Three-dimensional Laser-based Classification in Outdoor Environments

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Jens Behley

aus

Cottbus

Bonn, 2013

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

Erstgutachter: Prof. Dr. Armin B. Cremers, Bonn
Zweitgutachter: PD Dr. Volker Steinhage, Bonn
Tag der Promotion: 30.01.2014
Erscheinungsjahr: 2014

Abstract

Robotics research strives for deploying autonomous systems in populated environments, such as inner city traffic. Autonomous cars need a reliable collision avoidance, but also an object recognition to distinguish different classes of traffic participants. For both tasks, fast three-dimensional laser range sensors generating multiple accurate laser range scans per second, each consisting of a vast number of laser points, are often employed. In this thesis, we investigate and develop classification algorithms that allow us to automatically assign semantic labels to laser scans. We mainly face two challenges: (1) we have to ensure consistent and correct classification results and (2) we must efficiently process a vast number of laser points per scan. In consideration of these challenges, we cover both stages of classification — the feature extraction from laser range scans and the classification model that maps from the features to semantic labels.

As for the feature extraction, we contribute by thoroughly evaluating important state-of-the-art histogram descriptors. We investigate critical parameters of the descriptors and experimentally show for the first time that the classification performance can be significantly improved using a large support radius and a global reference frame.

As for learning the classification model, we contribute with new algorithms that improve the classification efficiency and accuracy. Our first approach aims at deriving a consistent point-wise interpretation of the whole laser range scan. By combining efficient similarity-preserving hashing and multiple linear classifiers, we considerably improve the consistency of label assignments, requiring only minimal computational overhead compared to a single linear classifier.

In the last part of the thesis, we aim at classifying objects represented by segments. We propose a novel hierarchical segmentation approach comprising multiple stages and a novel mixture classification model of multiple bag-of-words vocabularies. We demonstrate superior performance of both approaches compared to their single component counterparts using challenging real world datasets.

Überblick

Ziel des Forschungsbereichs Robotik ist der Einsatz autonomer Systeme in natürlichen Umgebungen, wie zum Beispiel innerstädtischem Verkehr. Autonome Fahrzeuge benötigen einerseits eine zuverlässige Kollisionsvermeidung und andererseits auch eine Objekterkennung zur Unterscheidung verschiedener Klassen von Verkehrsteilnehmern. Verwendung finden vorallem drei-dimensionale Laserentfernungssensoren, die mehrere präzise Laserentfernungsscans pro Sekunde erzeugen und jeder Scan besteht hierbei aus einer hohen Anzahl an Laserpunkten. In dieser Dissertation widmen wir uns der Untersuchung und Entwicklung neuartiger Klassifikationsverfahren zur automatischen Zuweisung von semantischen Objektklassen zu Laserpunkten. Hierbei begegnen wir hauptsächlich zwei Herausforderungen: (1) wir möchten konsistente und korrekte Klassifikationsergebnisse erreichen und (2) die immense Menge an Laserdaten effizient verarbeiten. Unter Berücksichtigung dieser Herausforderungen untersuchen wir beide Verarbeitungsschritte eines Klassifikationsverfahrens — die Merkmalsextraktion unter Nutzung von Laserdaten und das eigentliche Klassifikationsmodell, welches die Merkmale auf semantische Objektklassen abbildet.

Bezüglich der Merkmalsextraktion leisten wir ein Beitrag durch eine ausführliche Evaluation wichtiger Histogrammdeskriptoren. Wir untersuchen kritische Deskriptorparameter und zeigen zum ersten Mal, dass die Klassifikationsgüte unter Nutzung von großen Merkmalsradien und eines globalen Referenzrahmens signifikant gesteigert wird.

Bezüglich des Lernens des Klassifikationsmodells, leisten wir Beiträge durch neue Algorithmen, welche die Effizienz und Genauigkeit der Klassifikation verbessern. In unserem ersten Ansatz möchten wir eine konsistente punktweise Interpretation des gesamten Laserscans erreichen. Zu diesem Zweck kombinieren wir eine ähnlichkeitserhaltende Hashfunktion und mehrere lineare Klassifikatoren und erreichen hierdurch eine erhebliche Verbesserung der Konsistenz der Klassenzuweisung bei minimalen zusätzlichen Aufwand im Vergleich zu einem einzelnen linearen Klassifikator.

Im letzten Teil der Dissertation möchten wir Objekte, die als Segmente repräsentiert sind, klassifizieren. Wir stellen eine neuartiges hierarchisches Segmentierungsverfahren und ein neuartiges Klassifikationsmodell auf Basis einer Mixtur mehrerer bag-of-words Vokabulare vor. Wir demonstrieren unter Nutzung von praxisrelevanten Datensätzen, dass beide Ansätze im Vergleich zu ihren Entsprechungen aus einer einzelnen Komponente zu erheblichen Verbesserungen führen.

Acknowledgments

First of all, I would like to thank Prof. Dr. Armin B. Cremers for his support during the years of research and advice during this time. I furthermore want to express my gratitude to PD Dr. Volker Steinhage, who often discussed earlier drafts of my writings with me and put my research ideas in perspective.

The presented research in this thesis was mainly funded by the Fraunhofer FKIE and would not be possible without the technical support of the Unmanned Systems group. I would like to thank Dr. Dirk Schulz for fruitful discussions on the projects. Thanks to Achim Königs, Ansgar Tessmer, Timo Röhling, Frank Höller, Jochen Welle, and Michael Brunner for technical support with the Longcross robot and the Velodyne laser range scanner.

I thank Florian Schöler, Dr. Daniel Seidel, and Marcell Missura for long and invaluable discussions on my research topic. I also want to thank Stavros Manteniotis, Dr. Andreas Baak, Marcell Missura, Florian Schöler, Shahram Faridani, and Jenny Balfer, who helped with proofreading of the thesis and gave many, many comments that certainly improved the presentation and structure of the thesis. Thanks to Sabine Kühn, Eduard 'Edi' Weber, and Dr. Fabian Weber from the Food Technology department, who often cheered me up and introduced me to the wonders of food technology. A special thanks goes to our fantastic technical support of the department, the SGA.

A heartful thank-you to my parents, my brother, and Jenny Balfer for their encouragement and also patience during the period of writing the thesis.

Mathematical Notation

In course of the following chapters, we need some mathematical entities, which we denote consistently throughout the text. Most of these conventions are commonly used in contemporary books on machine learning. Therefore, the notation will look familiar to many readers. In order to enhance the readability, simplifications to the notation will be introduced in the corresponding chapters.

We often refer to sets, which we denote by calligraphic upper-case letters, such as $\mathcal{A}, \mathcal{X}, \mathcal{Y}$. Elements of these sets, $\mathcal{X} = \{x_1, \ldots, x_n\}$, are denoted by the corresponding Roman lower-case letters indexed by a number. The cardinality of a set is denoted by $|\mathcal{X}| = N$, where N is the number of elements in set $\mathcal{X}$. If we refer to multiple elements of a set, such as $\{x_j, x_{j+1}, x_{j+2}, \ldots, x_{k-1}, x_k\}$, we use the shorthand $x_{j:k}$. Common number systems – natural numbers $\mathbb{N}$ including 0, integers $\mathbb{Z}$, and real numbers $\mathbb{R}$ – are denoted by upper-case blackboard bold letters.

We use bold letters to distinguish scalars from vectors and matrices as explained in the following. A matrix is referred to by a Roman upper-case bold letter, such as $\mathbf{M} \in \mathbb{R}^{n \times m}$, where $n \times m$ shows the dimensions of the matrix, i.e., n rows and m columns. Vectors are denoted by Roman lower-case bold letters such as $\boldsymbol{u} \in \mathbb{R}^{1 \times m}$ or $\boldsymbol{v} \in \mathbb{R}^{n \times 1}$, where we made explicit that $\boldsymbol{u}$ is a row vector and $\boldsymbol{v}$ is a column vector. If not stated otherwise in the text, we use column vectors and therefore write $\boldsymbol{v} \in \mathbb{R}^n$ instead of $\boldsymbol{v} \in \mathbb{R}^{n \times 1}$. As common in literature, we use T to denote the transposition of a matrix $\mathbf{M}^T$ or a vector $\boldsymbol{v}^T$. Elements of a matrix and a vector are indexed by $\mathbf{M}^{(i,j)}$ or $\boldsymbol{v}^{(i)}$. Similar to sets, we use the shorthand $\boldsymbol{v}^{(j:k)}$ to refer to a sequence of elements, starting at index j and ending with index k.

Contents

Mathematical Notation **vii**

1 Introduction **1**
1.1 Contributions of the Thesis 3
1.2 Structure of the Thesis 4

2 Fundamentals **5**
2.1 Three-dimensional Point Cloud Processing 5
2.1.1 Data Acquisition 7
2.1.2 Neighbor Search 9
2.1.3 Normal Estimation 11
2.2 Classification 12
2.2.1 Softmax Regression 17
2.2.2 k-Nearest Neighbor Classification 19
2.2.3 Model Assessment 19
2.3 Summary 21

3 Histogram Descriptors for Laser-based Classification **23**
3.1 Related Work 25
3.2 Histogram Descriptors 26
3.3 Reference Frame and Reference Axis 30
3.4 Experimental Setup 31
3.5 Results and Discussion 35
3.6 Summary 43

4 Efficient hash-based Classification **45**
4.1 Related Work 47
4.2 Spectrally Hashed Softmax Regression 48
4.2.1 Spectral Hashing 49
4.2.2 Combining Spectral Hashing and Softmax Regression 51
4.3 Experimental Evaluation 53
4.4 Summary 60

5 Segment-based Classification **63**
5.1 Related Work . . . 66
5.2 Fundamentals . . . 68
5.2.1 Segmentation . . . 68
5.2.2 Bag-of-words Representation . . . 71
5.3 Approach . . . 72
5.3.1 Hierarchical Segmentation . . . 72
5.3.2 Learning a mixture of bag-of-words . . . 75
5.3.3 Hierarchical Non-maximum Suppression . . . 78
5.4 Improving the Efficiency . . . 78
5.4.1 Point Sub-Sampling . . . 80
5.4.2 Descriptor Sub-Sampling . . . 80
5.5 Experiments . . . 81
5.5.1 Bounding box overlap . . . 85
5.5.2 Detection performance . . . 86
5.5.3 Classification performance . . . 88
5.5.4 Runtime performance . . . 88
5.6 Summary . . . 90

6 Conclusions **93**

A Probability Theory **97**

B Additional Results **101**

List of Figures **105**

List of Tables **107**

Bibliography **109**

Index **121**

Chapter 1

Introduction

Many successful applications of industrial and automation robotics rely on robot-centered workspaces. In such environments, the robots can perform tasks with limited or even without knowledge about their vicinity. For instance, a manufacturing robot assembling cars always moves its manipulator in a pre-defined sequence without collisions. As another example, a transport robot in a large warehouse follows specified obstacle-free routes, which might even be marked by small metal wires in the ground. After arriving at the target position, the package to be transported is identified using a bar code. In these examples, the whole environment is specifically tailored to the abilities of the robot. In consequence, the robot needs only a rudimentary perception.

In addition, the state of the world changes only if the robot performs an action such as lifting a part of a car or removing a package from the storage rack. Thus, all parts always lie at a specific location in a certain orientation; packages stay at the same location in the storage rack. The environment is static and the intended operation of the robot can be seriously interfered if something happens outside of the robot's control.

In contrast to these industrial applications, the aim of modern robotics and artificial intelligence research is the development of autonomous systems, which are able to operate in natural environments without the need to change the entire structure by augmenting the environment with robot-suited markers or similar modifications. These systems should be able to act in highly dynamic environments, where the state not only changes by actions of the system, but also externally by other actors. The world state includes also other moving agents, such as vehicles, pedestrians, or other robots. For such intelligent systems, rich sensor input is essential — the robot needs to detect changes and to update its internal world state continuously. Thus, a major part of research focuses on the efficient and reliable robot perception incorporating potentially multiple sources of sensor input.

Lately, especially the development of self-driving cars attracted increasing interest in the robotics community. Since the early nineties self-driving cars were developed that can handle more and more complex tasks and scenarios. The development was recently further intensified by competitions aiming at developing autonomous cars able to drive in the desert [Thrun et al., 2006] or urban environment [Urmson et al., 2008]. In such environments, it is self-explanatory that perceiving autonomous systems capable of operating in natural, cluttered, and dynamic environments are needed. Major automobile companies, such as BMW, Volkswagen, Mercedes Benz, or Toyota, are working towards self-driving cars and some of the innovations that were developed in this context found already its application in current models.

The main requirement for self-driving cars is the safe and collision-free navigation — we must ensure at all times that the system neither harms any other traffic participants nor destroys itself. Effective collision avoidance needs the distance to objects and roboticists mainly employ laser range sensors, because of their robustness and precision. The recent emergence of fast three-dimensional laser rangefinders made it possible to investigate also other applications, such as mapping and localization [Levinson and Thrun, 2010, Moosmann and Stiller, 2010], object tracking [Petrovskaya and Thrun, 2009, Schöler et al., 2011] and object recognition [Munoz et al., 2009a, Xiong et al., 2011]. The interest for other applications using three-dimensional laser range data was mainly driven by the richer information and the higher update rate of the sensors, which made it possible to obtain more than 100,000 range measurements in a fraction of a second. Laser range scans are an interesting alternative to images, as they are invariant to illumination and directly offer shape information. Consequently, three-dimensional laser rangefinders are currently a de facto standard equipment for self-driving cars.

We investigate robot perception using three-dimensional laser range data in this thesis, since we also want to determine the categories of objects visible in the vicinity of an autonomous system. The *classification* of the sensor input allows the system to incorporate knowledge about the object classes into its decision making process. Especially the potentially dynamic objects, e.g., cars, pedestrians, and cyclists, are of fundamental importance in the context of self-driving cars, since each class shows very different kinematics. As we cannot easily describe heuristic rules to assign classes to objects by hand, we will extensively use machine learning to deduce these rules automatically from the data itself. Machine learning becomes increasingly important in many application areas, which were dominated by hand-crafted algorithms, such as computer vision, information retrieval, but also robotics, and replace many of these established methods by largely improved algorithms. Especially, the field of robotics offers many fundamental challenges, where machine learning could help to develop better methods to enable more intelligent behavior of robots. Many of these challenges can only be tackled and effectively learned by carefully designed machine learning models that capture the essence of the problems by learning on massive datasets. Note that machine learning does not solve these challenges by simply applying out-of-the-box learning algo-

rithms to a given problem, but needs engineering to specify a suitable model and to induce constraints on the problem. The No Free Lunch theorem [Wolpert, 1996] even proves that there is no single method that optimally solves every given supervised machine learning problem.

The goal of this thesis is the development of effective and efficient methods for classification of three-dimensional laser range data. We have to consider mainly two ingredients for this endeavor: the features derived from the sensor data and the classification model used to distinguish object classes represented by these features. Both aspects will be covered thoroughly in this thesis. In Chapter 3, we investigate suitable features. Based on these features, we propose novel models for classifying laser range data in Chapter 4 and 5.

1.1 Contributions of the Thesis

The thesis investigates the complete processing pipeline of classification and proposes novel methods for the classification of three-dimensional laser range data. For the classification of three-dimensional laser range data, we must tackle two fundamental problems: First, we have to process a massive amount of data, since a point cloud consists of up to 140.000 unorganized three-dimensional points. Second, we encounter a distance dependent sparsity of the point clouds representing objects, where we can observe very dense point clouds near to the sensor and sparse point clouds at far distances. Considering both challenges, we aim at algorithms that are efficient in respect to a huge amount of data and also robust regarding very different sparsities of the three-dimensional laser returns. The contributions of the thesis are as follows:

- In Chapter 3, “Histogram Descriptors for Laser-based Classification,” we experimentally evaluate histogram descriptors in a classification scenario. We show the influence of different design decisions using three different representative datasets and investigate the performance of two established classification approaches. Especially, the selection of an appropriate reference frame turned out to be essential for an effective classification. The presented results are the first thorough and systematic investigation of descriptors for laser-based classification in urban environments.
- Chapter 4, “Efficient hash-based Classification,” presents a novel algorithm combining similarity-preserving hashing and a local classification approach that improves the label consistency of the point-wise classification results significantly. These improvements are achieved with little computational overhead compared to the competing local classification approaches and enables therefore efficient classification of three-dimensional laser range data.

- Chapter 5, “Segment-based Classification,” presents a complete approach for segment-based classification of three-dimensional laser range data. We propose an efficient hierarchical segmentation approach to improve the extraction of consistent segments representing single objects. We then develop a new classification approach that combines multiple feature representations. For filtering of duplicate and irrelevant segments, we also develop an efficient non-maximum suppression exploiting the aforementioned segment hierarchies. We finally investigate methods to improve the efficiency of the proposed classification pipeline.

1.2 Structure of the Thesis

In the next part, Chapter 2, “Fundamentals,” we introduce fundamental concepts and terminology needed for a self-contained presentation of the thesis. We will first cover basics concerning three-dimensional laser range data, the acquisition and basic processing of this type of data. Then, we will introduce basic terminology of machine learning and the softmax regression in more detail, since this linear classification model will be extended in the following chapters.

In the subsequent chapters, we cover our contributions in more detail and present experimental results, which show exemplarily the claimed improvements over the state-of-the-art on real world datasets.

In Chapter 3, “Histogram Descriptors for Laser-based Classification,” we investigate suitable feature representations using two established classification models, the softmax regression and a more complex graph-based classification approach. The insights of this performance evaluation build the foundation for the following chapters, which concentrate on the improvement of the simple, but very efficient softmax regression.

In Chapter 4, “Efficient hash-based Classification,” we will improve the softmax regression to obtain a more consistent point-wise labeling.

The following Chapter 5, “Segment-based Classification,” is then concerned with the classification of segments of objects relevant for autonomous driving.

In the end of each chapter, we will point to future directions of research on top of the presented approaches.

Chapter 6, “Conclusions,” finally concludes the thesis by summarizing the main insights and by giving prospects of future work and open research questions.

Chapter 2

Fundamentals

This chapter covers basic concepts and formally introduces the terminology used in the rest of the thesis. Additional concepts or methods required only in a specific context will be introduced in the corresponding chapters.

In the first part of the chapter, Section 2.1, “Three-dimensional Point Cloud Processing,” we thoroughly discuss the processing of three-dimensional point clouds. In course of this part, we briefly introduce different data acquisition methods, data structures for fast neighbor search, and introduce the normal estimation using neighboring points. The remaining chapter introduces in Section 2.2, “Classification,” concepts and terminology of supervised classification. We first derive a basic discriminative classification model for multiple classes — the softmax regression. Afterwards, we discuss another model placed at the opposite end of the spectrum of classification approaches compared to the softmax regression – the k nearest neighbor classifier. While discussing these models, we will introduce basic terms encountered all over the thesis and lastly cover aspects of model complexity and model assessment.

