimation with an Extended Kalman Filter, were investigated in simulative studies. The computation of curvature descriptors from sampled surface points was presented and their sensitivity with respect to noise was demonstrated by a simulation study. Several more advanced point descriptors were discussed. In the remainder of the chapter, a selection of descriptive features for entire point clouds was presented, ranging from simple quantities such as the point centroid or the bounding box volume to advanced features such as 3D Zernike descriptors.

Chapter 4 The tools introduced in the second and third chapter set the stage for the segmentation framework developed in Chap. 4. Following a categorization of the numerous different segmentation approaches existent in literature, an algorithm that extracts surface regions with smooth curvature change was presented. The method operates by grouping neighboring points with similar normal vectors, in a fashion similar to but more efficient than region growing approaches or graph-based methods. Fast retrieval of candidate neighbors is achieved by an adaptive fixed-radius search on a kd-tree, and each point is connected to neighbors whose normal vector is within a given angle. In a second processing step basic instances of oversegmentation caused by partial occlusion are mended by assessing the plausibility of occlusion for nearby candidate segments. These are identified by a number of distance measures and the decision for merging is taken by tracing boundary lines between segments in an omnidirectional range image, which represents a spherical depth mapping of all points observed from one view. The efficiency of the entire segmentation toolchain was demonstrated on several simulated and real-life data sets. Segmentation times were shown to be on the order of the scan times, with the bulk of the processing time spent on the estimation of normal vectors and not the actual segmentation. In both simulation and experiment instances of occlusion that caused 'shattered' surfaces were correctly repaired. As illustrated by the visualized object-oriented bounding boxes, the segments provide a descriptive abstraction of the scanned objects.

Chapter 5 The chain of abstraction was completed by a framework for recognizing objects in 3D range data, using the representation of surface segments. The segmentation toolchain was used to automatically segment a set of labeled object 'views'. For each of the so-obtained surface segments a feature vector containing several of the segment features introduced in Chap. 3 was computed, and a greedy agglomerative clustering strategy was employed to learn a dictionary of the most common parts encountered in different views of each object and across different objects. This dictionary was enhanced by two types of characteristic spatial relations between pairs of co-occurring parts, namely the angle between part normal vectors and the distance between part centroids. The detection of parts (and relations) was performed by testing whether the Mahalanobis distance of a given segment (or pair of segments) in the feature space falls within a defined threshold. The meaning of this threshold under notions of cluster Gaussianity and non-Gaussianity was discussed. The detected parts and relations allow for the compact representation of objects in the form of histograms: a characteristic 'fingerprint' of each object is stored in its so-called mean histogram, and new views were classified by assessing the similarity of their individual histograms to these mean histograms. Rather than using one of the popular unbounded metrics for histogram comparison, this was achieved by a Bayesian formulation that yields the posterior probability of each object class given the histogram of a new view. The posterior was further refined by the incorporation of a prior that considers the number of observed parts. In an extensive simulation study involving 3D models of six chairs and six tables, the viability of the approach was established, with a small dictionary consistently capturing the part structure across different views, and a medium-sized dictionary allowing for perfect recognition of the training set and high recognition rates on previously unseen examples. As a final step, the good recognition performance of the framework was demonstrated on a set of real scans involving four furniture objects and lots of background data. A highly accurate classifier was learned from a small set of examples and, due to the part-based nature of the framework, in several instances objects were correctly classified although the large majority of points were missing due to occlusion. The current strengths and weaknesses of the framework were discussed in detail.

6.2 Discussion and Future Directions

Individual aspects of the topics presented in this thesis were discussed at the end of each chapter. This section comments on the most important issues of each processing step and outlines possible improvements as directions for future research.

Sensing As for sensing hardware, several developments can be expected to drastically improve the acquisition time and quality of 3D data in the future, as outlined in Chap. 2. While time-of-flight cameras are likely to become predominant for indoor scenarios in the long run, compact laser range finders with high scanning frequencies and clever actuation, such as continuous rotation, have the potential of providing an accurate and unobtrusive sensing modality for the near future. The systems presented in this thesis were by no means optimized for acquisition speed; rather their main purpose was to allow for the investigation of different scanning trajectories and characteristics. With an optimized hardware setup the acquisition of full 3D frames at rates of 2-4Hz should become possible, meaning that with suitable processing hardware mobile robots can assess the state of their environment at 1-2Hz. This would also enable the consideration of dynamic scenarios, a challenging issue that has not been covered in this thesis. Sensor artifacts such as the 'skimmed points' described in Sec. 5.8 will likely remain over the next generation of sensors and should thus be incorporated in simulation.

Features The computation of descriptive geometric features is an ongoing research area with a constant stream of publications on new and more sophisticated descriptors. A general conclusion to draw from the selection of features presented in Chap. 3 is that between simple local descriptors and complex quasi-global descriptors there exists a trade-off in descriptive power on the one hand and computational complexity and robustness on the other. Many approaches therefore seek to strike a balance between descriptiveness and robustness by using a collection of local descriptors or hierarchies of such collections, e.g. a set of representative local shape histograms. The lesson to be taken away from the part generalization presented in Chap. 5 is that a highly descriptive and compact object representation can also be derived from extremely simple features paired with a suitable clustering strategy. This holds both for the normal vectors used for clustering as well as the segment features used for learning the part dictionary.

Segmentation The segmentation presented in Chap. 4 represents a toolchain that was conceived with a strong focus on application to real-life data and performance. Because of this, it contains several heuristic parameters that arise from concrete practical considerations and must be tuned for a given sensing scenario. However, an important result to be seen was that once determined, one set of parameters yielded good segmentation results across different scenarios in both simulation and experiment. For real-life applications this may be preferable to non-parametric methods that yield excellent automatic performance on curated datasets but lack parameters to control them when undesired results occur on other data. As such, the segmentation toolchain provides an efficient processing module that can certainly be improved in speed, e.g. by parallelization.

Recognition by Parts The object recognition framework presented in Chap. 5 has provided numerous insights and many more interesting open points to pursue in future research. The main achievement is the demonstration of a viable approach to recognize objects in range data from a few labeled training examples. In contrast to existing methods, this recognition is not performed by the comparison of collections of discretized local features, but rather by the abstraction to a part-based representation. For certain scenarios, the discretization into parts and the comparison of objects at the part level is less forgiving than approaches using local shape histograms. E.g. if shape properties change slightly in many places of the object, no known part might be recognized and the object will consequently be classified as unknown. This may be disadvantageous for recognizing objects that exhibit great variability in shape; however, for the overwhelming majority of structured objects to be encountered in indoor scenarios the part representation is of great advantage. This is because it offers the possibility of robust detection by means of characteristic parts, even when large portions of the object of interest are occluded, as was demonstrated in the experimental evaluation.