2.1 Three-dimensional Point Cloud Processing

In robotic applications aiming at deploying autonomous systems in populated areas, we need to avoid collisions with people and other obstacles. Consequently, we have to ensure a safety distance of the robot to the surrounding objects at all times. Range data is the prevalent sensory input used for collision avoidance.

Laser rangefinders are favored over other ranging devices, as they provide precise range measurements at high update rates. A laser rangefinder or so-called *LiDAR* (Light Detection

(a) (b)

Figure 2.1: *The left image (a) shows a sketch of a common two-dimensional laser rangefinder with rotating mirror (yellow). The encoder disk (blue) is used to measure the rotation angle of the mirror. In indoor environments, two-dimensional laser rangefinders are usually mounted co-planar to the ground as depicted in the right image (b).*

time-of-flight

And Ranging) device measures the distance to an object by emitting and receiving laser beams. The range or distance is estimated using the *time-of-flight*, i.e., the time it takes to receive a previously emitted laser beam again.

Two-dimensional laser rangefinders, depicted in Figure 2.1a, commonly use a mirror to refract the laser beam and record two values at time t, the range r_t and the rotational angle or azimuth ϕ_t of the mirror. If we take measurement pairs $\{(r_0, \phi_0), \ldots, (r_M, \phi_M)\}$ of a mirror revelation and calculate their corresponding Cartesian points $(r_i \sin\phi_i, r_i \cos\phi_i)$, we get a range profile of a slice of the environment. In indoor environments, a robot moves in the plane and therefore it is usually sufficient to mount a two-dimensional laser rangefinder co-planar to the ground, as shown in Figure 2.1b. As long as there are no overhanging structures or staircases, such sensor setup can be used for a safe and collision-free navigation, even in complex and highly dynamic environments, such as museums [Burgard et al., 1999, Thrun et al., 1999] or home improvement shops [Gross et al., 2009].

In non-flat terrain, the aforementioned co-planar mounting is obviously insufficient. In such situations, three-dimensional laser rangefinders, which additionally vary a third degree of freedom to measure ranges, can be used to generate an adequate and complete three-dimensional representation of the environment. These measurements let the robot sense the complete shape of objects and the appearance of the terrain. As before, we can derive from the range r_t, inclination θ_t, and azimuth ϕ_t of such a rotating laser sensor the Cartesian coordinates $(r_t \sin\theta_t \cos\phi, r_t \sin\theta_t \sin\phi_t, r_t \cos\theta_t)$. We refer to $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$ with three-dimensional points $\boldsymbol{p}_i \in \mathbb{R}^3$ as *point cloud*. In the following, we assume no particular ordering of points or a specific data acquisition and use *scan* instead of point cloud to refer to the generated laser range data.

point cloud
scan

Before we introduce the acquisition of laser range scans in the next section, we first discuss the advantages and disadvantages of laser range data compared to images.

In images, colors and appearance of a scene may drastically vary, if they are captured under different illumination. Therefore most image descriptors rely on some contrast normalization or invariant properties, such as gradient orientation [Lowe, 2004], or relative intensities [Calonder et al., 2012]. Extracting segments, which correspond to objects, from an image is challenging using only image data and usually accomplished with complex graph-based methods [Forsyth and Ponce, 2012]. Laser range measurements contrariwise are not affected by different lighting, enabling for example the usage at night. Furthermore, we can usually extract coherent segments from the point cloud with rather simple methods.

However, laser rangefinders have also some notable disadvantages compared to color images. We only get the distance to the surface and the reflectance of the material, but not any other multi-spectral information like in images. Laser beams quite often get absorbed by black surfaces or refracted by glass, and therefore 'holes' without any range measurement occur frequently. Another shortcoming is the representation as three-dimensional point cloud, since we have no implicit neighboring information like in images. Thus, the runtime of certain operations, such as neighbor queries, is relatively high compared to the same operation in images.

In the following sections, we will discuss different fundamental methods for processing of laser range data. First, we discuss the acquisition of laser range data using common sensor setups. Then we briefly introduce efficient data structures for acceleration of neighbor searches and finally, the estimation of normals using eigenvectors is discussed.

2.1.1 Data Acquisition

Over the years, different setups for the generation of three-dimensional laser point clouds were developed. Earlier setups used primarily two-dimensional laser rangefinder and varied a third dimension. Until recently, generating a point cloud using such setup took more than a second. The recent development of ultra-fast three-dimensional laser rangefinders producing detailed points clouds in a fraction of a second stimulated the research of algorithms for the interpretation of this kind of data.

Three-dimensional laser range data is mainly generated using one of the following three sensor setups: (1) a sweeping planar laser range sensor, (2) a tilting planar laser range sensor, or (3) a rotating sensor.

In the first case, a two-dimensional sensor is fixated on the robot and a three-dimensional point cloud of the environment is generated as the robot moves forward (see Figure 2.2a). The laser rangefinder is swept over the surrounding structures, which makes is necessary to

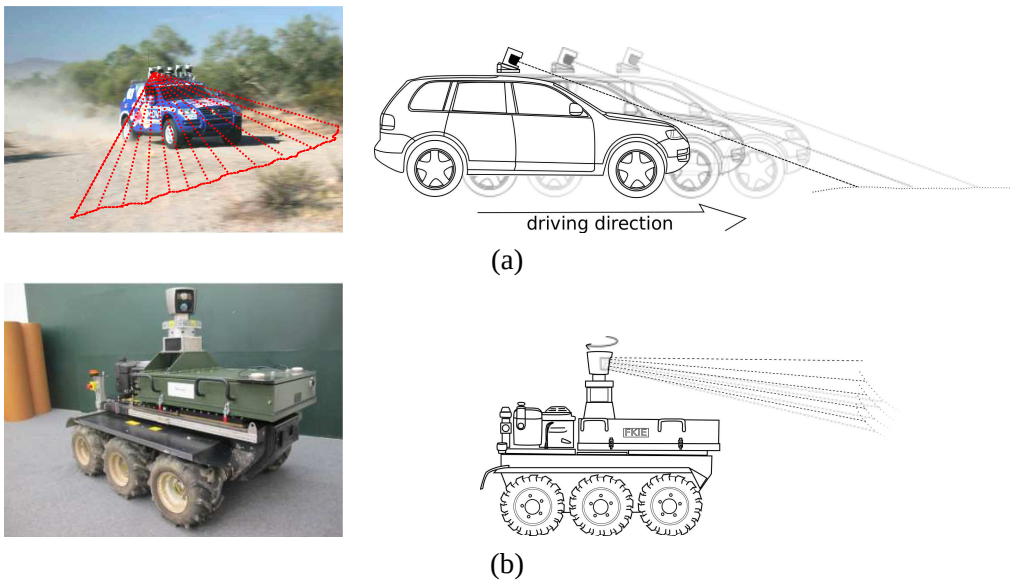


Figure 2.2: *Common laser scanner setups: (a) A two-dimensional laser scanner is mounted on a car and the road ahead is scanned as the car moves forward. In this specific example[1] the sensors are additionally tilted to increase the covered area in front of the car. Figure (b) shows a rotating laser range sensor, the Velodyne HDL-64 E, mounted on a Qinetiq Longcross robot[2]. The sensor covers the full* 360° *surrounding of the robot in contrast to the former setup.*

move the robot and offers only three-dimensional data for a restricted area in front or sideways of the robot. In navigation applications, this sensor setup is mainly used to get a precise point cloud in front of the robot and to decide where drivable ground [Kümmerle et al., 2013, Thrun et al., 2006] is located. To enlarge the covered area in front of the robot, a pan/tilt unit (PTU) can be attached to the sensor and with this setup, the robot is able to generate laser range scans without moving [Marder-Eppstein et al., 2010].

The second setup uses also a PTU to sweep the sensor over the environment, but here also the direction of the sensor is adjusted [Steder et al., 2011a]. A static robot is thus able to generate a complete 360° view of the environment by rotating the sensor in different directions. However, generating a complete point cloud of the vicinity usually takes several seconds. Due to the tilting of the sensor, the sensor must be decelerated and accelerated repeatedly causing high mechanical forces.

Lastly, the third setup uses a far more stable full rotation of the sensor, where the sensor just keeps spinning and decelerating the sensor is unnecessary (Figure 2.2b). Rotating sensors

[1] The photo was taken from the website of the Stanford Racing Team, which won the DARPA Grand Challenge: http://cs.stanford.edu/group/roadrunner/old/index.html. [Accessed: 10 Oct. 2013]

[2] Longcross photo by courtesy of Unmanned Systems Group, Fraunhofer FKIE.

are currently the preferred setup to generate three-dimensional laser range data, since a complete 360° three-dimensional laser range scan can be generated in a fraction of a second. A common setup is to mount a two-dimensional laser range sensor vertically, such that the rotation of the sensor generates vertical slices of the environments. Combining these slices finally results in a complete three-dimensional point cloud with a wide field of view.

We are mainly interested in the Velodyne HDL-64E S2 [Velodyne Lidar Inc., 2010], which was lately employed in many outdoor robotics applications, e.g., navigation [Hoeller et al., 2010], tracking [Schöler et al., 2011], object recognition [Teichman and Thrun, 2012], and simultaneous localization and mapping [Moosmann and Stiller, 2010]. The Velodyne laser range sensor is equipped with 64 laser diodes organized in two groups of 32 diodes, which are emitted simultaneously, while the sensor is rotating around its main axis (Figure 2.2b). The rotation speed of the sensor can be adjusted from 5 to 15 Hz, but this does not influence the frequency of the laser beam emissions. Thus, the sensor produces always approximately 1.3 million laser range measurements per second, but the number of laser points in every revelation varies according to the rotational speed. Nevertheless, we speak in the following of a *complete scan*, if one revelation of the sensor is completed. Developed for autonomous driving, this sensor generates only a narrow vertical field of view of 26.8° ranging from +2° to −24.8° inclination. Mounted at sufficient height on the car roof, the sensors field of view covers all relevant parts of the street. However, large objects, such as houses or trees, are often only represented in the point cloud by their lower parts due to the nearly horizontal upper boundary of the field of view.

complete scan

Common for all mentioned setups is the generation of millions of laser range points showing a distance dependent resolution. At small ranges up to 5 meters, a person is covered densely by range measurements, but at distances larger than 15 meters the same person is only sampled sparsely by the laser rangefinder. This challenge is rarely encountered in indoor environments, since there the workspace is less than 10 meters. With this large range of distances to objects, we have to ensure some kind of sampling invariance and develop methods, which are capable to work with both very dense and very sparse point clouds.

2.1.2 Neighbor Search

A fundamental operation needed by many approaches using point clouds is the search for neighboring points of a point $\boldsymbol{p}$. We denote the set of *radius neighbors* of a point $\boldsymbol{p} \in \mathcal{P}$ inside a radius δ by $\mathcal{N}_{\boldsymbol{p}}^{\delta} = \{\boldsymbol{q} \in \mathcal{P} \,|\, \|\boldsymbol{p} - \boldsymbol{q}\| \leq \delta\}$. Let $\mathcal{N}_p^{\leq} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N\}$ be the partially ordered set of points, where $\left\|\boldsymbol{q}_i - \boldsymbol{p}\right\| \leq \left\|\boldsymbol{q}_{i+1} - \boldsymbol{p}\right\|$. The set of *k-nearest neighbors* $\mathcal{N}_p^k$ is given by the first k elements of $\mathcal{N}_p^{\leq}$. Note that the k nearest neighbors are not unique, since there can be multiple neighbors with the same distance to the query point.

radius neighbors

k-nearest neighbors

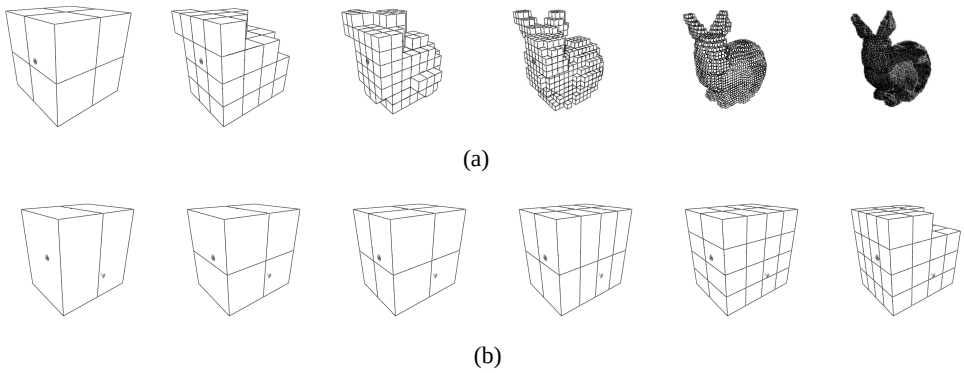


Figure 2.3: *First iterations of the subdivisions of octree (a) and k-d tree (b) build for the Standford bunny point dataset. Every picture shows non-empty nodes at a certain level of the tree. The subdivision of the space progresses faster in case of the octree, since every node in the octree can have* 8 *children. Subdivision in the k-d-tree is performed in the dimension with largest extend and the mean is used to split the point set.*

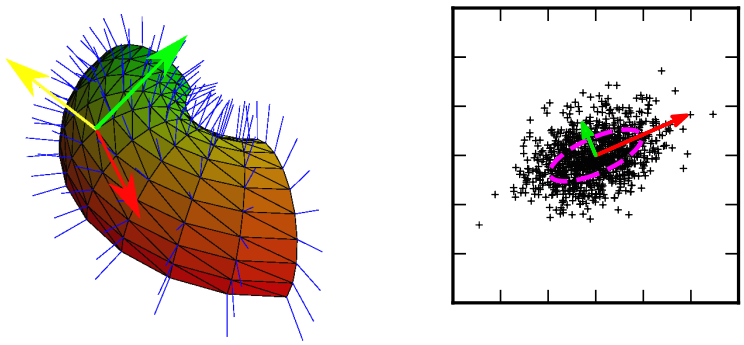
Both types of neighbor searches, radius and k-nearest neighbor search, can be performed efficiently using space partitioning trees [Pharr and Humphreys, 2010], i.e., spatial data structures that avoid linearly searching all points in $O(N)$. Two spatial subdivision data structures are commonly used to accelerate the neighbor search, the octree [Meagher, 1982] and the k-d tree [Friedman and Bentley, 1977]. While k-d trees can be used to accelerate search for neighbors in arbitrary dimensions, an octree is restricted to three-dimensional data sets.

octree The *octree* construction starts with an axis-aligned bounding box, which encloses all points of the point cloud. The bounding box is recursively splitted into 8 equally-sized octants, where we split the point cloud into subsets according to the boundaries of these octants. The subdivision is repeated until the size of the octants reaches a lower bound or a minimal number of points is reached.

k-d tree The *k-d tree* construction also starts with an axis-aligned bounding box enclosing the point cloud. However, the cuboid is subdivided along a single dimension such that almost equally sized partitions are formed. Then every subset itself is subdivided again at the dimension with maximal extent until a certain number of points are left. Hence the resulting tree is binary, where every node contains a threshold and a dimension parameter deciding which path to follow to reach a leaf containing points.

Figure 2.3 visualizes some stages of the construction of an octree and a k-d tree and shows the non-empty nodes at every level of the data structures. The figure depicts a faster progression of the subdivision for the octree due to a higher number of possible children in the resulting tree.

Searching for radius neighbors in both trees is accomplished by determining all nodes in the tree that overlap with a ball of radius δ and midpoint $\boldsymbol{p}$. Inside each node, the list of points



(a) (b)

Figure 2.4: *In figure (a) a mesh of a torus is depicted and corresponding normals (blue). Also shown are tangential vectors (red and green) of a surface point and the corresponding normal (yellow). In (b) a two-dimensional point set and the eigenvectors $\boldsymbol{v}_0$(green) and $\boldsymbol{v}_1$(red) are shown. The eigenvectors are scaled according to the corresponding eigenvalue, λ_0 and λ_1. The iso-contour of the covariance is shown as purple dashed ellipse.*

is then finally examined for neighbors inside the desired radius. K-nearest neighbors can be searched similarly, but here the maximal distance is dynamically reduced to the distance of the k-th neighbor. For small radii, we can achieve significant accelerations, because we only have to examine a very small set of points compared to the overall number of points.

In summary, both data structures are heavily used to accelerate point cloud neighbor search and recent results of Elseberg et al. [2012] suggest that the best strategy is highly data-dependent. We opt for using an octree for radius neighbor search in three-dimensional point clouds and we use a k-d tree [Arya et al., 1998] for higher dimensional data. For our datasets of urban environments, octrees showed faster retrieval times than the implementation of the exact search of Arya et al. [1998] using a k-d tree.

2.1.3 Normal Estimation

In many approaches, the (surface) normal is used as additional information besides the location of the point. The normal can be defined by the cross product $\boldsymbol{s}\times\boldsymbol{t}$ of two nonparallel tangent vectors, $\boldsymbol{s}$ and $\boldsymbol{t}$, at a particular point on a surface (cf. Figure 2.4). The orientation of the normal is usually chosen such that the normal points outside of the object.

However, we only observe point-wise range measurements as reflection of surfaces. We usually cannot easily generate a representation such as a triangular mesh from these three-dimensional points, which allows us to calculate directly the normal orientation using two sides of a triangle [Pharr and Humphreys, 2010]. Thus, we are only able to estimate the surface normal at a point $\boldsymbol{p}$ using the neighboring points $\mathcal{N}_{\boldsymbol{p}}^{\delta}$. Principle component analysis

(PCA) of the covariance matrix $\mathbf{C}$ is a common method for estimating the normal orientation of a point $\boldsymbol{p}$.

covariance matrix

The *covariance matrix* $\mathbf{C} \in \mathbb{R}^{3\times3}$ of a neighborhood $\mathcal{N}_{\boldsymbol{p}}^{\delta}$ of point $\boldsymbol{p} \in \mathbb{R}^3$ is defined by

$$\mathbf{C} = \frac{1}{\left|\mathcal{N}_p^{\delta}\right|} \sum_{\boldsymbol{q}\in\mathcal{N}_p^{\delta}} (\boldsymbol{q} - \bar{\boldsymbol{q}})^{\mathrm{T}} (\boldsymbol{q} - \bar{\boldsymbol{q}}) \tag{2.1}$$

$$= \frac{1}{\left|\mathcal{N}_p^{\delta}\right|} \sum_{\boldsymbol{q}\in\mathcal{N}_p^{\delta}} \boldsymbol{q}^{\mathrm{T}}\boldsymbol{q} - \bar{\boldsymbol{q}}^{\mathrm{T}}\bar{\boldsymbol{q}} \tag{2.2}$$

with $\bar{\boldsymbol{q}} = \left|\mathcal{N}_{\boldsymbol{p}}^{\delta}\right|^{-1} \sum_{\boldsymbol{q}\in\mathcal{N}_{\boldsymbol{p}}^{\delta}} \boldsymbol{q}$, i.e., the mean vector of the neighboring points. The covariance contains in $\mathbf{C}_{i,j}$ the covariances between dimension i and j, and thus represents the change of the point distribution in these dimensions. In addition, $\mathbf{C}$ is symmetric and positive semi-definite by construction. Therefore, all eigenvalues $\lambda_2 \geq \lambda_1 \geq \lambda_0 \geq 0$ are positive real valued and the corresponding eigenvectors $\boldsymbol{v}_2, \boldsymbol{v}_1$, and $\boldsymbol{v}_0$ are orthogonal to each other.

Intuitively, the eigenvalue λ_i expresses the change of the distribution in the direction of the eigenvector $\boldsymbol{v}_i$. Thus, if we think of a point cloud of a surface patch, as shown in Figure 2.4, we have the largest changes in direction of the surface patch, i.e., tangential to the surface. The smallest change is orthogonal to these tangential directions and therefore a good estimate of the normal direction.

However, the eigenvector orientation is ambiguous and therefore the smallest eigenvectors $\boldsymbol{v}_0$ for neighboring points can be orientated contrary. Hence, we might have to flip the orientation of the normal vectors, $\boldsymbol{n}_i = -\boldsymbol{v}_0$, such that all normals $\boldsymbol{n}_i$ point towards the known sensor location for a consistent normal orientation.

Depending on the environment and application, different values of neighbor radius δ are appropriate. In indoor environments or for retrieval tasks, a small radius is appropriate, since we are usually interested in very fine details and operate in small scales. The application area of our approaches is the outdoor environment, where we encounter large surfaces and objects and objects are generally scanned at larger distances compared to indoor applications. Therefore, we usually choose a large radius to allow the estimation of a normal direction for sparsely sampled surfaces.

2.2 Classification

We are interested in assigning each laser range point a pre-determined class or label, which corresponds to a specific category, such as *pedestrian*, *car*, *building*, *ground*, etc. Since we cannot easily write down a heuristic rule — such as using some numerical values of a

point and determining from this a label — we employ techniques from machine learning to extract such rules using labeled data. For this purpose, we specify a model and then 'fit' the model parameters to the dataset with inputs and given targets values until the fitted model explains the given data. This learning paradigm is called *supervised learning* and will be discussed in more detail in the following section.

supervised learning

In supervised learning, we are interested in a function or probabilistic model, which relates an input $\boldsymbol{x} \in \mathbb{R}^D$ to a target value y. We supervise the learning algorithm by an appropriately labeled *training set*, $\mathcal{X} = \{(\boldsymbol{x}_0, y_0), \ldots, (\boldsymbol{x}_N, y_N)\}$, representing the task we intend to solve. This chapter discusses particularly supervised classification, i.e., the output *class* or label $y \in \mathcal{Y} = \{1, \ldots, K\}$ is discrete.

training set

In particular, we want a probabilistic representation $P(y|\boldsymbol{x})$, where we get the predicted class y and additionally an estimate of the uncertainty of this prediction. As we get the distribution $P(y|\boldsymbol{x})$ *after* seeing the data $\boldsymbol{x}$, $P(y|\boldsymbol{x})$ is also called the *posterior distribution*.

posterior distribution

Using Bayes' rule, Equation A.4, we can derive the following equivalent representation:

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} \tag{2.3}$$

$$= \frac{P(\boldsymbol{x}|y)P(y)}{\sum_y P(y, \boldsymbol{x})} \qquad \text{using (A.1)} \tag{2.4}$$

$$= \frac{P(\boldsymbol{x}|y)P(y)}{\sum_y P(\boldsymbol{x}|y)P(y)} \qquad \text{using (A.3)} \tag{2.5}$$

The *prior distribution* $P(y)$ encodes our belief about the label distribution before seeing any input data. In addition, we refer to $P(\boldsymbol{x}|y)$ as *likelihood*, since it encodes how likely it is to observe data $\boldsymbol{x}$ given a certain label y.

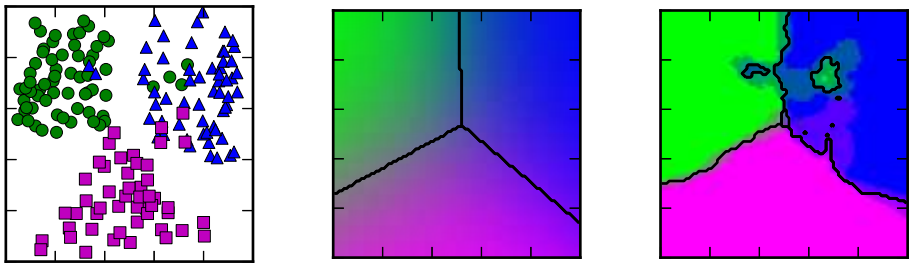
prior distribution
likelihood

Thus, we can decide on modeling either $P(\boldsymbol{x}|y)$ and $P(y)$, or $P(y|\boldsymbol{x})$ directly. In case of modeling $P(\boldsymbol{x}|y)$ and $P(y)$, we refer to this paradigm as a *generative model* and we estimate $P(y|\boldsymbol{x})$ using Equation 2.5. We can actually generate new data by sampling from $P(\boldsymbol{x}|y)$. If we model $P(y|\boldsymbol{x})$ directly, we call this a *discriminative model* and can usually save many parameters. In the following, we prefer a discriminative approach, since it is usually harder to specify a model of the data $P(\boldsymbol{x}|y)$ than specifying how the data affects the label $P(y|\boldsymbol{x})$.

generative and discriminative classification

Using the discriminative approach, we now have to decide on a suitable model for $P(y|\boldsymbol{x})$. Over the recent years, a multitude of different models were proposed [Barber, 2012, Bishop, 2006, Prince, 2012], which have very different properties and also model complexities. In this context, we use the term *model capacity* to refer to the kind of dependencies, which can be modeled and consequently learned from data. If the model capacity is higher, we are usually able to model more complex relationships between labels and data. Nevertheless,

model capacity



(a) (b) (c)

Figure 2.5: *Classification example. Subfigure (a) depicts a training set with* 3 *classes with clearly visible clusters, but some data points are outside of these clusters. Subfigure (b) and (c) graphically show the probability* $P(y|\boldsymbol{x})$ *for every possible point of two classification models learned with this data. Here the intensity of every label color corresponds to the probability – the brighter the color, the more certain is the classification model that the feature vectors belongs to the corresponding class. The classifier in (b) shows linear decision boundaries, whereas (b) shows more complex non-linear decision boundaries.*

increasing the model capacity is a double-edged sword as we will see later, when we will discuss overfitting in Section 2.2.3.

Suppose we get the simple two-dimensional training set given in Figure 2.5 containing three classes indicated by different colors and shapes of the points. Each point corresponds to an input vector $\boldsymbol{x}_i$ and the corresponding label y_i is indicated by its color. The input is also called *feature vector*, since the raw data is preprocessed commonly to generate an intermediate representation with features or characteristics relevant for the task. In the following, we will use *feature space* to refer to the vector space $\mathbb{R}^D$ of all feature vectors.

feature vector

feature space

Typically we do not have precise knowledge about the generating process producing the data and consequently any information about possible feature values. Hence, we have to decide on an appropriate model for modeling the dependencies between a feature vector $\boldsymbol{x}$ and the corresponding label y. These model assumptions induce a certain label assignment $\hat{y}$ for an unseen feature vector $\hat{\boldsymbol{x}}$. The set of feature vectors $\hat{\mathcal{X}} = \{\hat{\boldsymbol{x}}_0, \ldots, \hat{\boldsymbol{x}}_{\hat{N}}\}$ for which we are interested in predicting the label $\hat{y}_j$, is called *test set*.

test set

Using different model assumption, we might get the depicted assignments in Figure 2.5 (b) and (c). Here, colors indicate the class assignments, where the purity of a color corresponds to the certainty of the assignment, i.e., the brighter and purer the color is, the more certain or larger is $P(\hat{y}|\hat{\boldsymbol{x}})$ for this class. A *decision region*[3] $\mathcal{D}_k = \{\boldsymbol{x}|P(y = k|\boldsymbol{x}) \geq P(y = l|\boldsymbol{x})\}$ of class k is now the region of feature vectors $\boldsymbol{x}$, where $P(y = k|\boldsymbol{x})$ is maximal. The *decision*

decision region

decision boundary

[3] The definition is suited, if we aim at minimizing the misclassification rate. See Bishop [2006], chapter 1.5, for a more detailed discussion of decision theory.

boundary is defined as $\partial\mathcal{D} = \cup_{k,l\in\mathcal{Y},k\neq l}\mathcal{D}_k \cap \mathcal{D}_l$ and therefore separates all classes depicted by black strokes in Figure 2.5.

In (b) the model assumes a linear dependency between feature vector $\hat{\boldsymbol{x}}$ and predicted label $\hat{y}$, and hence the decision boundaries are straight lines. The model in (c) shows very different decision boundaries and models a non-linear dependency between feature vectors and label. Depending on the task and expert knowledge, either the first model or the second model is closer to the truth. The linear model treats some feature vectors of the training set as *outliers*, i.e., data that was generated by an unknown random effect, but not by the generating process itself. The more complex decision regions of subfigure (c), adjusted the model parameters to include some of these points. Thus, we can see inside the blue and green region small decision regions, where the model predicts a different class label.

outliers

Until now, we just described that we have to decide on different paradigms to model our supervised learning task, but we have not explained how to actually learn a model given the labeled training set. Every model is parameterized by a set of *model parameters* $\boldsymbol{\theta}$, which can be adjusted to change the output of the probabilistic model. As stated earlier, we aim at finding parameters, which best fit to the given training data $\mathcal{X}$ and are therefore interested in the probability distribution $P(\boldsymbol{\theta}|\mathcal{X})$.

model parameters

As before we can apply Bayes' rule to derive a more accessible and equivalent expression.

$$P(\boldsymbol{\theta}|\mathcal{X}) = \frac{P(\mathcal{X}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathcal{X})} \tag{2.6}$$

$$= \frac{P(\mathcal{X}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int P(\mathcal{X}|\boldsymbol{\theta})P(\boldsymbol{\theta})\, d\theta} \tag{2.7}$$

We can introduce prior knowledge using $P(\boldsymbol{\theta})$ and determine the likelihood by $P(\mathcal{X}|\boldsymbol{\theta})$. Assuming that the data is independent and identically distributed (i.i.d.)[4], we can further simplify Equation 2.7 and substitute the training data $\mathcal{X}$ by its elements $\boldsymbol{x}_i$ and y_i:

$$P(\mathcal{X}|\boldsymbol{\theta})P(\boldsymbol{\theta}) = \prod_i P(\boldsymbol{x}_i, y_i|\boldsymbol{\theta})P(\boldsymbol{\theta}) \qquad \text{using (A.7)} \tag{2.8}$$

$$= \prod_i P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})P(\boldsymbol{x}_i|\boldsymbol{\theta})P(\boldsymbol{\theta}) \qquad \text{using (A.3)} \tag{2.9}$$

$$= \prod_i P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})P(\boldsymbol{x}_i)P(\boldsymbol{\theta}) \tag{2.10}$$

$$= \prod_i P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})P(\boldsymbol{\theta}) \tag{2.11}$$

[4] All elements of the training set are independently drawn from the same generating distribution, i.e., we did not select any training sample accounting the selection of another training example.

In Equation 2.10 we exploit the independence of the feature vectors $\boldsymbol{x}_i$ from the parameters $\boldsymbol{\theta}$, i.e., $P(\boldsymbol{x}_i|\boldsymbol{\theta}) = P(\boldsymbol{x}_i)$. Finally, in Equation 2.11 we can cancel $P(\boldsymbol{x}_i)$ with the denominator from Equation 2.7.

Bayesian approach

In a full *Bayesian approach*, we would now have to estimate the likelihood of all possible model parameters $\boldsymbol{\theta}$ and use these values to infer the posterior $P(\hat{y}|\mathcal{X}, \hat{\boldsymbol{x}})$ using marginalization:

$$P(\hat{y}|\hat{\boldsymbol{x}}, \mathcal{X}) = \int P(\hat{y}|\hat{\boldsymbol{x}}, \boldsymbol{\theta}) P(\boldsymbol{\theta}|\mathcal{X}) \, d\boldsymbol{\theta} \tag{2.12}$$

However, determining the distribution $P(\boldsymbol{\theta}|\mathcal{X})$ over the parameters $\boldsymbol{\theta}$ is usually computationally intractable due to the integral in the denominator and can only be computed with specific distributions in closed form [Prince, 2012]. Thus, we usually work only with the best parameters $\boldsymbol{\theta}^\star$ and simply use $P(\hat{y}|\hat{\boldsymbol{x}}, \boldsymbol{\theta}^\star)$ instead of Equation 2.12 for inference. Estimating the best parameters is achieved by maximizing Equation 2.11 and yields

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \prod_i P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) P(\boldsymbol{\theta}). \tag{2.13}$$

maximum a posteriori

maximum likelihood

If we incorporate prior knowledge about the parameters, this kind of parameter estimation is called *maximum a posteriori* (MAP) estimation. A suitable prior regularizes the solution and can reduce the effects of lack in data evidence. A quite common approach is to use a uniform or flat prior, where all model parameters $\boldsymbol{\theta}$ are equally likely. This approach is called *maximum likelihood* estimation.

linear separable

Next, we will introduce two basic models for multi-class classification with very different capacities. The first model has only very few parameters and is restricted to the class of linearly separable classes. A feature space is *linear separable*, if we can choose arbitrary feature vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$ belonging to the same class y and then all other vectors

$$\boldsymbol{x}_k = \lambda \boldsymbol{x}_i + (1 - \lambda)\boldsymbol{x}_j, 0 \leq \lambda \leq 1 \tag{2.14}$$

on a straight line are also in the same class y.

Since some classification problems show classes that are not linear separable, we have to enrich our model with some flexibility. The second model discussed in this chapter is more flexible, but still easy to describe. However, we will later discuss the problems with too much flexibility, if we only have limited amount of data available to learn the model parameters.

The classification models discussed in this chapter are at opposite ends of the spectrum of classification models and there are many other possible choices [Barber, 2012, Bishop, 2006, Prince, 2012] in between. The first model, the softmax regression, is discussed more deeply, since it will be extensively used in the rest of the thesis and it is of particular interest

in our application as it enables very fast inference at prediction time in contrast to other more complex models. The second model, the *k*-nearest neighbor classifier, was chosen because of its simplicity and will be later used in context of point-wise classification for comparison purposes.

2.2.1 Softmax Regression

Assuming a linear relationship between the feature vector $\boldsymbol{x}$ and the class y, we can model $P(y|\boldsymbol{x})$ as follows:

$$P(y = k|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x})}{\sum_i \exp(\boldsymbol{\theta_i}^{\mathrm{T}} \cdot \boldsymbol{x})} \tag{2.15}$$

This model is usually called multi-class logistic regression or *softmax regression* [Bishop, 2006, Prince, 2012]. The term

softmax regression

$$s_i = \frac{\exp\left(\boldsymbol{a}^{(i)}\right)}{\sum_j \exp\left(\boldsymbol{a}^{(j)}\right)} \tag{2.16}$$

of a vector $\boldsymbol{a} \in \mathbb{R}^D$ corresponds to a smooth approximation of the maximum, which returns the largest value over all entries of $\boldsymbol{a}$ for the maximum of $\boldsymbol{a}$, and is therefore called *softmax*. The results of the softmax satisfy $0 \leq s_i \leq 1$ and sum up to 1. $P(y|\boldsymbol{x})$ is therefore a valid probability distribution.

softmax

Let the model be specified by model parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k) \in \mathbb{R}^{K \cdot D \times 1}$. As introduced earlier, we are interested in determining the parameters $\boldsymbol{\theta}$, which best explains the training set $\mathcal{X} = \{(\boldsymbol{x}_0, y_0), \ldots, (\boldsymbol{x}_N, y_N)\}$. Introducing the model parameters, we aim at maximizing the likelihood $L(\boldsymbol{\theta}) = P(\boldsymbol{\theta}|\mathcal{X})$. We prefer a MAP learning approach and choose a normal-distributed prior for $\boldsymbol{\theta} \sim \mathcal{N}(0, \Sigma)$ with circular covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{K \cdot D \times K \cdot D}$, i.e., a diagonal matrix with entries λ^{-1}. By adjusting λ we can regularize $\boldsymbol{\theta}$ such that the length $||\boldsymbol{\theta}||^2 = \boldsymbol{\theta}^T \boldsymbol{\theta}$ is constrained. Thus, this type of model is also called L2-regularized softmax regression. Assuming again i.i.d. training examples $(y_i, \boldsymbol{x}_i)$, we maximize the following objective:

$$\arg\max_{\theta} L(\boldsymbol{\theta}) = \arg\max_{\theta} \prod_i P(y = y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}) \tag{2.17}$$

$$= \arg\max_{\theta} \prod_i \frac{\exp(\boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i)}{\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)} \cdot \frac{1}{\sqrt{2\pi^{K \cdot D}|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}\right) \tag{2.18}$$

$$= \arg\max_{\theta} \prod_i \frac{\exp(\boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i)}{\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)} \cdot \exp\left(-\frac{1}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}\right) \tag{2.19}$$

$$= \arg\max_{\theta} \prod_i \frac{\exp(\boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i)}{\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)} \cdot \exp\left(-\frac{\lambda}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}\right) \tag{2.20}$$

We exploited the maximization in the last two lines and dropped the constant factors of the Gaussian. Unfortunately, there is no closed-form solution of Equation 2.20 and thus we have to optimize iteratively. Nevertheless, one can show that the resulting objective is concave [Barber, 2012] and therefore shows only a global maximum. It is numerically more stable to use the log of the likelihood, where some of the terms reduce to simpler ones:

$$\ln L(\boldsymbol{\theta}) = \ln\left[\prod_i \frac{\exp(\boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i)}{\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)} \cdot \exp\left(-\frac{\lambda}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}\right)\right] \tag{2.21}$$

$$= \sum_i \ln\left[\frac{\exp(\boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i)}{\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)}\right] + \ln\left[\exp\left(-\frac{1}{2}\lambda\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}\right)\right] \tag{2.22}$$

$$= \sum_i \ln\left[\exp(\boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i)\right] - \ln\left[\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)\right] - \frac{\lambda}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} \tag{2.23}$$

$$= \sum_i \boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i - \ln\left[\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)\right] - \frac{\lambda}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} \tag{2.24}$$

$$= \sum_i \boldsymbol{\theta}_{y_i}^{\mathrm{T}} \cdot \boldsymbol{x}_i - \ln\left[\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)\right] - \frac{\lambda}{2}\sum_k \boldsymbol{\theta}_k^{\mathrm{T}}\boldsymbol{\theta}_k \tag{2.25}$$

This transformation using the logarithm can be safely applied, as the logarithm is a monotone function and therefore does not change the location of the maximum [Prince, 2012]. We can use gradient descent [Boyd and Vandenberghe, 2004] on the negative log likelihood to optimize Equation 2.25, where we need the gradient and hence the partial derivatives in respect to $\boldsymbol{\theta}_j$:

$$\frac{\partial \ln L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j} = \sum_i \mathbf{1}\{y_i = j\}\boldsymbol{x}_i - \frac{1}{\sum_k \exp\left(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i\right)} \cdot \exp(\boldsymbol{\theta}_j^{\mathrm{T}} \cdot \boldsymbol{x}_i) \cdot \boldsymbol{x}_i - \lambda \cdot \boldsymbol{\theta}_j \tag{2.26}$$

$$= \sum_i \left(\mathbf{1}\{y_i = j\} - \frac{\exp(\boldsymbol{\theta}_j^{\mathrm{T}} \cdot \boldsymbol{x}_i)}{\sum_k \exp(\boldsymbol{\theta}_k^{\mathrm{T}} \cdot \boldsymbol{x}_i)}\right) \cdot \boldsymbol{x}_i - \lambda \cdot \boldsymbol{\theta}_j \tag{2.27}$$

$$= \sum_i \Big[\mathbf{1}\{y_i = j\} - P(y = j|\boldsymbol{x}_i)\Big] \cdot \boldsymbol{x}_i - \lambda \cdot \boldsymbol{\theta}_j \tag{2.28}$$

indicator function

Here, $\mathbf{1}\{s\}$ refers to the *indicator function*, which returns 1 if statement s is true and 0 otherwise. A more efficient optimization method is L-BFGS [Byrd et al., 1995], which approximates the Hessian and therefore can scale the gradient for faster convergence.