Supervised Learning As of now, the framework requires supervision for learning, that is, it must be given a set of labeled views for every new object to be recognized. This data can be integrated incrementally and seamlessly in both the dictionary and the histogram representation without the need to recompute either from scratch. While a large database of recognizable objects can therefore be constructed efficiently with supervision, in the long run, at least one of the following improvements would be desirable:

A first option is the unsupervised discovery of object classes in range scans. This would involve constantly clustering the segments which are rejected by the current vocabulary to discover new common parts and relations as well as views that share these words and thus constitute new object classes. This approach is, however, only feasible if a robot is able to acquire a large amount of representative scans containing the desired training examples on its own. As with any learning task, in the end the process is bound to be much more efficient if a human teacher provides at least some guiding information, such as the number of classes to be discovered.

A second option has been pointed out in Sec. 5.8. Assuming that the simulation of range data can be improved to a point where it mimics all artifacts observed in real range data, histogram representations valid for real objects could be learned completely autonomously from labeled simulation models. In the movie scene depicted on the cover of Chap. 5 the skill to fly a helicopter is simply 'uploaded' to the brain of one of the protagonists. In a similar fashion, the realistic simulation of real-life objects would allow to simply upload the representations of new objects to the robot without the need to learn them on site. While this concept obviously still suffers from a number of restrictions, for household environments, CAD models of structured objects such as IKEA furniture would be readily available. In the long run, the combination of both pre-built, scalable dictionaries as well as unsupervised on-site learning seems like a highly promising way of attaining a new level of environment understanding.

Extraction of Views The currently chosen method for extracting views from scans described in Sec. 5.5.1 is heuristic and error-prone when objects are very close to each other. In general, it cannot be expected that objects of interest will always be sufficiently separated. Cooking utensils could e.g. be placed in a cupboard, kitchen appliances on a cabinet etc. To this end the design of a more flexible approach to attain a grouping of potential views is a worthwhile direction for future research. The basis for this seems to be to reverse the current order of steps and perform part detection on all segments of the scene before extracting views. The detected parts can then be used in a variety of ways to infer possible object classes and groupings. A powerful approach known from computer vision is the use of probabilistic voting schemes to iteratively attain the most likely object hypothesis and grouping [44, 86].

Verification and Reconstruction Two applications that build on top of recognized objects and have not been covered in this thesis are object *verification* and *reconstruction*. Both are important when the coarse approximation provided by the part features, such as centroids and bounding boxes, are not sufficient for subsequent steps. When perception serves as input to planning and reasoning layers, these modules frequently need to know the exact extent and pose of objects. Verification generally aims to establish the presence and exact location of a geometrically known object in the data. For the segmented point cloud it could proceed by fitting a CAD model associated with the object class to the classified view, e.g. by means of the *iterative closest point* algorithm [17]. The localized object can be used for a variety of purposes, such as path and manipulation planning, visualization of the recognition result etc. When no generally applicable geometric model is associated with or can be built for the classified object class, surface reconstruction must be employed to retrieve the geometric structure from the sampled points. As such, this step is independent of recognition; however, since it is computationally involved and may only be relevant for a subset of objects in the environment, it benefits from the selective application to objects after they have been classified. Reconstruction yields an exact or approximate model of the geometry of the recognized object that can e.g. be used for grasp planning [160]. Generally, both verification and reconstruction aim to establish a correspondence between a (simplified) geometric model of the recognized object and the perceived sensor data. For autonomous mobile robots, the maintenance of such correspondences is essential to facilitate planning in a simulated representation of the real world [44, 138].

6.3 Closing Remarks

The construction of intelligent autonomous robotic agents that are able to assist humans in everyday tasks remains an ongoing and open research challenge. As initially outlined, the significant advances that have been made in robotic technologies and intelligent algorithms are starting to put the creation of non-trivial robotic assistants within reach. However, while robotic hardware gradually converges to a state that replicates relevant human abilities sufficiently well, the bigger challenge today is the creation of a cognitive architecture that orchestrates the sensorial and actuatorial resources to what is commonly perceived as intelligent behavior.

This thesis has presented a number of important contributions to the most fundamental capability that underlies all such behavior: intelligent perception. The framework developed throughout the chapters has demonstrated that today's technical systems can achieve an 'understanding' of 3D environments with state-of-the-art sensing technology and, more importantly, a set of efficient algorithms. For the scenario of household robotics, the transition from a list of point coordinates to a set of recognized objects represents a crucial prequisite for a plethora of exciting applications. However, the methods for feature computation, surface-based segmentation, and part-based object recognition are by no means confined to this scope; in general, they are useful tools for any scenario involving a surface sampling of structured 3D geometry. The viability of the individual methods as well as the overall framework demonstrated in this thesis should thus serve as a motivating stepping stone in the constant quest for more intelligent autonomous mobile robots.

A Appendix

A.1 Nearest Neighbor Searching

Many of the processing steps described in this thesis require the retrieval of nearest neighbors in a metric space. For a set of n points this implies finding the k points that have the smallest distance to a query point $\boldsymbol{x}_q$ under the used distance metric.

The most straightforward solution to this problem is *linear search*: First, the distance between the query point and every point in the set is computed. The distances are then stored in a list (with references to the points) and sorted in increasing order. The first k references of the sorted list contain the nearest neighbors. With an efficient sorting algorithm such as *quicksort* or *heapsort*, finding k nearest neighbors thus takes $O(n \log(n))$ expected time.

When the distance metric can be expressed as a Minkowski L_p norm of the form

$$\rho(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p = \sqrt[p]{\sum_{m=1}^{d} |x_{im} - x_{jm}|^p}, \tag{A.1}$$

which includes the L_1 norm, the Euclidean norm, as well as the L_∞ norm, the process can be accelerated efficiently by the use of spatial data structures. These process a fixed set of data points and store them in a way that facilitates the fast retrieval of nearest neighbors, e.g. in binary trees, bins and buckets or by hashing point coordinates. Apart from *quadtrees* and *octtrees*, which represent specific acceleration structures for two- and three-dimensional data, the most efficient generic data structure are *kd-trees*.

A *kd-tree*, short for *k-dimensional tree*, is a spatial data structure that stores a set of points in a *binary space partitioning tree* with axis-aligned splitting hyperplanes [30]. The concept was originally conceived by Friedman et al. [41]. Here, instead of k, which already denotes the number of nearest neighbors, the dimensionality of the space shall be denoted by d. The key idea is that at every level of the tree, the data is split in a different dimension by cycling through each of the d dimensions. Thus e.g. for 3D points with $[x\ y\ z]$-coordinates, the root node splits the data on the x-axis, its children on the y-axis, its grandchildren on the z-axis, its great-grandchildren again on the x-axis and so on. To obtain a balanced tree, the split point is usually chosen to be the median of the point coordinates of the current node; however, refined splitting strategies are possible [195]. When searching for nearest neighbors, the number of points that are examined can be reduced drastically in comparison to linear search, because the splitting planes allow to recursively exclude entire subtrees corresponding to regions that do not intersect the hypersphere of possible nearer neighbors, as illustrated in Fig. A.1.

Construction of a kd-tree for a set of n points has $O(n \log(n))$ time complexity, if a linear

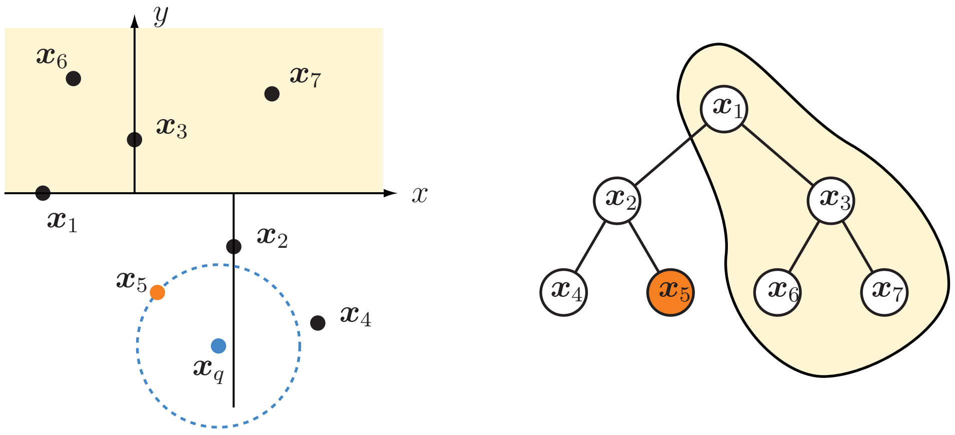


Figure A.1: A kd-tree can greatly speed up nearest neighbor search: The distance to the query point $\boldsymbol{x}_q$ is calculated for each node in a tree during a depth-first descend if the region defined by that node can contain points that are closer than the currently nearest point. In this 2D example, the nearest neighbor is found in the left half of the tree, eliminating the need to examine the right half (shaded), which does not intersect the search hypersphere (dashed) in which nearer points could be found.

median-finding algorithm is used [30]. For randomly distributed points, the expected time complexity of finding the nearest neighbor is $O(\log(n))$, and thus naïvely $O(k \log(n))$ for k nearest neighbors. This is much more efficient than linear search; however, in most applications including range sensing, the data will not be distributed randomly. In the worst case, the runtime complexity of retrieving the nearest neighbor is $O(d \cdot n^{1-\frac{1}{d}})$ [85]. This 'curse of dimensionality' causes kd-trees to become inefficient for high-dimensional data spaces. If d is close to n, no significant speedup is obtained in comparison to linear search. This problem has partially been remedied by the development of methods for *approximate nearest neighbor* searching. If one is willing to accept a small error ϵ, meaning the returned point might not be the true nearest neighbor but will not be more than a factor of $(1+\epsilon)$ away, then significant speedups can be achieved also for high-dimensional spaces [6].

The software package used for nearest neighbor searching throughout all implementations of this thesis is the *Approximate Nearest Neighbor* (ANN) library by Mount and Arya [195]. The library supports a large number of efficient search routines with so-called box-decomposition trees, a refinement of kd-trees, which balance the aspect ratio of regions. Because the dimensionality of the data used in this thesis is small (between 3 and 6 dimensions) in comparison to the number of data points (several thousand scan points) no approximation was used, i.e. $\epsilon = 0$. Several algorithms made use of fixed-radius search, see Sec. 4.2.3, which is also directly supported by the library. These queries are even faster than regular kNN queries, because numerous regions can be directly excluded from the start of the tree traversal, see Fig. A.1.

A.2 Least Squares Regression and Singular Value Decomposition

Several of the methods presented in Chap. 3 involve the solution of an optimization problem of the form

$$\min_{\boldsymbol{b}} \|\boldsymbol{X}\boldsymbol{b}\|_2^2, \quad \text{s.t. } \|\boldsymbol{b}\|_2 = 1. \tag{A.2}$$

This is the classical problem of *total least squares regression*, where the $n \times d$ data matrix $\boldsymbol{X}$ contains n measurements or observations and $\boldsymbol{b}$ is the parameter vector to be optimized. The entries of $\boldsymbol{b}$ represent a $(d-1)$-dimensional hyperplane (the set of all vectors perpendicular to $\boldsymbol{b}$) fixed at the origin, that is fitted to the set of d-dimensional data. For each of the data items $\boldsymbol{x}_i$, the residual is conveniently calculated by the absolute value of the inner product $\|\langle \boldsymbol{x}_i, \boldsymbol{b}\rangle\|_2$, resulting in the expression $\|\boldsymbol{X}\boldsymbol{b}\|_2^2$ for the squared residual of all data points. The usual assumption in least squares regression is that the problem is overdetermined, i.e. there are more measurements than parameters to be optimized. In this case, the objective is to minimize the sum of squared residuals as formulated in (A.2) to obtain the best possible fit.

Following Simoncelli [203], the problem can be solved by performing a singular value decomposition (SVD) of the data. The SVD of $\boldsymbol{X}$ yields

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{A.3}$$

with $\boldsymbol{U}$ and $\boldsymbol{V}$ orthogonal and $\boldsymbol{S}$ diagonal with positive decreasing elements. This can be used to rewrite the cost functional of (A.2) as

$$\begin{aligned}\|\boldsymbol{X}\boldsymbol{b}\|_2^2 &= \boldsymbol{b}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} \\ &= \boldsymbol{b}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{b} \\ &= \boldsymbol{b}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{b}.\end{aligned} \tag{A.4}$$

Introducing the substitution $\boldsymbol{a} = \boldsymbol{V}^T\boldsymbol{b}$, the optimization problem becomes

$$\min_{\boldsymbol{a}} \left\|\boldsymbol{a}^T\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{a}\right\|_2^2, \quad \text{s.t. } \|\boldsymbol{a}\|_2 = 1, \tag{A.5}$$

where $\|\boldsymbol{a}\|_2 = 1$ holds because $\boldsymbol{V}$ is orthogonal and does not change the length of $\boldsymbol{b}$. Since $\boldsymbol{S}$ is diagonal with positive decreasing elements $s_1, \ldots, s_d$, the cost functional of (A.5) is bounded as follows:

$$\boldsymbol{a}^T\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{a} = \sum_{i=1}^{d} s_i^2 a_i^2 \geq \sum_{i=1}^{d} s_d^2 a_i^2 = s_d^2 \sum_{i=1}^{d} a_i^2 = s_d^2 \|\boldsymbol{a}\|_2^2,$$

and with $\|\boldsymbol{a}\|_2 = 1$ one obtains

$$\boldsymbol{a}^T \boldsymbol{S}^T \boldsymbol{S} \boldsymbol{a} \geq s_d^2. \tag{A.6}$$

In other words, the minimum value that the cost functional can attain is the square of the smallest singular value of $\boldsymbol{X}$. It is attained if and only if $\boldsymbol{a} = \boldsymbol{e}_d = [0\ 0\ \cdots\ 0\ 1]^T$. Reversing the transformation $\boldsymbol{a} = \boldsymbol{V}^T \boldsymbol{b}$, the desired optimizer $\boldsymbol{b}^*$ is

$$\boldsymbol{b}^* = \boldsymbol{V} \boldsymbol{e}_d = \boldsymbol{v}_d. \tag{A.7}$$

Thus, the optimal solution for the total least squares regression problem from (A.2) is simply the singular vector $\boldsymbol{v}_d$ of the decomposed data matrix $\boldsymbol{X}$.