However, optimizing the objective 2.25 using the gradient is usually prone to numerical overflows, if the arguments of the exponentiation gets too large. Far more stable is to exploit

the following equivalence:

$$\frac{\exp\left(\boldsymbol{a}^{(i)}\right)}{\exp\left(\sum_j \boldsymbol{a}^{(j)}\right)} = \frac{\exp\left(\boldsymbol{a}^{(i)}\right) \cdot \exp(z)}{\exp\left(\sum_j \boldsymbol{a}^{(j)}\right) \cdot \exp(z)} \tag{2.29}$$

$$= \frac{\exp\left(\boldsymbol{a}^{(i)} + z\right)}{\exp\left(\sum_j \boldsymbol{a}^{(j)} + z\right)}. \tag{2.30}$$

We set $z = -\max_j \boldsymbol{a}^{(j)}$, resulting in smaller arguments for the exponentiation, even if the weight vectors $\boldsymbol{\theta}_j$ get large.

Using the derivations of the objective, Equation 2.25, and the gradient, Equation 2.28, we can optimize the model parameters $\boldsymbol{\theta}$ using labeled training data with the help of L-BFGS. For inference, we only have to compute Equation 2.15 with the optimal parameters $\boldsymbol{\theta}^\star$ to determine the probability for a given class k. In Chapter 3, we will show that such a linear model can be as effective as more complex models using suitable features. We will then extend the very efficient softmax regression in Chapter 4 and Chapter 5 to improve the label consistency and get furthermore a more flexible approach for segment-based classification.

2.2.2 k-Nearest Neighbor Classification

The *k-nearest neighbor (knn) classifier* is a different approach, which allows more complex dependencies between features and the class label. Despite this flexibility, it is the simplest model to learn – we just have to store the entire training data set including the labels! knn classifier

Let $\mathcal{X} = \{(\boldsymbol{x}_0, y_0), \ldots, (\boldsymbol{x}_N, y_N)\}$ be the training set and $\hat{\boldsymbol{x}}$ an unseen feature vector for which we want to estimate $P(\hat{y}|\hat{\boldsymbol{x}})$. The k-nearest neighbor classifier models $P(\hat{y}|\hat{\boldsymbol{x}})$ as follows:

$$P(\hat{y}|\hat{\boldsymbol{x}}) = \frac{1}{k}\left|\left\{x_i \in \mathcal{N}_{\hat{\boldsymbol{x}}}^k | y_i = \hat{y}\right\}\right| \tag{2.31}$$

Thus, the probability of assigning a certain class does only depend on the distribution of class labels of the k nearest neighbors. As shown earlier in Section 2.1.2, we can build a k-d tree storing the training feature vectors to considerably accelerate the nearest neighbor search.

2.2.3 Model Assessment

In the previous chapters, we introduced two models for classification with very different properties. Softmax regression induces linear decision boundaries and needs a quite com-

plicated optimization for fitting the model parameters. On the other hand, the *k*-nearest neighbor can model arbitrary distributed datasets and the learning is very easy to implement.

It might appear that using *k*-nearest neighbor classifier is a good choice, but this is not always true. *K*-nearest neighbor is far more flexible, but this flexibility also introduces a high variance in the resulting decision regions – small variations in the training data can drastically change the decision boundaries. Softmax regression is less affected by the specific distribution of the training data, but imposes rather strong restriction to the shape of the decision boundaries. Thus, softmax regression shows a large bias towards the appearance of the decision regions, but a small variance in decision regions due to changes in the training data. Whereas the *k*-nearest neighbor shows an opposite behavior, small bias and high variance for small *k*. This so-called *bias/variance trade-off* occurs generally in supervised classification — having a higher bias incurs usually low variance, and vice versa.

bias/variance trade-off

Another problem might be the amount of training data needed to get a good model using a *k*-nearest neighbor classifier. Suppose, we try to learn a *k*-nearest neighbor classifier of a dataset, where the class of feature vectors is locally consistent. Furthermore, suppose it is sufficient to regularly sample data points in each dimension – say only 10 samples per dimension. If we now have a 1 dimensional feature vector, we need consequently only 10 examples to model the data perfectly; feature vectors of 2 dimensions, we need $10 \cdot 10 = 100$ examples, and so on. With only 12 dimensions, we would need in this thought experiment 10^{12} training examples, which is more than the number of stars in the Milky Way Galaxy [Swift et al., 2013]. It should be obvious that this amount of data is simply not manageable and this effect is usually known as *curse of dimensionality*. Nonetheless, real world data is usually restricted to a subspace and might show dependencies between feature values, which can be exploited to get reasonable results even with smaller training sets.

curse of dimensionality

Despite these considerations, which of the aforementioned approaches is now more effective in a certain scenario? As already seen, we can perfectly predict the class of every training case, if we use an 1-nn classification model. Hence, we are unable to make sensible conclusions about the quality of a model, i.e., how well the model represents real data, using only training data. Consequently, training error is a bad estimate of the quality and we have to rely on other measures.

A good starting point to estimate the quality of a learned model is the usage of a labeled *validation set*, which is not used to train the model. Since we know the label of every instance in the validation set, we can determine the predicted labels of our learned model and compare the prediction with our expected label. The ratio of wrongly predicted instances divided by the overall number of classified instances is now the *validation error*. The validation error is an estimate of the resulting test error, but is strongly influenced by the choice of the validation set. The influence of a specific choice of the validation set is minimized in the *cross-validation*, where we randomly split the labeled data into multiple parts and

validation set

validation error

cross-validation

take every part as separate validation set. The average of the resulting validation errors is a more accurate estimate of the test error. However, the validation error of one fold might be strongly influenced by the class distribution in the fold. Stratification is a common practice to reduce the influence of a dominating class and therefore reduces the variance in the validation errors. Here, the labeled data is split into parts with the same class distribution, i.e., every validation set contains the same number of instances of each class in every fold. Thus, the classification error is less influenced by the composition of the validation set.

stratification

A discrepancy between training error and (cross-)validation error is often an indicator for *over-fitting*. Over-fitting happens when we fitted our model parameters such that we are only able to predict the training set correctly. Over-fitting can be combated by using larger training sets, learning models with higher bias and therefore smaller model capacity, or regularizing the model parameters.

over-fitting

2.3 Summary

In this chapter, we briefly introduced concepts needed for the understanding of the rest of the thesis. We first discussed several aspects of three-dimensional point cloud processing and showed some essential procedures. The main part of this chapter covered different concepts of supervised classification and introduced the terminology. We introduced two basic classification models with very different capabilities – the softmax regression and the k-nearest neighbor classifier. In particular, we presented the softmax regression in greater detail, since it will be the basis for our own extensions in later chapters. Last, we outlined methods for assessing the quality of such models including cross-validation and stratification.

This chapter covers only machine learning concepts relevant for the understanding of the next chapters. Our aim was to introduce these concepts in a very concise manner. We refer to Prince [2012][5] for a more detailed discussion of logistic regression and different variants of this model. Another thorough introduction to different aspects of probabilistic classification is given by Bishop [2006], from a more statistical view point by Hastie et al. [2009][6] and more bayesian way of an introduction is used by Barber [2012][7]. In context of computer vision applications, Prince gives a very good introduction to classification in his book [Prince, 2012]. An excellent introduction to general convex optimization is given by Boyd and Vandenberghe [2004][8].

[5] See http://www.computervisionmodels.com/ [Accessed: 10 Oct 2013] for a free online version.
[6] Available at http://www-stat.stanford.edu/~tibs/ElemStatLearn/ [Accessed: 10 Oct 2013]
[7] See http://web4.cs.ucl.ac.uk/staff/D.Barber/pmwiki/pmwiki.php?n=Brml.Online [Accessed: 10 Oct 2013]
[8] Available at http://www.stanford.edu/~boyd/cvxbook/ [Accessed: 10 Oct 2013].

Next Chapters. In upcoming chapters, we investigate different aspects of the classification of three-dimensional laser range data in outdoor environments. We are interested in assigning the objects visible in the laser range scan a semantic label. For this purpose, we apply descriptors to get a descriptive representation of a laser point and its neighbors. Such feature vectors are then used to determine the object classes by using supervised classification models.

In the next Chapter 3, "Histogram Descriptors for Laser-based Classification," we evaluate different choices for such descriptors with the aim to determine suitable parameter ranges and reference frames. We additionally compare the softmax regression with a more complex graph-based model, the Functional Max-Margin Markov Networks. In the following Chapter 4, "Efficient hash-based Classification," we use the insights from the comparison to develop a new classification model combining nearest neighbor classification and softmax regression. Chapter 5, "Segment-based Classification," presents our work on a segment-based classification approach further improving the consistency of the point-wise classification results.

Chapter 3

Histogram Descriptors for Laser-based Classification

The classification of three-dimensional laser range data comprises two components — the classification *model* and the *data*. Recently, much scientific work concentrated on the development of more complex and expressive models, such as Conditional Random Fields [Agrawal et al., 2009, Anguelov et al., 2005, Munoz et al., 2009a, Triebel et al., 2006], or stacked classification [Xiong et al., 2011]. Nonetheless, we also have to consider the data part for the development of a robust classification approach, namely the extracted features. The classification model and the features are two sides of the same coin: a more complex model can compensate for insufficient features, and better features can compensate a too simplistic model. Put differently, a linear classifier with features capable of linearly separating the different classes should be ideally as effective as more complex and non-linear classifier with very simple features.

In this and the following chapter, we aim at predicting the class of every laser range point, as we do not only want to classify distinct objects with well-defined boundaries, but also surfaces with less clearly defined boundaries, such as ground, vegetation, and tree canopies. However, we cannot expect to get sensible conclusions about the class from a single three-dimensional point. Hence, we always build a more descriptive feature vector using the point and its neighboring points – the so-called *support*. A feature vector contains properties or statistics of the support and in this chapter we are particularly interested in histograms, since this type of descriptors is prevalent in current research.

support

As introduced in Chapter 2.1, “Three-dimensional Point Cloud Processing,” entails the usage of laser range data some specific challenges. One of these challenges is the distance

dependent coverage with laser range measurements of the scanned objects; we usually encounter very dense point clouds near the sensor and contrariwise very sparse point clouds at far distances. We therefore have to ensure range invariance of the generated feature vector and consequently normalize the feature vector to get a distance independent description.

We thoroughly investigate critical parameters of different histogram-based features for the classification of rigid outdoor objects. As stated earlier, we are particularly interested in a point-wise classification to distinguish surface properties or objects with vague boundaries, such as vegetation. Hence, we cannot exploit the range data in terms of first generating a segmentation and then classifying the segments [Himmelsbach et al., 2009], or even use tracks to segment dynamic objects of interest [Teichman et al., 2011].

More precisely, we are interested in answering the following questions: (1) What do we expect from feature representations to get a robust and state-of-the-art classification result? (2) Which feature representations are in this sense suitable to classify laser range data of an urban environment? And (3), which parameters are required to attain state-of-the-art classification results?

In this chapter, we show experimental results on three urban datasets generated using sensor setups introduced in Section 2.1.1 — sweeping 2D lasers, tilting 2D lasers, and a Velodyne 3D laser range scanner. Furthermore, we propose a novel histogram descriptor, which relies on the spectral values in different scales. We employ softmax regression (see Section 2.2.1) and a more complex collective classification approach [Munoz et al., 2009a]. As discussed earlier, the softmax regression facilitates very efficient efficient inference, but uses only the feature representation of a single point to deduce a label – this corresponds to a local classification. The second approach uses label information of neighboring points to smooth the individual classification results of a laser point and implements the most widely used state-of-the-art approach for point-wise classification. However, this so-called collective approach needs a graph defining the neighbor relations and furthermore needs a more complex inference scheme to propagate label information through the graph, which is also more time consuming than a local classification approach. These different capabilities motivates also the investigation of the duality mentioned in the beginning: Do more complex features enable a local classifier to attain results that are similar to the results of a more complex collective classification approach using simple features?

The contents of this chapter were partially published in [Behley et al., 2012] and will be presented in more detail in this thesis. In addition to these earlier evaluation, we also discuss the classifier performance more detailed and evaluate the runtime performance of the descriptors.

In the computer vision community several studies on the quality of descriptors for matching and object recognition were conducted [Kaneva et al., 2011, Mikolajczyk and Schmid, 2005]. Three-dimensional point cloud descriptors were mainly investigated in context of

shape retrieval [Johnson and Hebert, 1999, Tangelder and Veltkamp, 2008]. However, for the purpose of (point-wise) classification of three-dimensional laser range data, only a very few studies were conducted [Rusu et al., 2008]. To the best of our knowledge is this the first thorough experimental investigation of descriptors in the context of classification of three-dimensional laser range data.

The rest of the chapter is organized as follows. In Section 3.1, "Related Work," we introduce recent work in the context of the performance evaluation of histogram-based features. In Section 3.2, "Histogram Descriptors," we describe the evaluated histogram-based descriptors concentrating on descriptors used in previous work on point-wise classification. Then in Section 3.3, "Reference Frame and Reference Axis," we discuss different reference frames, a local and a global variant. The next Section 3.4, "Experimental Setup," specifies the methodology of the performance evaluation, the evaluated datasets, and the investigated classification approaches. In Section 3.5, "Results and Discussion," we discuss the experimental results and present the main findings of our performance evaluation. Finally, in Section 3.6, "Summary," we summarize the main contributions of the chapter and outline future work.

3.1 Related Work

Local three-dimensional shape descriptors, as used in this chapter, were especially evaluated in context of shape retrieval applications. In shape retrieval, one is interested in retrieving similar objects to a selected query object from a large database of three-dimensional objects, either represented by meshes or point clouds. See the survey of Tangelder and Veltkamp [2008] for an extensive overview of the field. A whole workshop series, the Eurographics Workshop on 3D Object Retrieval, covers three-dimensional object retrieval. In conjunction with this workshop, the Shape Retrieval Contest (SHREC) compares the current state-of-the-art in shape retrieval in different categories, such as "Generic 3D Model Retrieval" [Li et al., 2012]. However, the contest aims at comparing the retrieval performance of complete methods, which includes features, but also specifically tuned parameters by the competing researchers.

While some of these methods could be applied to extract useful feature representations for the classification of laser range data, we generally pursue a different objective. The object retrieval from shape databases aims at finding an instance of the database, which is very similar to the queried object. Therefore, the employed methods aim at deriving very detailed representations that enables a matching approach to distinguish different instances of the same category. In our application, we are more interested in deriving a feature representation enabling us to distinguish different categories rather then single instances.

In recent years, many approaches for the classification of three-dimensional laser range data [Agrawal et al., 2009, Anguelov et al., 2005, Munoz et al., 2009a, Triebel et al., 2006] and [Spinello et al., 2011, Teichman et al., 2011, Xiong et al., 2011] proposed different local features. These features usually are chosen to suit the specific application, but an evaluation on the influence of parameter choices is missing. Most approaches combine multiple features, ranging from simple statistical properties to more complex shape histograms. Rusu et al. [2008] compared their method with different other classifiers – SVMs with different kernels, *k* nearest neighbors and *k*-means with different distance metrics. Hence, their experimental evaluation concentrates mainly on the performance of different classification methods, but not on the parameters of the employed descriptors.

Recently, Arbeiter et al. [2012] evaluated different local descriptors for the classification of surface properties, i.e., planar, edge, corner, cylindrical and spherical. They evaluated the Fast Point Feature Histograms [Rusu et al., 2009], Radius Surface Descriptors [Marton et al., 2010], and so-called Principle Curvatures using cluttered indoor environments. In contrast to the evaluation presented in this chapter, they focused on accuracy and runtime with two fixed parameter settings for close and far range, respectively.

3.2 Histogram Descriptors

descriptor

histogram descriptors

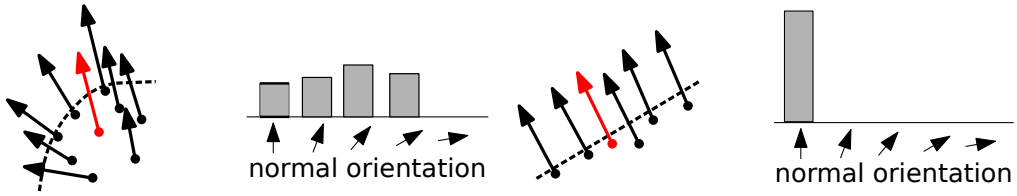
reference axis and frame

In the following, we use the term *descriptor* to denote a vectorized feature representation, which is a discriminative representation of a laser point and its neighborhood instead of a single shape property. We focus here on *histogram descriptors* [Tombari et al., 2010] maintaining a histogram of neighboring points or their properties. For the histograms, we need a *reference axis* or *reference frame* in which we determine the bin index of the property we want to measure.