A.3 Principal Component Analysis

For several of the features computed in Chap. 3, *principal component analysis* (PCA) represents an important tool that yields orthonormal vectors along which the variance of a set of points is explained in an intuitive and meaningful fashion. Here, two methods for computing the principal components and variances along them are presented.

Given an $n \times d$ data matrix $\boldsymbol{X}$, the objective of PCA is to find a system of orthonormal basis vectors such that the greatest variance of the data occurs along the first basis vector, the second greatest variance along the second (under the constraint that it be normal to the first) and so on [75]. The method was originally conceived by Hotelling [64]. A natural derivation follows from consideration of the sample covariance matrix of the data. The sample mean of $\boldsymbol{X}$ is computed from the column sum of the data matrix,

$$\bar{\boldsymbol{x}} = \frac{1}{n} \boldsymbol{X}^T \boldsymbol{1}_n, \tag{A.8}$$

and the sample covariance matrix from the sum of outer products of the mean-free data items,

$$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T. \tag{A.9}$$

By the spectral theorem, an eigendecomposition of the positive symmetric matrix $\boldsymbol{\Sigma}$ yields a diagonalization

$$\boldsymbol{\Sigma} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{q}_1 & \cdots & \boldsymbol{q}_d \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_1^T \\ \vdots \\ \boldsymbol{q}_d^T \end{bmatrix}, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \tag{A.10}$$

that decouples the data variance. The eigenvectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d$ define an orthonormal coordinate system in which the sample variance σ_i occurs only along the basis vectors and is given by the eigenvalues $\lambda_1, \ldots, \lambda_d$. Since the eigenvalues are ordered by decreasing value in the decomposition, the corresponding eigenvectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d$ represent the desired principal components of the data.

The computation of an eigendecomposition of the sample covariance matrix is not the best way to retrieve the principal components, because it first requires the computation of the sample covariance matrix itself. A more efficient method is the use of an SVD of the mean-free data matrix

$$\widetilde{\boldsymbol{X}} = \boldsymbol{X} - \frac{1}{n}\mathbf{1}_n\bar{\boldsymbol{x}}^T = \boldsymbol{X} - \frac{1}{n}\mathbf{1}_{n,n}\boldsymbol{X} = (\boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_{n,n})\boldsymbol{X}, \tag{A.11}$$

which is simply the data matrix $\boldsymbol{X}$ with its column mean $\bar{\boldsymbol{x}}$ subtracted from every row. The sample covariance matrix from (A.9) can then be written as

$$\boldsymbol{\Sigma} = \frac{1}{n}\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}}. \tag{A.12}$$

Assuming that $\widetilde{\boldsymbol{X}}$ has full rank, its SVD yields

$$\widetilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{A.13}$$

and the covariance matrix can be written as

$$\boldsymbol{\Sigma} = \frac{1}{n}\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \frac{1}{n}\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{V}(\frac{1}{n}\boldsymbol{S}^2)\boldsymbol{V}^T \tag{A.14}$$

which is equivalent to the $\boldsymbol{\Sigma} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$ eigendecomposition from (A.10). The principal components are therefore equivalent to the singular vectors of $\widetilde{\boldsymbol{X}}$, and the variance σ_i^2 along these components is computed from the singular values s_i as $\sigma_i^2 = s_i^2/n$. Consequently, the singular values s_i of $\widetilde{\boldsymbol{X}}$ are proportional to the sample standard deviation by a factor of $\sqrt{n}$.

A.4 Mahalanobis Distance

The *Mahalanobis distance* $\delta(\boldsymbol{X}, \boldsymbol{Y})$, conceived by Mahalanobis [91] for the purpose of comparing measurements of skulls, is a scale-invariant distance measure that determines the similarity of a new sample or sample set $\boldsymbol{Y}$ to a reference sample set $\boldsymbol{X}$. In many applications it represents a more sophisticated way of assessing similarity than a simple Euclidean distance metric because it takes into account the covariance of the reference samples. The key idea is that a new point with a certain distance from the centroid of a point set should be regarded as less similar if it is located in a direction of low variance of that set and as more similar if it is located in a direction of high variance. The concept is illustrated in Fig. A.2. For a point set with sample mean $\bar{\boldsymbol{x}}$ and sample covariance $\boldsymbol{\Sigma}$ the areas of equidistant Mahalanobis distance are hyperellipsoids centered at $\bar{\boldsymbol{x}}$ that have the

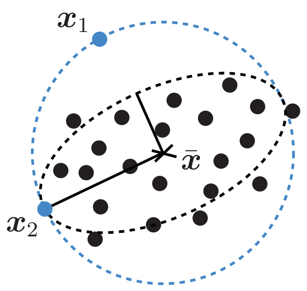


Figure A.2: Which of the 2D points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is closer to the sample mean $\bar{\boldsymbol{x}}$? Under a Euclidean distance metric (dashed circle), both points have the same distance. The Mahalanobis distance measure considers the covariance of the data (dashed ellipsoid) and thus $\boldsymbol{x}_1$ is further away than $\boldsymbol{x}_2$.

eigenvectors of $\boldsymbol{\Sigma}$ as principle axes. Because only the relative location of a point from the reference samples is assessed, the Mahalanobis distance is scale-invariant.

For the simple case of a single one-dimensional sample $\boldsymbol{Y} = y$ and a point set $\boldsymbol{X} = \boldsymbol{x}$ with mean $\bar{x}$ and variance σ, the Mahalanobis distance corresponds to the so-called *normalized distance*

$$\delta(\boldsymbol{x}, y) = \sqrt{\frac{(y-\bar{x})^2}{\sigma^2}}. \tag{A.15}$$

The distance of point y to the mean $\bar{x}$ of the set is normalized by the variance of the data; thus if the point is exactly one standard deviation away from the mean it has a Mahalanobis distance of 1.

For the multivariate case, the variance may differ for different directions in the data space, thus instead of dividing by a single value, the data is scaled by the inverse of the covariance matrix $\boldsymbol{\Sigma}$. The Mahalanobis distance for a d-dimensional sample $\boldsymbol{y}$ to a point set $\boldsymbol{X}$ with mean $\bar{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{\Sigma}$ is

$$\delta(\boldsymbol{X}, \boldsymbol{y}) = \sqrt{(\boldsymbol{y}-\bar{\boldsymbol{x}})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y}-\bar{\boldsymbol{x}})}. \tag{A.16}$$

Because it encapsulates statistical properties of the reference sample set, the Mahalanobis distance can be used to assess the probability of a new point belonging to the set. Specifically, if the reference samples are known to stem from a multivariate Gaussian distribution, the squared distance terms in (A.16) follow a χ^2 distribution [38]. For a chosen probability, the bounding squared Mahalanobis distance can then be looked up from a χ^2 table.

Bibliography

Articles, Books

[1] A. S. Acampora and J. H. Winters. Three-dimensional ultrasonic vision for robotic applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):291–303, 1989.

[2] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 149–155, 2004.

[3] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[4] B. A. am Ende. 3D mapping of underwater caves. *IEEE Computer Graphics and Applications*, 21(2):14–20, 2001.

[5] T. Anderson and D. Darling. Asymptotic theory of certain ”goodness-of-fit” criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212, 1952.

[6] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the Fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 271–280, 1993.

[7] G. A. Atkinson and E. R. Hancock. Shape estimation using polarization and shading from two views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):2001–2017, 2007.

[8] A. Bab-Hadiashar and N. Gheissari. Range image segmentation using surface selection criterion. *IEEE Transactions on Image Processing*, 15(7):2006–2018, 2006.

[9] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, 1982.

[10] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38:91–109, 2001.

[11] B. Bartczak, K. Koeser, F. Woelk, and R. Koch. Extraction of 3D freeform surfaces as visual landmarks for real-time tracking. *Journal of Realtime Image Processing*, 2(2-3):81–101, November 2007.

[12] M. Basseville. Information: entropies, divergences et moyennes (in french). Technical report, IRISA, 1996.

[13] R. Benlamri. Range image segmentation of scenes with occluded curved objects. *Pattern Recognition Letters*, 21(12):1051–1060, 2000.

[14] J. Beraldin, F. Blais, L. Cournoyer, G. Godin, and M. Rioux. Active 3D sensing. Technical report, Institute for Information Technology, National Research Council Canada, 2000.

[15] M. Bertozzi, A. Broggi, C. Caraffi, M. Del Rose, M. Felisa, and G. Vezzoni. Pedestrian detection by means of far-infrared stereo vision. *Journal of Computer Vision and Image Understanding*, 106(2-3):194–204, 2007.

[16] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 1988.

[17] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[18] B. Bhanu, S. Lee, C. Ho, and T. Henderson. Range data processing: representation of surfaces by edges. In *Proceedings of the Eighth International Conference on Pattern Recognition*, pages 236–238, 1986.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[20] M. Boshra and M. A. Ismail. Recognition of occluded polyhedra from range images. *Pattern Recognition*, 33(8):1351–1367, 2000.

[21] J. Branch, F. Prieto, and P. Boulanger. Automatic hole-filling of triangular meshes using local radial basis function. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 727–734, 2006.

[22] J. E. Bresenham. Incremental line compaction. *Computer Journal*, 25(1):116–120, 1982.

[23] M. J. Brooks. *Shape from shading.* MIT Press Series Of Artificial Intelligence, 1989.

[24] R. Brooks. New approaches to robotics. *Science*, 253:1227–1232, 1991.

[25] M. Buss, M. Beetz, and D. Wollherr. CoTeSys – Cognition for Technical Systems. *International Journal of Assistive Robotics and Mechatronics*, 8:1–10, 2006.

[26] T. S. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *Proceedings of the 11th IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[27] R. M. Cesar, E. Bengoetxea, I. Bloch, and P. L. aga. Inexact graphmatching for model-based recognition: Evaluation and comparison of optimization algorithms. *Pattern Recognition*, 38:2099–2113, 2005.

[28] R. Chaine, P.-M. Gandoin, and C. Roudet. Mesh connectivity compression using convection reconstruction. In *Proceedings of the ACM symposium on Solid and physical modeling*, pages 41–49, New York, NY, USA, 2007. ACM.

[29] K. M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[30] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2001.

[31] M. Cummins and P. Newman. Highly scalable appearance-only slam – FAB-MAP 2.0. In *Proceedings of the International Conference on Robotics, Science, and Systems*, 2009.

[32] F. Dell'Acqua and R. Fisher. Reconstruction of planar surfaces behind occlusions in range images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(4):569–575, 2002.

[33] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun. Structure from motion without correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000.

[34] T. K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Eurographics Symposium on Point-Based Graphics*, 2005.

[35] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience Publication, 2000.

[36] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard. Unsupervised discovery of object classes from range data using latent Dirichlet allocation. In *Proceedings of the International Conference on Robotics, Science, and Systems*, 2009.

[37] L. Euler. Recherches sur la courbure des surfaces. *Memoires de l'academie des sciences de Berlin*, 16:119–143, 1760.

[38] M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions (2nd ed.).* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[39] P. Fechteler and P. Eisert. Adaptive color classification for structured light systems. *Computer Vision and Pattern Recognition Workshop*, 0:1–7, 2008.

[40] R. B. Fisher. *From surfaces to objects: computer vision and three dimensional scene analysis.* John Wiley & Sons, Inc., New York, NY, USA, 1989.

[41] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[42] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2004.

[43] S. Gächter, A. Harati, and R. Siegwart. Incremental object part detection toward object classification in a sequence of noisy range images. In *Proceedings of the International Conference on Robotics and Automation*, 2008.

[44] S. Gächter, A. Harati, and R. Siegwart. Structure verification toward object classification using a range camera. In *Proc. of the 10th International Conference on Intelligent Autonomous Systems (IAS)*, 2008.

[45] C. F. Gauss. Disquisitiones generales circa superficies curvas (in latin). In *Werke: Wahrscheinlichkeitsrechnung und Geometrie*, volume 4. Dieterich, 1827.

[46] R. Gens and J. Vangenderen. SAR interferometry: Issues, techniques, applications. *International Journal of Remote Sensing*, 17(10):1803–1835, 1996.

[47] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, 2003.

[48] D. Gibson. Accuracy problems in ultrasonic and light-beam rangefinders. *Journal of the BCRA Cave Radio and Electronics Group*, 6:9, 1991.

[49] G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[50] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum*, 19(3):467–478, 2000.

[51] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, 1971.

[52] G.Podnar. The neptune mobile robot. Technical Report CMU-RI-TR-86-04, Robotics Institute, Carnegie Mellon University, 1985.