Over the last years, a variety of descriptors for matching of point clouds [Rusu et al., 2009, Tombari et al., 2010], object recognition [Johnson and Hebert, 1999, Steder et al., 2011b] and point-wise classification [Agrawal et al., 2009, Munoz et al., 2009a, Triebel et al., 2006, Xiong et al., 2011] representing properties of the support of a point were proposed. In this section, we briefly introduce histogram descriptors used in recent work, which emerged to be a good choice for a descriptive representation of laser points in terms of shape and geometry.

requirements

We have some special requirements on descriptors for point-wise classification of three-dimensional laser range data. We want to distinguish between different classes or categories, but not single instances like in shape retrieval. In addition, the description should result in well separated and localized clusters in the feature space, which enables the usage of simpler and therefore more efficient classification approaches. We furthermore want a robust feature representation, which is only marginally affected by partial occlusions often



(a) (b)

Figure 3.1: *Normal histogram for curved and flat surfaces. In both images the query point and the corresponding reference axis, i.e., the normal of the point, is highlighted in red. A curved surface leads to a more uniform distribution of histogram entries, whereas a flat surface induces a more peaked histogram as shown in (a) and (b), respectively.*

encountered in real-world laser range scans. Last, we are looking for descriptors that can handle different sparsities of object point clouds. This requirement is seldom encountered in shape retrieval applications, where we find similar sampling rates in the database, and indoor object recognition applications, as there we usually encounter near range scans.

The descriptors that we present in the following sections were selected in respect to these requirements and we investigate their capabilities to produce general descriptions and also well separated clusters in feature space for efficient point-wise classification of rigid outdoor objects. Following the taxonomy of Tangelder and Veltkamp [2008], these descriptors can be classified as *local features*, since they represent the local neighborhood of a point instead of determining a global description of the whole segmented object. Thus, we get a local representation, which is less affected by partial occlusions and additionally independent of a given segmentation. As all descriptors use a radius neighborhood $\mathcal{N}_p^\delta$, we get a sampling invariant representation by a proper normalization of the feature vectors.

local features

The *normalization constant* will be denoted by η and calculated separately for each feature vector. We empirically determined that normalizing the feature vector $\boldsymbol{v}$ with the maximal entry $\eta = \max_i \boldsymbol{v}^{(i)}$ is superior to a normalization by the sum of all entries. We use $\boldsymbol{r} \in \mathbb{R}^3$ to refer to the reference axis and $\mathbf{R} \in \mathbb{R}^{4\times4}$ to denote the reference frame used to determine the histogram indices.

normalization constant

Histogram of Normal Orientations. Triebel et al. [2006] used a *normal histogram* storing the angle between the reference axis $\boldsymbol{r}$ and the normal of a neighboring point $\boldsymbol{n}_q$, $\boldsymbol{q} \in \mathcal{N}_p^\delta$, as depicted in Figure 3.1. The histogram descriptor $\boldsymbol{h} \in \mathbb{R}^b$ with b entries is defined as follows:

normal histogram

$$h_i = \eta \left| \left\{ \boldsymbol{q} \;\middle|\; \cos\left(\frac{i \cdot \pi}{b}\right) \leq \boldsymbol{r} \cdot \boldsymbol{n}_q < \cos\left(\frac{(i+1) \cdot \pi}{b}\right) \right\} \right| \tag{3.1}$$

Regions with a strong curvature result in a uniformly distributed histogram, while flat areas lead to a peaked histogram (see Figure 3.1). The histogram is parameterized by the number of bins b and the size of the support region δ.

spin image

Spin Image. The *spin image* by Johnson and Hebert [1999] is the most prominent histogram descriptor and is used in several retrieval, matching or classification approaches [Patterson et al., 2008, Teichman et al., 2011, Xiong et al., 2011]. A spin image can be imagined by spinning a grid around the reference axis $\boldsymbol{r}$, where grid cells "collect" or "count" the neighboring points $\boldsymbol{q} \in \mathcal{N}_p^\delta$. An entry of the spin image $\mathbf{SI} \in \mathbb{R}^{b \times b}$ with indexes (i, j) is calculated using the distance to the line defined by $\boldsymbol{p} + \lambda \cdot \boldsymbol{r}$ with parameter $\lambda \in \mathbb{R}$, and the distance to the plane originating in $\boldsymbol{p}$ and normal $\boldsymbol{r}$. The local coordinates (α, β) in respect to the reference axis are given by $\alpha = \|\boldsymbol{r} \times (\boldsymbol{q} - \boldsymbol{p})\|$ and $\beta = \boldsymbol{r} \cdot (\boldsymbol{q} - \boldsymbol{p})$. The indices (i, j) in the image are calculated from α and β by $i = \lfloor \rho^{-1} \cdot \alpha \rfloor$ and $j = \lfloor \frac{1}{2}\rho^{-1} \cdot (\beta + \delta) \rfloor$, where $\rho = \delta b^{-1}$ is the grid resolution of the spin image. We bilinearly interpolate the contributions to avoid quantization artifacts. The spin image is parameterized by the number of bins b (width and height of the spin image) and the radius of the support δ.

Note that we search for radius neighbors using the maximum norm, i.e., $\boldsymbol{q} \in \mathcal{P}$ is a neighbor of $\boldsymbol{p}$, if $\max_i |p^{(i)} - q^{(i)}| < \delta$, since we need all neighbors in a cylinder. We approximate this with our euclidean neighbor data structures form Section 2.1.2 by increasing the radius δ by a factor of $\sqrt{3}$, which corresponds to the diagonal of a cube with side length δ.

distribution histogram

Distribution Histogram. The *distribution histogram* by Anguelov et al. [2005] tries to capture the shape around a point in a cube defined by the reference frame $\mathbf{R} \in \mathbb{R}^{4x4}$. In order to transform a neighboring point $\boldsymbol{q} \in \mathcal{N}_p^\delta$, the reference frame is inverted, i.e., $\boldsymbol{q}' = \mathbf{R}^{-1}\boldsymbol{q}$. The distribution histogram $\boldsymbol{h} \in \mathbb{R}^{b \times b \times b}$ is then defined as follows:

$$\boldsymbol{h}_{i,j,k} = \eta \left| \left\{ \boldsymbol{q}' \left| \left\lfloor \frac{b}{2} \cdot \left(\frac{q'}{\delta} + \mathbf{1} \right) \right\rfloor = \begin{pmatrix} i \\ j \\ k \end{pmatrix} \right\} \right|, \tag{3.2}$$

where $\mathbf{1} \in \mathbb{R}^3$ denotes the vector that contains only ones.

The only parameter of the distribution histogram is the number of bins b per dimension. Similar to the Spin Image, we approximate the search for neighbors inside the cube by multiplying the support radius δ by $\sqrt{3}$.

SHOT

Signature of Histograms of Orientations (SHOT). Recently, Tombari et al. [2010] proposed to use a combination of histograms and signatures. Their descriptor subdivides the

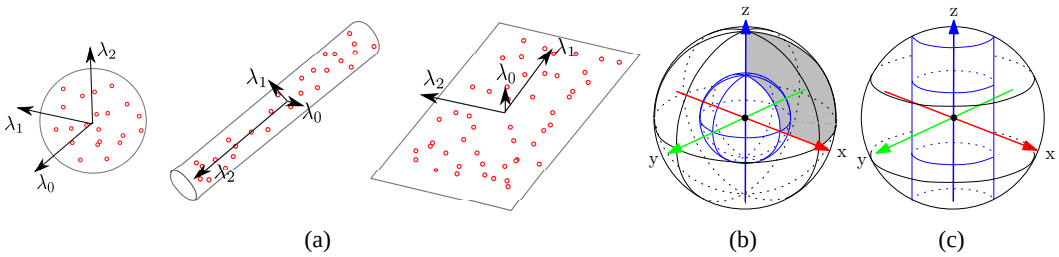


Figure 3.2: *In the first image, different point distributions are shown and their corresponding eigenvalues λ_i are depicted by appropriately scaled vectors. For a point-like distribution the resulting eigenvalues are almost equal. In case of a linear distribution of the points, we can see a significantly larger eigenvalue λ_2 compared to λ_1 and λ_0. Lastly, we can observe in flat distributions of points a very small λ_0 and almost equally large λ_1 and λ_2. Subdivisions used by panel (b) the SHOT descriptor and panel (c) the spectral histogram. The inner sectors or shells are depicted in blue. In panel (b) one sector of the SHOT descriptor is highlighted in light gray.*

space around the query point $\boldsymbol{p}$ into sectors (see Figure 3.2b). Then, for every sector a histogram of normal orientations between the neighboring point inside the sector and the query point is calculated.

More precise, the histogram index i of a neighboring point $\boldsymbol{q} \in \mathcal{N}_p^\delta$ inside a sector is calculated by $\frac{1}{2}(1 + \boldsymbol{r} \cdot \boldsymbol{n}_q)b$, where b denotes the number of bins in the histogram and $\boldsymbol{n}_q$ is the estimated normal of point $\boldsymbol{q}$. A point also contributes to histograms in neighboring sectors of the subdivision using a quadrilinear interpolation to reduce quantization errors.

The authors suggested to use 8 azimuth divisions, 2 elevation divisions and 2 radial divisions for the subdivision. The remaining parameters of interest are the radius of the support region δ, and the number of bins in the sector histograms b.

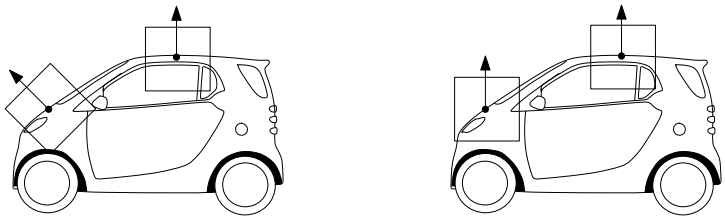
Spectral Histogram. Motivated by the results of experiments with spectral shape signatures, we propose to use a *Spectral Histogram* [Behley et al., 2012]. Similar to the SHOT descriptor, we calculate for every sector of a subdivision three signature values based on spectral values of points falling inside the sector.

Spectral Histogram

The eigenvalues of the covariance matrix $\mathbf{C} \in \mathbb{R}^{3\times3}$, as introduced in Section 2.1.3, encode the general distribution of the points (see Figure 3.2a). Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of the covariance matrix $\mathbf{C}$ and $\hat{\lambda}_i = \lambda_i/\lambda_2$ the normalized eigenvalues. A measure of "point-ness" is then defined by $\hat{\lambda}_0$, "surface-ness" by $\hat{\lambda}_1 - \hat{\lambda}_0$, and "linear-ness" by $\hat{\lambda}_2 - \hat{\lambda}_1$ [Medioni et al., 2000, Munoz et al., 2009a]. We will refer to these properties as *spectral shape features* in the following.

spectral shape features

From these spectral shape features we build a descriptive representation of the support as follows. We subdivide the space around a point in different shells and slices, as shown in Figure 3.2c. Let s be the number of radial shells, l the number of slices, and $\boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{v}_z$ the



(a) (b)

Figure 3.3: *Panel (a) depicts a local reference frame depending on the local normal orientation for two query points. Panel (b) shows the same query points using a global reference frame that is always pointing upwards.*

base vectors of the reference frame. Then we add to the local covariance of sector (i, j) the point $\boldsymbol{q'} = \mathbf{R}^{-1}\boldsymbol{q}$, if $i = \lfloor \frac{1}{2\delta}(q'^{(3)} + \delta)l \rfloor$ and $j \leq \lfloor \frac{1}{\delta}\|\boldsymbol{q'} \times \boldsymbol{v}_z\|s \rfloor$. Hence, every shell collects all points up to its radius. For every radial shell in every slice, we get a different scale of the point distribution. The descriptor is rotation invariant around the z axis of the reference frame and parameterized by the number of slices l, the number of radial shells s, and the support radius δ.

3.3 Reference Frame and Reference Axis

The only question left is the choice of the reference frame or the reference axis, which are required to calculate the indices in the histograms. We evaluated two canonical choices—the local reference frame based on eigenvectors and a global reference frame based on the global z-axis (see Figure 3.3).

local reference frame

The *local reference frame* $\mathbf{R}_{\text{local}} \in \mathbb{R}^{4\times4}$ of a point $\boldsymbol{p}$ is based on the normalized eigenvectors $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2$ of the covariance matrix of neighboring points $\boldsymbol{q} \in \mathcal{N}_p^\delta$. From the eigenvectors with eigenvalues $\lambda_0 \leq \lambda_1 \leq \lambda_2$ we can build the following homogeneous transformation:

$$\mathbf{R}_{\text{local}} = \begin{bmatrix} \boldsymbol{v}_2 & \boldsymbol{v}_1 & \boldsymbol{v}_0 & \boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

A local reference axis $\boldsymbol{r}_{\text{local}} \in \mathbb{R}^3$ is given by $\boldsymbol{v}_0$, which corresponds to the point normal $\boldsymbol{n}_p$ of point $\boldsymbol{p}$.

global reference frame

The *global reference frame* can be constructed using the global z-axis denoted by $\boldsymbol{z}$. We decided to use the normal $\boldsymbol{n}_p$ to get a rotation invariant reference frame. Following from this we get the global reference frame $\mathbf{R}_{\text{global}} \in \mathbb{R}^{4\times4}$:

$$\mathbf{R}_{\text{global}} = \begin{bmatrix} \frac{(\boldsymbol{n}_p \times \boldsymbol{z}) \times \boldsymbol{z}}{\|(\boldsymbol{n}_p \times \boldsymbol{z}) \times \boldsymbol{z}\|} & \frac{\boldsymbol{n}_p \times \boldsymbol{z}}{\|\boldsymbol{n}_p \times \boldsymbol{z}\|} & \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|} & \boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

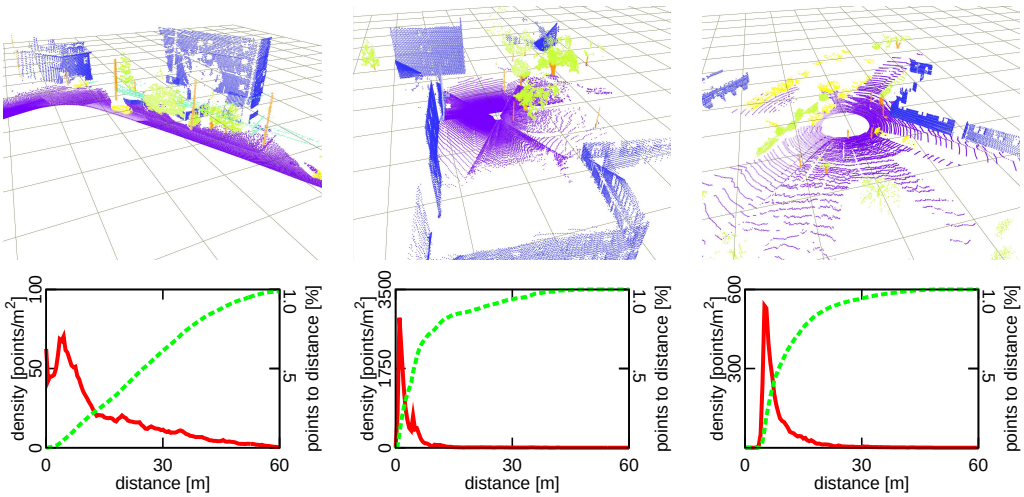


Figure 3.4: *Panel (a) depicts a laser range scan from Pittsburgh, panel (b) from Freiburg, and panel (c) from Wachtberg, respectively. The different labels are colored as follows: purple = ground, blue = building facades, green = vegetation, orange = poles, yellow = vehicles, cyan = wire. The distribution of laser returns per distance to the laser scanner is depicted below every scan. The red solid curve depicts the number of laser points per m^2 at this distance. The green dashed curve is the fraction of laser returns up to the distance.*

Tombari et al. [2010] proposed to use a weighted version of the covariance for determining the eigenvectors, thus getting a more stable reference frame in point clouds with clutter and also a disambiguation scheme of the directions of the eigenvectors based on the point density. We applied this weighted covariance and the disambiguation scheme only to the SHOT descriptor.

3.4 Experimental Setup

In the upcoming sections, we introduce the framework of our evaluation before we discuss the experimental results. Starting with the datasets, we will introduce the evaluation criterion and finally briefly introduce the evaluated classification approaches.

Datasets. In the following evaluation we use datasets generated by three common 3D laser rangefinder setups—a pan-tilting 2D laser rangefinder, 2D sweeping laser rangefinders, and a Velodyne HDL64-E laser rangefinder [Velodyne Lidar Inc., 2010], which we already encountered in Section 2.1.1, “Data Acquisition.” Figure 3.4 depicts example scans from these datasets and the distribution of laser returns for the specific setup.

The first dataset was recorded at the University of Freiburg, Germany, using a SICK LMS laser rangefinder mounted on a pan-tilt unit. The point clouds were manually labeled[1] as pavement, sidewalk, lawn, pole, shrub, bush, foliage, tree trunk, building facade, window, door, bicycle, and car. For the evaluation we only used a subset of these classes and combined subclasses into more general classes: (1) *ground* consisting of pavement, sidewalk, and lawn, (2) *vegetation* containing shrub, foliage, and bushes, (3) *facades* subsuming building facades, doors, and windows, (4) *poles* combined with tree trunks.

We chose these more general classes because they contain the surfaces and objects most relevant for outdoor applications. Furthermore, the distinction of pavement, sidewalk and lawn is often only possible by using contextual knowledge. Poles are in particular interesting, because they allow to reveal registration errors and thus can be useful to assess the performance of SLAM approaches.

The second dataset was acquired on the campus of the Carnegie Mellon University in Pittsburgh with a Jeep equipped with SICK laser scanners facing sideways. The dataset contains the same labels as the Freiburg dataset, but we additionally use *vehicles* and *wire* like Xiong et al. [2011]. The dataset was filtered and registered to get a complete point cloud and chunks of approximately 100.000 laser points were extracted.

The last dataset was recorded at the Fraunhofer FKIE in Wachtberg, Germany, using a Velodyne HDL-64E S2 laser range scanner mounted on a vehicle. We also manually labeled the dataset with the classes *ground*, *vegetation*, *facades*, *vehicles*, and *poles*.

All three datasets show different characteristics. Figure 3.4 depicts the point density and the number of laser points per square meter. In case of the Pittsburgh dataset, we find homogeneous sampling of the surfaces and nearly linear increase of points per distance (green dashed curve in the plots). The Freiburg and Wachtberg dataset contrariwise show a significant drop in the sampling rate at larger distances, which is common for raw data without a specific preprocessing. As the Velodyne HDL-64 rotates to generate a full 360° scan, we also see a ring pattern with points in the same ring much closer to each other than points in adjacent rings.