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

[54] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision (2nd ed.).* Cambridge University Press, Cambridge, UK, 2004.

[55] M. Hebert. Active and passive range sensing for robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[56] A. Hertzmann and S. M. Seitz. Example-based photometric stereo: Shape reconstruction with general, varying BRDFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, 2005.

[57] A. Hlaoui and S. Wang. A new algorithm for inexact graph matching. In *Proceedings of the 16th International Conference on Pattern Recognition*, 2002.

[58] R. Hoffman and A. K. Jain. Segmentation and classification of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):608–620, 1987.

[59] I. Hofman and R. Jarvis. Object recognition via attributed graph matching. In *Proceedings of the Australian Conference on Robotics and Automation*, 2000.

[60] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

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

[62] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[63] B. Horn. Obtaining shape from shading information. In P. Winston, editor, *The Psychology of Computer Vision*, New York, 1975. McGraw-Hill.

[64] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[65] J. Huang and C.-H. Menq. Automatic data segmentation for geometric feature extraction from unorganized 3-d coordinate points. *IEEE Transactions on Robotics and Automation*, 17:268–279, 2001.

[66] T. S. Huang and A. N. Netravali. Motion and structure from feature correspondences: a review. *Proceedings of the IEEE*, 82(2):252–268, 1994.

[67] D. Huber and M. Hebert. Fully automatic registration of multiple 3D data sets. *Image and Vision Computing*, 21(1):637–650, 2003.

[68] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann. Grasping known objects with humanoid robots: A box-based approach. In *Proceedings of the International Conference on Advanced Robotics*, 2009.

[69] R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:122–139, 1983.

[70] T. Jebara, A. Azarbayejani, and A. Pentland. 3D structure from 2D motion. *IEEE Signal Processing Magazine*, 16(3):66–84, 1999.

[71] X. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Machine Vision and Applications*, 7(2):115–122, 1994.

[72] X. Y. Jiang, H. Bunke, and U. Meier. Fast range image segmentation using high-level segmentation primitives. In *WACV '96: Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision (WACV '96)*, 1996.

[73] S. Jin, R. R. Lewis, and D. West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1-2):71–82, 2005.

[74] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):433–449, 1999.

[75] I. T. Jolliffe. *Principal Component Analysis (2nd ed.)*. Springer, 2002.

[76] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[77] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier, New York, NY, USA, 1996.

[78] I. Khalifa, M. Moussa, and M. Kamel. Range image segmentation using local approximation of scan lines with application to CAD model acquisition. *Machine Vision and Applications*, 13(5-6):263–274, 2003.

[79] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry, 2nd ed.* Wiley-Interscience., 1996.

[80] K. Koeser, B. Bartczak, and R. Koch. Robust GPU-assisted camera tracking using free-form surface models. *Journal of Realtime Image Processing*, 2(2-3):133–147, November 2007.

[81] A. Kolb, E. Barth, and R. Koch. ToF-sensors: New dimensions for realism and interactivity. In *Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on ToF Camera based Computer Vision (TOF-CV)*, 2008.

[82] D. Kragic and M. Vincze. Vision for robotics. *Foundations and Trends in Robotics*, 1(1):1–78, 2009.

[83] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[84] M. Laurenzis, F. Christnacher, and D. Monnin. Long-range three-dimensional active imaging with superresolution depth mapping. *Optics Letters*, 32(21):3146–3148, 2007.

[85] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9:23–29, 1977.

[86] B. Leibe, A. Leonardis, and B. Schiele. An implicit shape model for combined object categorization and segmentation. In *Towards Category-Level Object Recognition*, pages 496–510. Springer, 2006.

[87] M. A. Lippow, L. P. Kaelbling, and T. Lozano-Perez. Learning grammatical models for object recognition. In *Dagstuhl Seminar Proceedings, Logic and Probability for Scene Interpretation*, 2008.

[88] J. R. Llata, E. G. Sarabia, and J. P. Oria. Three-dimensional robotic vision using ultrasonic sensors. *Journal of Intelligent and Robotic Systems*, 33(3):267–284, 2002.

[89] A. Lobay and D. A. Forsyth. Shape from texture without boundaries. *International Journal of Computer Vision*, 67(1):71–91, 2006.

[90] E. Magid, O. Soldea, and E. Rivlin. A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data. *Computer Vision and Image Understanding*, 107(3):139–159, 2007.

[91] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, 1936.

[92] D. Marshall, G. Lukacs, and R. Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 23(3):304–314, 2001.

[93] Z. C. Marton, R. B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[94] A. M. McIvor and R. J. Valkenburg. A comparison of local surface geometry estimation methods. *Machine Vision and Applications*, 10(1):17–26, 1997.

[95] A. Medina, F. Gayá, and F. del Pozo. Compact laser radar and three-dimensional camera. *Journal of the Optical Society of America A*, 23(4):800–805, 2006.

[96] O. Michel. Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[97] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. D. Schutter. Active sensing for robotics - a survey. In *Proceedings of the 5th International Conference On Numerical Methods and Applications*, pages 316–324, 2002.

[98] M. J. Milroy, C. Bradley, and G. W. Vickers. Segmentation of a wrap-around model using an active contour. *Computer-Aided Design*, 29(4):299–320, 1997.

[99] M. L. Minsky. *The Society of Mind*. Simon and Schuster, New York, NY, USA, 1985.

[100] A. A. Miranda, Y.-A. Borgne, and G. Bontempi. New routes from minimal approximation error to principal components. *Neural Processing Letters*, 27(3):197–207, 2008.

[101] M. Montemerlo and S. Thrun. A multi-resolution pyramid for outdoor robot terrain perception. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2004.

[102] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1985.

[103] E. Natonek. Fast range image segmentation for servicing robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.

[104] S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994.

[105] H. G. Nguyen, A. G. Kogut, A. R. Barua, A. A. Burmeister, N. Pezeshkian, A. D. Powell, A. N. Farrington, M. Wimmer, B. B. Cicchetto, B. C. Heng, and V. R. B. A segway rmp-based robotic transport system. In *SPIE Proceedings 5609: Mobile Robots XVII*, pages 26–28, 2004.

[106] M. Novotni and R. Klein. Shape retrieval using 3D Zernike descriptors. *Computer Aided Design*, 36:1047–1062, 2004.

[107] D. OuYang and H.-Y. Feng. On the normal vector estimation for point cloud data from smooth surfaces. *Computer-Aided Design*, 37(10):1071–1079, 2005.

[108] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization*, 2002.

[109] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[110] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[111] M. W. Powell, K. W. Bowyer, X. Jiang, and H. Bunke. Comparing curved-surface range image segmenters. In *Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[112] E. Prados, F. Camilli, and O. Faugeras. A unifying and rigorous shape from shading method adapted to realistic data and applications. *Journal of Mathematical Imaging and Vision*, 25(3):307–328, 2006.