All these artifacts in the data acquisition must be considered, while designing a classification approach. A normalization of the feature vectors to account for differences in the number of laser returns at different distances is essential to get similar feature vectors of the same object.

[1] The registered laser range scans of the Freiburg campus with the robot odometry are available at http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/ [Accessed: 10 Oct 2013]. Our annotations of the point clouds can be downloaded at http://www.iai.uni-bonn.de/~behley/data/.[Accessed: 10 Oct 2013]

Evaluation Criterion. In Section 2.2.3, "Model Assessment," we introduced cross validation to evaluate a classification approach and also the concept of stratification, where we split the data into folds of similar class distributions. As we want to keep the single scans spatially separated, we can not simply shuffle the feature vectors of different scans to ensure equal class distributions in every fold. But we still want to get an unbiased evaluation measure, which is not affected by the mere count of a single class, such as ground, like the classification accuracy. To this end, we will use here a different evaluation measure, which is now formally defined.

Let $\hat{\mathcal{X}} = \{\hat{\boldsymbol{x}}_i\}, |\hat{\mathcal{X}}| = M$ be a set of test instances with corresponding ground truth labels $\mathcal{Y}^\star = \{y_i^\star\}, |\mathcal{Y}^\star| = M$, and $\hat{y}_i$ the predicted label of a classifier trained on a separate training set $\mathcal{X} = \{(\boldsymbol{x}_i, y_i)\}, |\mathcal{X}| = N, \mathcal{X} \cap \hat{\mathcal{X}} = \emptyset$.

The *class-wise precision* p_k of a class or label k is given by the ratio of correctly classified instances $\hat{\boldsymbol{x}}_i$ with $\hat{y}_i = y_i^\star$ of the test set and all instances classified as class k:

$$p_k = \frac{\left|\left\{\hat{\boldsymbol{x}}_i \in \hat{\mathcal{X}} \,\middle|\, \hat{y}_i = k \wedge y_i^\star = k\right\}\right|}{\left|\left\{\boldsymbol{x}_i \in \hat{\mathcal{X}} \,|\, \hat{y}_i = k\right\}\right|} \tag{3.5}$$

The *class-wise recall* r_k is given by the ratio of correctly classified instances and all instances with reference label k:

$$r_k = \frac{\left|\left\{\hat{\boldsymbol{x}}_i \in \hat{\mathcal{X}} \,\middle|\, \hat{y}_i = k \wedge y_i^\star = k\right\}\right|}{\left|\left\{\boldsymbol{x}_i \in \hat{\mathcal{X}} \,\middle|\, y_i^\star = k\right\}\right|} \tag{3.6}$$

The F_1 *measure* is defined as the average over the class-wise precisions p_k and recalls r_k:

$$F_1 = \frac{1}{K} \sum_k \frac{2 \cdot p_k \cdot r_k}{p_k + r_k}, \tag{3.7}$$

where K is the number of classes. The F_1 measure is independent of the actual number of instances, because it uses only relative measures, and penalizes high precision with low recall.

Classification Approaches. We evaluated the descriptor performance using two different classification approaches — the already introduced softmax regression and a more complex approach based on a Conditional Random Field (CRF) [Lafferty et al., 2001], which is the most prominent method to classify three-dimensional laser range data.

The softmax regression introduced in Section 2.2.1 acts as a baseline approach in this study. The main advantage of softmax regression is the fast inference given a learned weight vector, local classification

which is of particular interest if we want to classify millions of laser range measurements. However, it is a local classification approach that uses only information encoded in the descriptor. Thus, the overall posterior $P(y_{1:N}|\boldsymbol{x}_{1:N})$ of a laser range scan factors into independent parts $P(y_i|\boldsymbol{x}_i)$ assuming independence between labels of different points given the descriptor.

collective classification

The local classifier is compared with a state-of-the-art collective classification approach using a CRF—the Functional Max-Margin Markov Networks (FM3N) [Munoz et al., 2009a]. Collective classification approaches, which take labels of neighboring points into account to estimate a laser point label, have shown to be quite effective [Anguelov et al., 2005, Munoz et al., 2009a, Triebel et al., 2006]. These approaches try to find the joint assignment of all labels $P(y_{1:N}|\boldsymbol{x}_{1:N})$, which maximizes the posterior of the joint assignment of classes to y_i given the features $\boldsymbol{x}_i$:

$$P(y_{1:N}|\boldsymbol{x}_{1:N}) = \frac{1}{Z(x_{1:N})} \prod_{y_i \in \mathcal{V}} \phi(y_i, x_i) \prod_{(y_i,y_j)\in\mathcal{E}} \psi(y_i, y_j, \boldsymbol{x}_i, \boldsymbol{x}_j), \tag{3.8}$$

where the underlying graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is given by vertices y_i representing random variables for labels and $\mathbf{x}_i$ for feature vectors of every laser point, and edges $(y_i, y_j) \in \mathcal{E}$ between them, if there exist a direct dependency. $\phi(y_i, x_i)$ refers to the node potential, which encode the compatibility of the label y and the feature vector $\boldsymbol{x}_i$, and $\psi(y_i, y_j, \boldsymbol{x}_i, \boldsymbol{x}_j)$ to the edge potential, which encodes the compatibility of labels y_i and y_j regarding the corresponding features $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. $Z(x_{1:N})$ denotes the partition function and is a normalizer depending on the feature vectors, and the computation of the normalizer is usually the reason for approximate inference. We refer to Koller and Friedman [2009] for more details on probabilistic graphical models.

The edges are given by the k-nearest neighbors of a laser point $\boldsymbol{p}_i$, i.e., $(y_i, y_j) \in \mathcal{E}$, if $\boldsymbol{p}_j \in \mathcal{N}^k_{\mathbf{p}i}$. As proposed by Xiong et al. [2011], we use a "similarity" edge potential $\psi(y_i, y_j, \boldsymbol{x}_i, \boldsymbol{x}_j)$ computed from the node features $\boldsymbol{x}_i, \boldsymbol{x}_j$:

$$\psi(y_i, y_j, \boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbf{1}\{y_i = y_j\} \cdot \exp(-w_i^T \cdot \boldsymbol{x_{ij}}), \tag{3.9}$$

where the k-th entry of $\boldsymbol{x}^{(k)}_{ij} \in \mathbb{R}^m$ is calculated as

$$\boldsymbol{x}^{(k)}_{ij} = \left(1 + \left|\boldsymbol{x}^{(k)}_i - \boldsymbol{x}^{(k)}_j\right|\right)^{-1}, \tag{3.10}$$

i.e., the more similar the node features, the larger the entries in the edge feature $\boldsymbol{x_{ij}}$.

An edge between individual labels y_i and y_j encourages smooth label assignments over parts of the graph. This can help to correct failures in the label assignment and furthermore

Table 3.1: *Parameters of the descriptors. Values used for node and edge potentials are bold.*

Descriptor	Parameter values
Normal Histogram (NH)	$b = \{\mathbf{5}, \mathbf{10}, \mathbf{15}\}$
Distribution Histogram (DH)	$b = \{\mathbf{3}, \mathbf{5}, 7\}$
Spin Image (SI)	$b = \{\mathbf{5}, \mathbf{10}, 20\}$
Signature of Histograms of Orientations (SHOT)	$b = \{\mathbf{5}, \mathbf{10}, 15\}$
Spectral Histogram (SH)	$l = \{\mathbf{3}, \mathbf{5}, 7\}$, $s = 5$

ensures consistent labels for a segment, but also causes errors when wrong information is propagated through the graph.

Compared to local classification models, we additionally have to determine the graph structure and this involves the time consuming construction of a nearest neighbor graph. Furthermore, the edge potential must be calculated for every edge and also separately stored for learning of the model. Thus, the overall preprocessing and also learning is very time consuming compared to local classification approaches, which only take the local neighborhood into account.

3.5 Results and Discussion

We performed a 5-fold cross validation (cf. Section 2.2.3) in all experiments. For this purpose, we selected 5 representative and non-overlapping 360° laser range scans from every dataset. We only evaluated a subset of parameters with the CRF, which allowed us to store the networks with node and edge potentials in memory. Hence, we were able to evaluate the CRF using large support radii in reasonable time. Table 3.1 shows the descriptor parameters and bold parameter values indicate the subset of parameters used for the CRF.

In this chapter, we present only a subset of the results for the sake of brevity. We concentrate here on our main findings and defer more detailed plots of the experimental results to the Appendix B, “Additional Results.”

Implementation details. The histogram descriptors were implemented using C++ and we adapted the available implementation of the SHOT descriptor from the Point Cloud Library(PCL) [Rusu and Cousins, 2011]. We used an octree to determine the nearest neighbors and the normals were estimated (cf. Section 2.1.3) using a radius of 0.6 m.

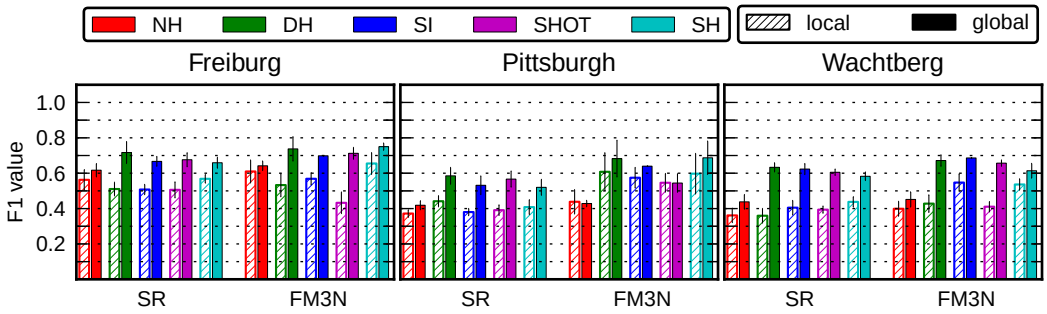


Figure 3.5: *Influence of the reference frame. The performance of the evaluated descriptors using the local reference frame and the global reference frame are shown. The results show a clear advantage of the global reference frame over the local reference frame with both classification approaches.*

The regularization parameter λ of the softmax regression (cf. Section 2.2.1) was set to 0.01 and the intercept was fixed to 1.0. The FM3N implementation[2] was adapted to our needs and used only pair-wise potentials with linear regressors. We choose 50 iterations to learn linear regressors with learning rate of 0.1. We used $k = 5$ nearest neighbors in all our experiments and restricted the radius to 2 m, i.e., we add at most 5 edges in the neighbor graph and all of neighbors must be inside a radius of 2 m. The relatively large number of nearest neighbors compared to other works using CRFs [Munoz et al., 2009a, Xiong et al., 2011] was motivated by the ring pattern of Velodyne laser scans. We experienced that less then 5 neighbors adds only edges inside a ring, but not between adjacent rings and increasing k helps to propagate label information between rings. All experiments were performed on an Intel Xeon X5550 with 2.67 GHz and 12 GB memory.

Reference frame. In a first set of experiments, we evaluated the influence of different reference frames and axes on the classification performance of the presented classifications approaches. For the matching task, Tombari et al. [2010] showed that the reference frame and its stability significantly affects the matching performance. Is this also true for the classification of three-dimensional laser range data?

Figure 3.5 shows the performance of the descriptors using the local and global reference frame. We show here the results using a support radius $r = 0.5$ m and a medium number of bins, i.e., 10 bins or 5 bins, respectively. We see a significant improvement using the global reference frame over the local reference frame with almost all descriptors regardless of the employed classification approach. In particular, the distribution histogram is strongly affected by the choice of the reference frame with improvements up to 40%. The normal histogram only shows a small effect if the reference frame is changed.

[2] Available at http://www.cs.cmu.edu/~vmr/software/. [Accessed: 14 Oct 2013]

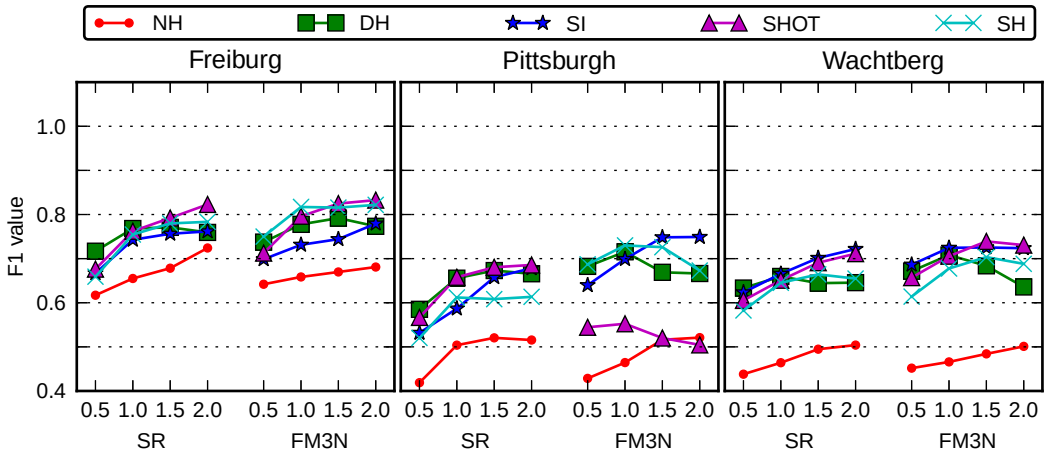


Figure 3.6: *Influence of the support radius on classification performance using a global reference frame. We see an improvement of the classification performance with an increasing support radius in all datasets.*

The presented results show a significant influence of the choice of the reference frame and confirm our hypothesis that a global reference frame is more stable than a local reference frame. For a robust and stable local reference frame, we have to ensure a reproducible calculation of the eigenvectors. However, the direction of the eigenvectors is affected by the distribution of the points and this might change drastically in the presence of clutter. Another explanation for the observable strong performance of the global reference frame might be the occurring object classes in the datasets. All datasets are mainly composed of man-made structures and objects, which show only a restricted number of orientations. For instance, cars never appear up-side down and building walls always have a vertical orientations. Thus, a z-axis based reference frame is more effective than a normal-based reference frame, since we do not have to cope with arbitrary rotations of the objects in urban environments. Additionally, we get more discriminative histograms for ground and facade points, which improves the distinction between these flat surfaces.

In the following discussion, we will use a global reference frame and vary only the remaining parameters. Results for the local reference frame can be found in the Appendix B, “Additional Results.”

Support radius. Figure 3.6 shows an increase in classification performance with increasing support radius for most descriptors. The improvements are not as large as with the reference frame, but using a larger support radius increases especially the performance of the softmax regression.

An explanation for this result might be an increasing support radius, which includes more

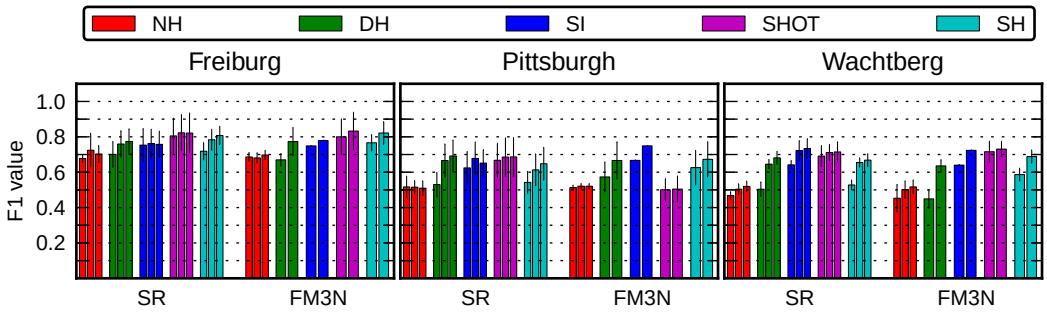


Figure 3.7: *The figure shows the effect of changing the number of bins for each descriptor using a support radius $r = 2\,\text{m}$ and a global reference frame. We see a small gain in classification performance with an increasing number of bins, but this effect is only considerable for few bins. The increase is negligible with more than* 10 *or* 5 *bins, respectively.*

contextual information of the vicinity. For example, all classifiers performed better in classifying cars if we increased the support radius. This consequence is remarkable, because cars are obviously rather complex objects with both flat and curved surfaces. Nevertheless, vehicles are usually parked/driven on flat ground, which seems to be the discriminating property of cars especially in the Pittsburgh dataset. Thus, we can indirectly encode—to some extent—the context in the descriptor, which helps to learn correct class assignments even with local classifiers. This is explicitly achieved in a collective approach by a more complex model, but can be also directly incorporated by stacking with multiple logistic regressions [Xiong et al., 2011].

In the Wachtberg dataset, such contextual information does not always lead to better performance of the classification approach, as there are shrubs/bushes on flat lawn. Hence, in this dataset the classes *vehicle* and *vegetation* are more often confused with other classes than for instance in the Pittsburgh dataset. We also experimented with larger radii than 2.0 m, but these turned out to entail no significant improvement, or even decreased the performance of the classification approaches.

In conclusion, the results indicate that a large support radius increases consistently the performance of most descriptors. Therefore, it is beneficial to use large descriptors in the context of urban environments. However, this advantage comes at the cost of increased computation time of the descriptors, since increasing the radius of the neighborhood also increases the search time of the octree neighbor search.

Number of Bins. The number of bins is the last dimension we varied for the different descriptors. Figure 3.7 shows the classification performance using different number of bins and a support radius $r = 2\,\text{m}$. We can also observe an increase of the overall accuracy using more bins, but this effect is not as strong as the other evaluated dimensions. The

Table 3.2: *Precision and recall on the Freiburg dataset.*

C	Feature	b	δ	F_1	ground	facade	pole	vegetation
SR	NH	15	2.0	70.3	98/95	94/88	12/24	82/86
	DH	7	1.5	77.7	99/99	90/82	57/56	77/79
	SI	10	2.0	76.2	99/99	84/81	55/62	79/74
	SHOT	10	2.0	82.3	99/99	93/91	60/63	85/84
	SH	7	2.0	80.8	98/98	93/87	51/66	84/83
FM3N	NH	15	2.0	69.7	98/95	96/88	6/10	83/87
	DH	5	1.5	79.2	99/99	92/84	53/66	80/82
	SI	10	2.0	77.9	99/99	90/83	46/71	81/80
	SHOT	10	2.0	83.3	99/99	94/90	61/68	85/87
	SH	5	2.0	82.2	100/98	94/87	47/78	84/88

Table 3.3: *Precision and recall on the Pittsburgh dataset.*

C	Feature	b	δ	F_1	wire	pole	ground	vegetation	facade	vehicle
SR	NH	5	1.5	53.3	18/43	1/25	99/98	92/90	89/72	25/41
	DH	7	2.0	69.2	24/42	55/59	99/99	94/87	79/81	65/69
	SI	10	2.0	67.8	27/45	57/64	99/99	94/84	74/80	69/64
	SHOT	15	2.0	68.7	26/33	40/56	99/100	96/92	88/85	74/71
	SH	7	1.5	65.0	22/34	56/73	99/99	91/85	79/74	40/60
FM3N	NH	15	2.0	55.9	0/0	1/20	100/98	97/90	82/80	67/62
	DH	5	0.5	73.2	33/50	70/66	99/99	97/90	84/86	53/74
	SI	10	2.0	74.9	34/55	64/72	99/100	98/92	88/89	74/73
	SHOT	10	1.0	55.2	46/42	43/50	99/91	98/85	12/59	49/72
	SH	5	1.0	73.0	43/53	61/71	100/100	96/90	85/83	57/78

largest improvement in classification performance is visible, if we use 10 instead of 5 or 5 instead of 3 bins. Further increasing the number of bins does not improve considerably the classification performance. The SHOT descriptor is less effected by the number of bins than the other descriptors and the distribution histogram benefits from an increasing number of bins the most.

Class-wise Performance. Tables 3.2, 3.3 and 3.4 show the precision and recall per class with the global reference frame. In this tables, we only show the best results of every descriptor and the class-wise precision and recall.

Table 3.4: *Precision and recall on the Wachtberg dataset.*

C	Feature	b	δ	F_1	vehicle	ground	facade	pole	vegetation
SR	NH	15	2.0	52.0	9/37	93/87	94/85	2/23	71/69
	DH	7	1.5	68.7	41/55	96/92	80/77	50/66	74/73
	SI	10	2.0	73.4	60/64	97/94	76/81	55/70	78/74
	SHOT	15	2.0	71.5	54/56	96/92	91/90	40/55	76/78
	SH	7	1.5	68.7	22/57	97/93	79/77	63/80	79/70
FM3N	NH	15	2.0	51.7	5/28	95/85	96/87	0/20	73/72
	DH	5	1.0	70.8	31/66	98/91	84/74	61/76	79/79
	SI	10	1.5	72.5	46/65	98/93	86/84	41/77	82/78
	SHOT	10	1.5	73.9	55/68	98/89	92/90	49/60	75/81
	SH	5	1.5	68.6	36/61	97/92	79/75	52/69	73/72

The class-wise precision and recall reveal the deficiency of the different descriptors. Generally, the classes *ground*, *facades*, and *vegetation* could be well distinguished from the other classes. These classes show consistent appearances in the different datasets and are therefore easily to distinguish from other classes even locally. Vegetation is mostly characterized by a scattered point distribution.

Classifying *poles*, *vehicles*, and *wire* is far more challenging as the results indicate. Poles and wires are sometimes only represented by a few laser range points and are therefore often confused with scatter from vegetation. Vehicles are the most complex objects in the datasets and show very different surfaces, which are locally indistinguishable from walls or sometimes vegetation. As discussed earlier, a larger support radius can include valuable contextual information, but can also lead to wrong classifications. Especially, in the Wachtberg dataset, vehicles were often confused with vegetation, since the front part of a car is similar to lower bushes and shrubs. In all datasets, windows in the facades and the induced sparsity of laser returns in these areas leads to a misclassification of building points as vegetation.

In summary, the softmax regression showed the best results with the SHOT, spin image and distribution histogram using the global reference frame and a support radius of 1.5 m and 2.0 m. Interestingly, the softmax regression is close to the performance of the more complex FM3N with large support radii regardless of the employed feature representation. This confirms the intuition formulated in the beginning of this chapter: more complex features can compensate a simple classification model. Thus, the advantage of collective classification approaches seems to appear only with small support radii.

Classifier performance. Figure 3.8 shows the best results of the local and the collective classification approach with the Wachtberg dataset. We see in general a more consistent

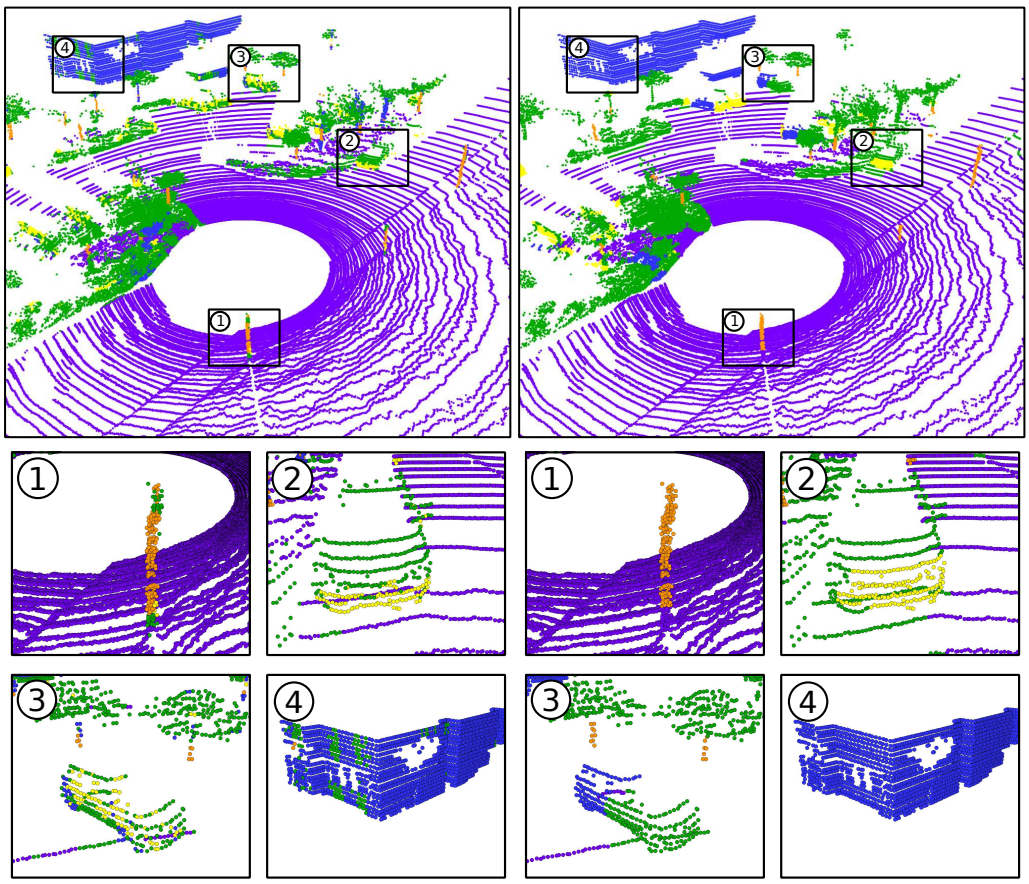


Figure 3.8: *Qualitative results of the evaluated classifiers with the Wachtberg dataset. The labels are colored as follows: purple = ground, blue = building facades, green = vegetation, orange = poles, yellow = vehicles. In subfigure (a) the softmax regression results are shown and subfigure (b) depicts the Functional Max-Margin Markov Networks results.*

classification result of the collective classification approach. The FM3N propagates label information through the graph and label assignments of a point can be outvoted by neighboring points. In contrast to the labeling of the collective approach, changes the label assignment in the local classifier regardless of the neighboring points and therefore a more diverse labeling is observable. However, we can also observe regions, where label propagation leads to wrong classifications, shown in the third example of Figure 3.8. A single confident label outvotes neighboring labels if these have a smaller value in the node potential.

The softmax regression is unable to separate the diverse appearances of a certain class from other classes. For example, the pole shows vegetation labels at the top and the bottom part

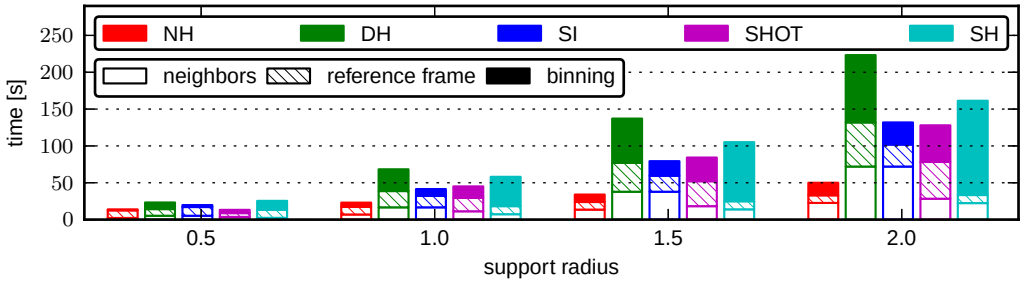


Figure 3.9: *Average time needed to calculate the evaluated descriptors for every laser point. We separated the neighbor search, reference frame calculation, and actual calculation of the bin indices for each descriptor. The increase in neighbor search time for the spin image and distribution histogram results from the increased search radius.*

(see Figure 3.8a, example 1). This is mainly caused by the very different feature representation of the middle part compared to the upper and lower part. We essentially can observe here three parts of the pole, which are differently encoded in the descriptors. In these cases, it is certainly not possible to achieve well localized clusters of feature vectors of a single class in the feature spaces, which would enable the linear classifier to find separating decision boundaries. The FM3N shows better and more consistent classification results in this case due to the flexibility in the decision boundaries.

Runtime performance. In Figure 3.9, we show the average computation time of the histogram descriptors measured on the Wachtberg dataset. In this diagram, we separated the different processing steps of the descriptor calculation, i.e., the nearest neighbor search, the calculation of the reference frame, and the actual calculation of the bin indices. Increasing the support radius increases the number of neighboring points and consequently the overall computation time. Note, in case of the neighbor search of the spin image and distribution histogram, we increase the search radius by $\sqrt{3}$ to get all neighbors inside a cube instead of a sphere, which directly affects the time needed to search for neighbors.

If we only account for the computation time, then the normal histogram and spin images are the most efficient descriptors. The distribution and spectral histogram are more complex to compute and show the worst efficiency. But if we also consider the quality of the classification results, then the spin image is the most efficient descriptor combining fast computation with high accurate classification results.

Generally, increasing the support radius leads to better performance of the local classifier, but this increase must be payed with more complex feature computations. However, the computation time of the descriptors of at least 3 seconds for the smallest normal histogram is still too high for real-time processing. But we are certain that the overall computation

time can be significantly reduced by a more careful implementation using concurrent computation of the descriptors and efficient sub-sampling strategies.

3.6 Summary

In this chapter, we evaluated several histogram-based descriptors for the classification of three-dimensional laser range data in urban environments. We used datasets acquired with different state-of-the-art sensor setups and showed that the choice of features dramatically influences the performance of all investigated classification approaches. From the presented results and the discussion in the last chapter, we can draw the following conclusions:

1. A proper choice of the reference frame significantly improves the performance of all evaluated classification approaches; the global reference frame was superior to the usual normal-based reference frame, since we get more stable reference frames and can furthermore distinguish very similar surface classes, e.g., ground and building facades.
2. The performance strongly correlates to the support radius; a larger support radius improved the descriptiveness significantly and usually leads to a better performance, since we implicitly encode contextual information, which leads to more discriminative features.
3. The number of bins showed no significant influence on the classification performance, but the influence increases at larger support radii, since a too fine quantization of the surrounding leads to instable feature representations.
4. Taking both the classification accuracy and the computation time in consideration, the spin image and the SHOT descriptor are the most effective and efficient descriptor.
5. The spectral histogram showed competitive performance in most cases, but is too costly to compute compared to the other approaches.

In the next chapters, we will use these insights for the development of new classification models for point-wise and segment-based classification. The choice of the reference frame is the most important finding: Independent of the classification approach, the global reference frame improves the descriptiveness of all descriptors significantly.

Future Work. One next step in the investigation of suitable features for three-dimensional classification is certainly the investigation of the combination of histogram descriptors. The general feature computation is currently simply not efficient enough for real-time computation in our implementation. An investigation of strategies to enhance the runtime performance is therefore mandatory to apply such point-wise classification in practice. We expect significant reductions in overall computation time by removing irrelevant points and the usage of concurrent computation, since each descriptor evaluation is independent of all other

descriptor computations. Combining both strategies, i.e., reducing the overall number of points and concurrent computation, should enable a near real-time point-wise classification in urban environments. The recent work of Pastuszka [2013] showed that the overall computation time can be significantly reduced by sub-sampling of the point cloud and an intelligent reduction of descriptor computations without sacrificing classification accuracy.

Next Chapter. Our analysis showed that the classification of most classes is plagued by a diverse local appearance of these objects. For example, the part at the top of a pole is different from the appearance at the bottom near the ground. Thus, it is understandable that a linear classifier, such as the softmax regression, faces considerable problems in finding hyperplanes, which separates both appearances together from all other classes. We showed that an increased support radius can alleviate these problems by increasing the similarity of both feature vectors, but this is dearly bought by computation time. In the next chapter, we propose an approach resolving appearance ambiguities differently and develop a novel model combining efficient softmax regression and ideas from nearest neighbor classification.

Chapter 4

Efficient hash-based Classification

Supervised classification using efficient linear models, such as softmax regression, is only possible if we encounter classes showing linearly separable clusters. Consequently, if feature vectors of a single class are scattered over the whole feature space we can not expect to learn a linear classifier enabling us to accurately predict the classes. In point-wise classification of three-dimensional laser range data, scattering of feature vectors is prevalent rather than exceptional. For instance, local appearance of complex objects, such as cars, varies drastically for different parts of the object. Every part induces a different cluster in feature space and might make it impossible to find linearly separating hyperplanes between different classes.

As discussed in the previous chapter, increasing the support radius helps to alleviate this problem by making feature vectors of different object parts more similar. Alternatively, projecting feature vectors into high-dimensional spaces can also lead to linearly separable classes — support vector machines, for example, use this (kernel) trick. Another option is to use more complex classification models that are able to learn non-linear decision boundaries. One of these alternative classification models is the collective classification that takes the vicinity of a laser point into account: intuitively, class labels should propagate smoothly among neighboring points. All solutions, however, come at the expense of higher computation time for learning and/or inference and the vast number of available laser range points might render such approaches impossible.

A recent development in the machine learning community has been the insight that massive datasets are not only challenging, but can also be seen as an opportunity [Torralba et al., 2008a]. Instead of developing more complex classification models, massive datasets allow to move in the opposite direction: How much can the data itself help us in solving the

problem? Halevy et al. [2009] describe this concept as exploiting "the unreasonable effectiveness of data." Massive datasets are likely to capture even rare aspects of the problem at hand. Does this also hold for our classification task? Can we learn the characteristics of objects from very dense scans without learning complex models?

Nearest neighbor classifiers exploit this data-centered view in its purest form: simply store all available data and use that data for prediction. Although conceptually simple, applying nearest neighbor classifiers on three-dimensional laser range data requires highly efficient ways of (1) storing millions of training examples in memory and (2) quickly finding neighbors at prediction time. Our main contribution is to address both issues by representing each feature vector by a compact binary code that is constructed so that similar feature vectors have similar binary codes. In turn, similar neighbors have codes within a small Hamming distance of the code. Then we learn a softmax regression model locally over all vectors with the same binary code word. More precisely, we use Weiss et al.'s spectral hashing to compute compact binary codes [Weiss et al., 2008]. Using codes learned by spectral hashing, retrieval can be very fast – millions of queries per second on off-the-shelf PCs. Our experiments show that the resulting approach, called spectrally hashed softmax regression (SHSR), can efficiently represent very different appearances of objects and improve the softmax regression results significantly without sacrificing computational efficiency. Spectrally hashed softmax regression works very well in our application: identifying cars, foliage, walls and load bearing areas in three-dimensional laser range data. To our best knowledge, we are the first to apply spectral hashing to a robotics task and combine this with a softmax regression.

This chapter is mainly based on our work published in Behley et al. [2010]. In contrast to this earlier work, we adapted the model to incorporate prediction of local models learned using softmax regression as introduced in Section 2.2.1, "Softmax Regression." We furthermore extended the experimental evaluation and evaluated our approach on the datasets presented in Chapter 3, "Histogram Descriptors for Laser-based Classification." These additional results strengthen the earlier findings on the superior performance of the proposed combination of similarity-preserving hashing and local classification models. We can show a more extended and differentiated view on the performance of the evaluated classification approaches.

The rest of the chapter is structured as follows. First, we discuss related work in Section 4.1, "Related Work." In Section 4.2, "Spectrally Hashed Softmax Regression," we introduce the proposed classification approach employing spectral hashing, which is also briefly introduced in this section. Section 4.3, "Experimental Evaluation," presents experimental results on the datasets presented in the previous chapter and finally Section 4.4, "Summary," concludes the chapter and outlines future work.

4.1 Related Work

The recent advent of fast three-dimensional laser sensors producing millions of laser range measurements in a fraction of a second lately attracted increasing interest in the robotics community. Particularly in outdoor applications, precise range measurements to objects in the vicinity are essential for a safe and collision-free navigation.

Mostly, approaches based on Conditional Random Fields (CRF) [Lafferty et al., 2001] have been used to classify three-dimensional point clouds. Anguelov et al. [2005] applied Associative Markov Networks [Taskar et al., 2004] for this purpose, and most of the following approaches were based on this collective classification approach. However, these techniques require quadratic and linear programming for learning and inference, respectively, which is almost intractable for massive datasets. Several methods have been proposed to speed up the overall processing, either by employing data reduction [Lim and Suter, 2007, Triebel et al., 2006] or by using more efficient learning and inference methods [Lu and Rasmussen, 2012, Munoz et al., 2009a, 2008, 2009b]. In the following, we will briefly mention the most recent approaches for supervised three-dimensional classification of laser range data and summarize their main ideas. We explicitly concentrate on approaches aiming at predicting the class of every point instead of using segmented point clouds [Himmelsbach et al., 2009, Spinello et al., 2011] or exploiting track information [Himmelsbach and Wuensche, 2012, Teichman et al., 2011, Teichman and Thrun, 2012].