[113] R. W. Prager, U. Z. Ijaz, A. H. Gee, and G. M. Treece. Three-dimensional ultrasound imaging. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 224:193–223, 2009.

[114] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977.

[115] T. Rabbani, F. van den Heuvel, and G. Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):248–253, 2006.

[116] N. S. Raja. *Obtaining generic parts from range images using a multi-view representation.* PhD thesis, Michigan State University, East Lansing, MI, USA, 1992.

[117] G. Roth and M. D. Levine. Extracting geometric primitives. *Computer Vision and Image Understanding*, 58(1):1–22, 1993.

[118] S. Ruiz-Correa, L. G. Shapiro, and M. Meila. A new paradigm for recognizing 3D object shapes from range data. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[119] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927 – 941, 2008.

[120] F. Sadjadi. Passive 3D reconstruction using polarimetric infrared imagery. In *The 20th Annual Meeting of the IEEE Lasers and Electro-Optics Society*, 2007.

[121] R. Sagawa, T. Oishi, A. Nakazawa, R. Kurazume, and K. Ikeuchi. Iterative refinement of range images with anisotropic error distribution. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2002.

[122] A. Sappa and M. Devy. Fast range image segmentation by an edge detection strategy. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling*, 2001.

[123] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. *EEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:195, 2003.

[124] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001.

[125] B. Schiele and J. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36:31–52, 2000.

[126] S. R. Schmidt-Rohr, M. Lösch, Z. Xue, and R. Dillmann. Hardware and software architecture for robust autonomous behavior of a domestic robot assistant. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, 2008.

[127] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[128] J. Schoenwald. Strategies for robotic sensing using acoustics. In *Ultrasonics Symposium Proceedings*, 1985.

[129] S. Schuon, C. Theobalt, J. Davis, and S. Thrun. High-quality scanning using time-of-flight depth superresolution. *Computer Vision and Pattern Recognition Workshop*, 0:1–7, 2008.

[130] W. I. Sellers and A. T. Chamberlain. Ultrasonic cave mapping. *Journal of Archaeological Science*, 25(9):867–873, 1998.

[131] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[132] R. Sheh, N. Jamali, M. W. Kadous, and C. Sammut. A low-cost, compact, lightweight 3D range sensor. In *Proceedings of the Australian Conference on Robotics and Automation*, 2006.

[133] J. Shin, S. Gächter, A. Harati, C. Pradalier, and R. Siegwart. Object classification based on a geometric grammar with a range camera. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, 2009.

[134] B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[135] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*, 2005.

[136] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, 2003.

[137] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH Conference Proceedings*, 2006.

[138] S. Srinivasa, C. Ferguson, D. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe. Herb: A Home Exploring Robotic Butler. *Autonomous Robots*, 28:5–20, 2009.

[139] B. Stanczyk. *Development and Control of an Anthropomorphic Telerobotic System*. Number 1122 in VDI Fortschritt-Berichte Reihe 8. VDI- Verlag Düsseldorf, 2007.

[140] M. Strand and R. Dillmann. Using an attributed 2D-grid for next-best-view planning on 3D environment data for an autonomous robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.

[141] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision*, 77(1):291–330, 2008.

[142] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[143] G. J. Székely and M. L. Rizzo. A new test for multivariate normality. *Journal of Multivariate Analysis*, 93(1):58–80, 2005.

[144] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[145] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* The MIT Press, 2005.

[146] P. H. S. Torr and A. Zisserman. MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[147] R. Triebel and W. Burgard. Recovering the shape of objects in 3D point clouds with partial occlusions. In *Proceedings of the 6th International Conference on Field and Service Robotics*, 2007.

[148] E. Trucco and R. B. Fisher. Experiments in curvature-based segmentation of range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):177–182, 1995.

[149] S. Ullman. The interpretation of structure from motion. In *MIT AI Memo*, 1976.

[150] M. Vanco. *A Direct Approach for the Segmentation of Unorganized Points and Recognition of Simple Algebraic Surfaces.* PhD thesis, University of Technology Chemnitz, 2003.

[151] C. Wang, H. Tanahashi, H. Hirayu, Y. Niwa, and K. Yamamoto. Comparison of local plane fitting methods for range data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 663–669, 2001.

[152] J. Wang and M. M. Oliveira. A hole-filling strategy for reconstruction of smooth surfaces in range images. In *Proceedings of the 16th Brazilian Symposium on Computer Graphics and Image Processing*, 2003.

[153] Y. F. Wang and J. K. Aggarwal. An overview of geometric modeling using active sensing. *IEEE Control Systems Magazine*, 3(2):7–13, 1988.

[154] M. A. Wani and H. R. Arabnia. Parallel edge-region-based segmentation algorithm targeted at reconfigurable multiring network. *The Journal of Supercomputing*, 25(1):43–62, 2003.

[155] M. A. Wani and B. G. Batchelor. Edge-region-based segmentation of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):314–319, 1994.

[156] S. Watanabe and M. Yoneyama. An ultrasonic visual sensor for three-dimensional object recognition using neural networks. *IEEE Transactions on Robotics and Automation*, 8(2):240–249, 1992.

[157] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *MIT Press Series Of Artificial Intelligence*, Shape from shading:513–531, 1989.

[158] O. Wulf and B. Wagner. Fast 3D-scanning methods for laser measurement systems. In *Proceedings of the 14th International Conference on Control Systems and Computer Science (CSCS14)*, 2003.

[159] G. Yahav, G. J. Iddan, and D. Mandelboum. 3D imaging camera for gaming application. In *International Conference on Consumer Electronics, Digest of Technical Papers*, 2007.

[160] K. Yamazaki, M. Tomono, and T. Tsubouchi. Picking up an unknown object through autonomous modeling and grasp planning by a mobile manipulator. In *Results of the 6th International Conference on Field and Service Robotics*, 2007.

[161] K. Yamazaki, R. Ueda, S. Nozawa, Y. Mori, T. Maki, N. Hatao, K. Okada, and M. Inaba. A demonstrative research for daily assistive robots on tasks of cleaning and tidying up rooms. In *Proceedings of The 14th Robotics Symposia*, 2009.

[162] M. Yang and E. Lee. Segmentation of measured point data using a parametric quadric surface approximation. *Computer-Aided Design*, 31(7):449–457, 1999.

[163] L. Zhang, B. Curless, and S. M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, 2002.

[164] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[165] S. Zhang and P. S. Huang. High-resolution, real-time three-dimensional shape measurement. *Optical Engineering*, 45(12):1–8, 2006.

Own Publications

[166] A. Bauer, **K. Klasing**, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss. The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 1(2):127–140, 2009.

[167] G. Lidoris, **K. Klasing**, A. Bauer, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss. The Autonomous City Explorer project: Aims and system overview. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[168] **K. Klasing**. Parallelized sampling-based path planning for tree-structured rigid robots. Master's thesis, Institute of Automatic Control Engineering, TU München, 2009.

[169] **K. Klasing**, D.Althoff, D. Wollherr, and M. Buss. Comparison of surface normal estimation methods for range sensing applications. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[170] **K. Klasing**, G. Lidoris, A. Bauer, F. Rohrmüller, D. Wollherr, and M. Buss. The Autonomous City Explorer: Towards semantic navigation in urban environments. In *Proceedings of the 1st International Workshop on Cognition for Technical Systems*, 2008.

[171] **K. Klasing**, D. Wollherr, and M. Buss. Cell-based probabilistic roadmaps (CPRM) for efficient path planning in large environments. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, 2007.

[172] **K. Klasing**, D. Wollherr, and M. Buss. A clustering method for efficient segmentation of 3D laser data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[173] **K. Klasing**, D. Wollherr, and M. Buss. Joint dominance coefficients: A sensitivity-based measure for ranking robotic degrees of freedom. In T. Kröger and F. Wahl, editors, *Advances in Robotics Research*, pages 1–10. Springer Verlag, 2009.

[174] **K. Klasing**, D. Wollherr, and M. Buss. Realtime segmentation of range data using continuous nearest neighbors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

Manuals, Software, Web References

[175] CGAL - Computational Geometry Algorithms Library. www.cgal.org/, 2010.

[176] GNU Scientific Library. http://www.gnu.org/software/gsl/, 2010.

[177] 2d3 Ltd. Company website. http://www.2d3.com, 2010.

[178] AGE Solutions S.r.l. Maestro 3D dental scanner technical specifications. http://www.maestro3d.com/index.asp?page0=container&page1=maestro3d.dental.scanner.technical, 2010.

[179] Artec Group. Artec 3D scanner specification. http://artec-group.com/3dscanning/broadway/specification.html, 2010.

[180] Canesta, Inc. Introduction to 3D vision in CMOS. http://www.canesta.com/assets/pdf/technicalpapers/Canesta101.pdf, 2008.

[181] E. Coumans. The bullet physics library. http://www.bulletphysics.com, 2010.

[182] Focus Robotics. nDepth vision system datasheet. http://www.focusrobotics.com/docs/focus_ndepth_pci_brief.pdf, 2010.

[183] Friendly Robotics. Robomow - company website. http://www.robomow.com, 2010.

[184] GFMesstechnik GmbH. TopoCAM - high-end optical 3D measurement of components. http://www.gfm3d.com/index.php?view=article&id=102, 2010.

[185] Hokuyo Automatic Co., Ltd. URG-04LX scanning range finder datasheet. http://www.sentekeurope.com/docs/datasheet/URG-04LX-UG01%20datasheet.pdf, 2010.

[186] Hokuyo Automatic Co., Ltd. UTM-30LX scanning range finder datasheet. http://www.sentekeurope.com/docs/datasheet/UTM-30LX%20datasheet.pdf, 2010.

[187] iRobot Corporation. iRobot: Cleaning robots - company website. http://store.irobot.com/shop/index.jsp, 2010.

[188] K. Klasing. PP - a simple sampling-based path planning library written in C++. http://www.lsr.ei.tum.de/research/software/pp/, 2009.

[189] Kongsberg SIM AS. Coin3D - 3D graphics developer kit. http://www.coin3d.org, 2010.

[190] Leica Geosystems AG. Leica ScanStation C10 data sheet. http://www.leica-geosystems.com/common/shared/downloads/inc/downloader.asp?id=11889, 2010.

[191] Mesa Imaging AG. Swiss Ranger^{TM} SR4000 data sheet. http://www.mesa-imaging.ch/dlm.php?fname=pdf/SR4000_Data_Sheet.pdf, 2010.

[192] Microsoft Corp. Microsoft robotics developer studio. http://www.microsoft.com/Robotics/, 2008.

[193] Microsoft Corporation / 3DV Systems. ZCam product data sheet. http://www.engadget.com/photos/microsofts-project-natal-roots-revealed-3dv-systems-zcam-2/2056911/, 2008.

[194] M. Montemerlo, N. Roy, S. Thrun, D. Haehnel, C. Stachniss, and J. Glover. The Carnegie Mellon robot navigation toolkit (CARMEN). http://carmen.sourceforge.net, 2008.

[195] D. M. Mount. ANN programming manual, version 1.1. Technical report, Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, 2010.

[196] D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/~mount/ANN/, 2010.

[197] Next Engine, Inc. NextEngine desktop 3D scanner data sheet. https://www.nextengine.com/HD_DataSheet.pdf, 2010.

[198] PMD Technologies GmbH. PMD[vision]®CamCube 2.0 data sheet. http://www.pmdtec.com/fileadmin/pmdtec/downloads/documentation/datasheet_camcube.pdf, 2010.

[199] Point Grey Research, Inc. Product catalog stereo. http://www.ptgrey.com/products/Point_Grey_stereo_catalog.pdf, 2010.

[200] PrimeSense Ltd. The PrimeSensor^{TM}reference design 1.08. http://www.primesense.com/files/PDF/PrimeSensor_RD1.08_Datasheet.PDF, 2010.

[201] SICK AG. The LMS-400 measurement system. www.sick-automation.ru/images/File/pdf/LMS%20400.pdf, 2006.

[202] SICK AG. LMS200/211/221/291 laser measurement systems technical description. www.mysick.com/saqqara/get.aspx?id=IM0012759, 2006.

[203] E. Simoncelli. Least squares optimization, lecture notes. http://www.cns.nyu.edu/~eero/teaching.html, 2003.

[204] Solutionix Corp. Rexcan DS2 measurement for quality and speed. http://www.solutionix.com/product/file/RexcanDS2_Brochure_en_down.zip, 2010.

[205] Solutionix Corp. Rexcan III 3D measurement for quality and speed. http://www.solutionix.com/product/file/Rexcan3_brochure_eng.zip, 2010.

[206] Surveyor Corporation. Surveyor stereo vision system SVS specifications. http://www.surveyor.com/stereo/, 2010.

[207] The Geometry Center, University of Minnesota. QHull. http://www.qhull.org/, 2010.

[208] Velodyne Lidar, Inc. HDL-64E S2 data sheet. http://www.velodyne.com/lidar/products/brochure/HDL-64E%20S2%20datasheet_lowres.pdf, 2006.

[209] Vicon. Company website. www.vicon.com, 2010.

[210] Videre Design LLC. STH-MDCS3 Datasheet. http://www.videredesign.com/assets/docs/datasheets/sth-mdcs3-data-sheet.pdf, 2010.

[211] Willow Garage, Inc. PR2 Robot - technical specifications. http://www.willowgarage.com/pages/robots/technical-specs, 2010.