Munoz et al. [2009b] showed how high-order interactions between cliques instead of pairwise couplings and already classified scans can be used for accurate on-board classification. Furthermore, they proposed to use functional gradient boosting [Ratliff et al., 2007] for learning node potentials as weighted sums of linear regressors instead of the usually used log-linear potentials [Munoz et al., 2009a]. Agrawal et al. [2009] augmented a CRF with object potentials generated by segmenting the scene into objects and calculating the covariances of the objects' laser points. Lai and Fox [2010] use a probabilistic exemplar-based approach leveraging three-dimensional models from the web, and applied domain adaption in order to remove artifacts not visible in real laser range data. Patterson et al. [2008] employed a nearest neighbor approach using spin images [Johnson and Hebert, 1999] and extended Gaussian images (EGI) [Horn, 1984]. First, a set of reference points is sampled from the labeled training scene, spin images are computed and stored in a database for later retrieval. Classifying unseen scans is achieved by calculating spin images of sampled points and matching these against the database containing also labels. Finally, the clusters of labeled hypotheses are verified using the EGIs. Xiong et al. [2011] use stacked softmax regressions to successively improve the classification results by incorporating contextual information. An initial point-wise classification and its labeling is used to learn contextual models that encode spatial relations such as "tree foliage is above a trunk" and vice versa. Later, these relations are used to incrementally improve the classification results us-

ing stacking [Wolpert, 1992]—classification results of earlier stages are features of later stages. Lu and Rasmussen [2012] smooth classification results of a local classifier using distance-based potentials in a CRF.

In summary, a lot of effort has been put into the development of more complex classification models and most of the presented approaches need a lot of processing power for inference. As we pointed out in the motivation, we move in the opposite direction, inspired by the work of Torralba et al. [2008b], who employed the power of a vast number of images to label arbitrary scenes according to a very large database of images from the well-known LabelMe [Russell et al., 2008] dataset. In line with their work, we use distance-preserving hashing to enable a fast retrieval of approximate nearest neighbors. However, we additionally use local classification models to avoid the linear search of the best matching neighbor to deduce a label.

Finally, we have to mention the well-known locally weighted learning of Atkeson et al. [1997], which fits local models using k nearest neighbors from the training data for each query. Our aim is to avoid the need for exact calculation of k nearest neighbors, since this is intractable for massive datasets.

4.2 Spectrally Hashed Softmax Regression

In our application, we have a huge amount of laser range points and we can undoubtedly assume that the induced decision boundaries between classes are non-linear. To this end, nearest neighbor classifiers are an elegant and flexible tool for classification in such a regime, as introduced in Section 2.2.2, “k-Nearest Neighbor Classification.” However, as we must repeatedly find nearest neighbors for every prediction, we need fast nearest neighbor retrieval techniques.

similarity-preserving hashing

Recently, *similarity-preserving hashing* for fast approximate nearest neighbor search has received considerable interest by researchers within the machine learning [Gong et al., 2012, Li et al., 2011, Salakhutdinov and Hinton, 2009, Weiss et al., 2008], and also computer vision communities [Gong and Lazebnik, 2011, He et al., 2013, Kulis and Grauman, 2012]. Traditional hashing methods try to embed vectors such that collisions, i.e., different elements getting the same hash value, are avoided. Similarity-preserving hashing, however, learn codes, which result in collisions, if the original vectors are similar in respect to some similarity measure. In our case, the similarity is expressed as euclidean distance: the smaller the distance between the vectors, the more similar these vectors are.

Similarity-preserving hashing methods learn a mapping from the high-dimensional input data to a low-dimensional Hamming, i.e., binary, space. Note that the fact that the embeddings are binary is critical to ensure fast retrieval times, which enables the use of hardware-

based intrinsic binary comparisons. As Weiss et al. [2008] report, this kind of retrieval can be very fast; millions of queries per second on standard computers. The Hamming distance between two binary codewords can be computed via an XOR operation and a bit count. Moreover, if the input dimension is very high, hashing methods lead to enormous memory savings as few bits are often already sufficient to compactly encode the whole dataset. This beneficial property lead also to increasing interest in the computer vision community for fast retrieval of similar images from massive image collections [Kulis and Grauman, 2012, Torralba et al., 2008b]. In that application, every image is encoded using only a few bits and the whole collection can be queried using the binary codeword of a query image. The retrieved images are then simply ranked according to their hamming distances to the query. Li et al. [2011] show that hashing can also be directly used for learning of classifiers on large-scale datasets, if feature vectors are binary codes.

Hashing naturally leads to the following point-wise classification approach:

1. **(Hashing)** Learn a similarity-preserving hash function h resulting in compact binary codes for a given set of N scans.
2. **(Local Classification)** Learn a local classification model $P(y|\boldsymbol{x}, h(\boldsymbol{x}))$ on all scan points $\boldsymbol{x}$ sharing the same binary code $h(\boldsymbol{x})$.
3. **(Prediction)** For classifying a new scan point $\boldsymbol{x}$, compute the binary code of $\boldsymbol{x}$, look-up the associated local model $P(y|\boldsymbol{x}, h(\boldsymbol{x}))$ and use it to assign a class label y.

Indeed, this non-parametric large-scale classification approach is a special case of locally weighted regression [Atkeson et al., 1997], since we perform classification around a point of interest using all training scans that have identical binary codes. As we argue in the next section, if the lookup of the code for a new scan is efficient, this can yield very fast classification performance. Furthermore, as our experimental evaluation will show, this approximation works surprisingly well in our classification setting — outperforming nearest neighbor and softmax regression.

4.2.1 Spectral Hashing

We use spectral hashing from Weiss et al. [2008] to compute compact binary codes. The main benefit of spectral hashing is that the partitioning of the feature space can be computed in linear time. Recent studies show that spectral hashing is competitive to other more complex approaches, if the desired output dimension of the binary codes is small.

Following the original derivation of Weiss et al. [2008], spectral hashing works as follows. To preserve similarities, one is interested in a hash function that maps nearby input vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ to binary hash codes with a small Hamming distance. Thus, the objective for a

hash function $h : \mathbb{R}^n \mapsto \{0,1\}^k$, which helps us to search efficiently $\boldsymbol{x}_i \in \mathbb{R}^n$ in a large dataset that is distributed according to a distribution $P(\boldsymbol{x})$, can be formulated as follows:

$$\text{minimize} \int K(\boldsymbol{x}_i, \boldsymbol{x}_j) \cdot \left\| h(\boldsymbol{x}_i) - h(\boldsymbol{x}_j) \right\|_H^2 \cdot P(\boldsymbol{x}_i) \cdot P(\boldsymbol{x}_j)\, d\boldsymbol{x}_i\, d\boldsymbol{x}_j \tag{4.1}$$

$$\text{subject to} \int h(\boldsymbol{x}) P(\boldsymbol{x})\, d\boldsymbol{x} = 0 \tag{4.2}$$

$$\int h(\boldsymbol{x}) h(\boldsymbol{x})^T P(\boldsymbol{x})\, d\boldsymbol{x} = \mathbf{Id} \tag{4.3}$$

Here, the function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ defines the similarity between different data points. A natural choice is the Gaussian kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2 / \epsilon^2)$, i.e., vectors with a small euclidean distance are assigned values near 1 and the value quickly flattens as we increase the distance. The two constraints encode the requirements that the different bits of codewords should be independent (Equation 4.2) and uncorrelated (Equation 4.3). As Weiss et al. [2008] show, finding such codes is NP hard, but the problem can be solved in polynomial time by relaxing the constraint that the codewords need to be binary, $h(\boldsymbol{x}) \in \{0,1\}^k$. Indeed, it has been shown that the solution is given by an eigenfunction $\Phi(x)$. If $P(\boldsymbol{x})$ is separable, i.e. $P(\boldsymbol{x}) = \prod_j P_j(\boldsymbol{x}^{(j)})$, and the similarity is defined by the Gaussian kernel then the solution $\Phi(x)$ is given by the product of the one-dimensional eigenfunctions

$$\Phi_1\left(\boldsymbol{x}^{(1)}\right) \Phi_2\left(\boldsymbol{x}^{(2)}\right) \cdots \Phi_n\left(\boldsymbol{x}^{(n)}\right) \tag{4.4}$$

and eigenvalue $\lambda_1 \cdot \lambda_2 \cdots \lambda_n$. Especially, if $P_j(x)$ is a uniform distribution on the interval $[a, b]$, the eigenfunctions $\Phi_j(x)$ are given by

$$\Phi_j(x) = \sin\left(\frac{\pi}{2} + \frac{j\pi}{b-a} x\right) \tag{4.5}$$

$$\lambda_j = 1 - \exp\left(-\frac{\epsilon^2}{2} \left|\frac{j\pi}{b-a}\right|^2\right). \tag{4.6}$$

Assuming that the data is uniformly distributed, we can now calculate the eigenfunctions and threshold the values at 0 to obtain a codeword. This results in the following algorithm to determine a hash function h for data points $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^n\}$:

Spectral Hashing

1. Calculate the k principle components using eigenvalue decomposition of the covariance matrix $\mathbf{C}$. Rotate vectors $\boldsymbol{x}_i$ according to the k largest eigenvectors, resulting in $\tilde{\boldsymbol{x}}_i^{(j)}, 0 \leq j < k$.
2. Determine for every dimension $a^{(j)} = \min_j\left(\tilde{\boldsymbol{x}}_i^{(j)}\right)$ and $b^{(j)} = \max_j\left(\tilde{\boldsymbol{x}}_i^{(j)}\right)$ and compute the eigenvalues according to (4.6).
3. Threshold the k eigenfunctions $\Phi_k(x)$ with smallest eigenvalue at 0 to obtain the hash code.

As empirically validated by Weiss et al. [2008], the algorithm is not restricted to uniformly distributed data, and can generate hash codes that are capable of finding a good partition of the data, which allows to efficiently search for nearest neighbors. In the next section, we show how the feature space is partitioned using spectral hashing. We show, furthermore, that the hash function can be learned efficiently, since we do not need to handle every data point explicitly: computing the covariance is sufficient. The computation of the covariance can be done incrementally (see Equation 2.2 in Section 2.1.3), and we can therefore even handle datasets that do not fit into memory. In turn, we only have to determine the minimum and maximum of the rotated feature vectors to get a partition of the feature space.

4.2.2 Combining Spectral Hashing and Softmax Regression

The main idea underlying locally weighted learning is to use local models learned from k neighboring points of a query point. Learning models for classification at prediction time is known as lazy classification and with this paradigm it is also possible to approximate non-linear target functions. However, determining k nearest neighbors for each prediction is inefficient for large training sets and the advantage of local prediction turns into a disadvantage in terms of computational complexity.

To overcome this, we partition the feature space using the hash function h and learn local models directly from the training data, and finally store local models for every partition induced by the hash function h, when necessary. In particular, we first determine for each example $(\boldsymbol{x}_i, y_i) \in \mathcal{X}$ of the training set $\mathcal{X}$ the bin $c = h(\boldsymbol{x}_i)$ in a hash table $\mathcal{H}$. For each occupied entry c of the hash table, we determine the classes C^c inside the bin $\mathcal{H}_c$ and learn a local softmax regression model on the subset $\{(\boldsymbol{x}, y) | h(\boldsymbol{x}) = c\}$, if $|C^c| > 1$. If only feature vectors of a single class are hashed to a codeword, we skip the learning of a local classification model and simply store the class label in C^c. The learning of the spectrally hashed softmax regression (SHSR) is summarized in Algorithm 1 on the next page. Note that the proposed method is not restricted to softmax regression, so that we could even use non-linear classifiers for local classification within each partition defined by the hashing function.

SHSR learning

To determine the label distribution $P(\hat{y}|\hat{\boldsymbol{x}})$ of an unseen feature vector $\hat{\boldsymbol{x}}$, we have to distinguish several cases. Let $\hat{c} = h(\hat{\boldsymbol{x}})$ be the codeword of feature vector $\hat{\boldsymbol{x}}$.

SHSR inference

1. If $\left|C^{\hat{c}}\right| > 1$, we simply return the label distribution $P(\hat{y}|\hat{\boldsymbol{x}}, \hat{c})$ of the previously learned local classification model. Note, that we assume $P(\hat{y} = j|\hat{\boldsymbol{x}}, \hat{c}) = 0$, if $j \notin C^{\hat{c}}$, since we have not encountered any training example with such a label.
2. If $\left|C^{\hat{c}}\right| = 1$, we set $P(\hat{y} = j|\hat{\boldsymbol{x}}, \hat{c}) = 1$, $j \in C^{\hat{c}}$ and $P(y = k|\hat{\boldsymbol{x}}, \hat{c}) = 0, k \notin C^{\hat{c}}$ otherwise.

Algorithm 1: Learning spectrally hashed softmax regression

Data: training set $\mathcal{X} = \{(\boldsymbol{x}_i, y_i)\}$
with features $\boldsymbol{x}_i \in \mathbb{R}^D$ and labels $y_i \in \mathcal{Y}, |\mathcal{Y}| = K$
Result: hash function h,
softmax regressions $P(y|\boldsymbol{x}, c)$ with parameters $\boldsymbol{\theta}^c = \{(\boldsymbol{\theta}^c_1, \ldots, \boldsymbol{\theta}^c_{|C^c|})\}$,
lookup table of classes C^c per codeword c

learn hashing function h (cf. Section 4.2.1)
`/* build hash table */`
foreach $(\boldsymbol{x}_i, y_i) \in \mathcal{X}$ **do**
  $c = h(\boldsymbol{x}_i)$
  $\mathcal{H}^c = \mathcal{H}^c \cup \{(\boldsymbol{x}_i, y_i)\}$
end
`/* learn local softmax regressions */`
foreach $c \in \{0, \ldots, 2^{k-1}\}$ **do**
  $C^c = \{y|(\boldsymbol{x}, y) \in \mathcal{H}^c\}$
  if $|C^c| > 1$ **then**
    minimize Equation 2.25 on $\mathcal{H}^c$ from Section 2.2.1
  end
end

3. If $\left|C^{\hat{c}}\right| = 0$, we have no model associated with the codeword. And therefore set to a uniform distribution, $P(\hat{y}|\hat{\boldsymbol{x}}) = |\mathcal{Y}|^{-1}$. We increase the search radius and determine the contribution from neighboring hashes with increased hamming distance to $h(\hat{\boldsymbol{x}})$.

More precisely, we first use models with radius 0, i.e., $\|h(\hat{\boldsymbol{x}}) - h(\boldsymbol{x}_n)\| = 0$. If we are unable to retrieve such model, we continue with neighboring partitions for hashes $h(\boldsymbol{x}_n)$, where $\|h(\hat{\boldsymbol{x}} - h(\boldsymbol{x}_n)\| = 1$. We continue increasing the search radius until we find a neighboring partition that contains a model.

The final label distribution $P(\hat{y}|\hat{\boldsymbol{x}})$ is then the mean over all neighboring codewords $\mathcal{N}$:

$$P(\hat{y}|\hat{\boldsymbol{x}}) = \frac{1}{|\mathcal{N}|} \sum_{c \in \mathcal{N}} P(\hat{y}|\hat{\boldsymbol{x}}, c) \tag{4.7}$$

Note that if we already encounter a model for $h(\hat{\boldsymbol{x}})$, we simply have $P(\hat{y}|\hat{\boldsymbol{x}}) = P(\hat{y}|\hat{\boldsymbol{x}}, h(\hat{\boldsymbol{x}}))$. Since the hashing function is similarity-preserving, using models of codewords with increasing hamming distance lead to predictions in the sense of locally weighted learning. However, since inference in SHSR is a simple lookup of local classification models in a table, we can determine the prediction with only a little overhead compared to traditional nearest neighbor models.

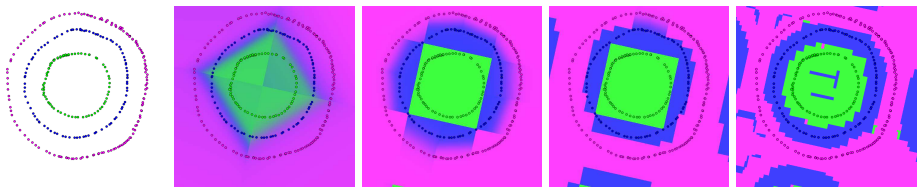


Figure 4.1: *Some examples of the partition of a highly non-linear feature space (a) using the proposed spectrally hashed softmax regression. Subfigure (b) is generated using* 2 *bits, (c) uses* 4 *bits, (d) was trained with* 8 *bits and in (e) we used* 16 *bits for the hashing of the feature space. The repetition in class assignments is caused by the sinusoid in the eigenfunction.*

Figure 4.1 visualizes some examples of spectrally hashed softmax regression for different numbers of bits k. With increasing codeword size, the partitioning increases and also the decision boundaries of the local softmax regressions adapt to the non-linear feature space, as we have argued in the beginning of this section. Furthermore, smaller partitions lead to less data points inside a partition, thus the learning of the softmax regression can be performed more efficiently due to the reduced size of the training set. But also a negative side effect is visible: as the number of possible bits increases, it gets more likely to perform overfitting, as we show in the next section.

4.3 Experimental Evaluation

In this section, we present results on the classification performance and the efficiency of our combination of spectral hashing and softmax regression. We performed extensive experiments on several datasets to show different properties of our proposed approach. In a first set of experiments, we present results on three different datasets and use the same experimental setup as presented in Section 3.4, “Experimental Setup.” After discussion of these results, we show the efficiency of the inference in comparison to other local classification approaches — nearest neighbor classification, spectrally hashed nearest neighbor classification, and softmax regression. We furthermore evaluate the influence of the parameters on the classification performance — number of bits used in the spectral hashing and the maximal search radius.

All classifiers were implemented in C++ and the experiments were performed on an Intel Xeon X5550 with 2.67 GHz using a single core and 12 GB memory. If not stated differently, we used codewords of 8 bits and maximal search radius of hamming distance 4 for the spectrally hashed softmax regression. The local softmax regressions used a fixed regularization of $\lambda = 0.01$ and the intercept was fixed to 1.0. The parameters of the single softmax regression were also fixed to $\lambda = 0.01$ and intercept 1.0.

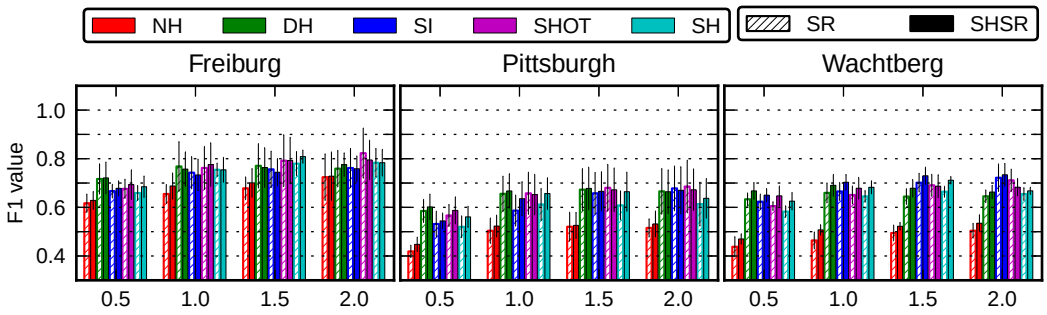


Figure 4.2: *Comparison between Softmax Regression (SR) and Spectrally Hashed Softmax Regression (SHSR) using* 8 *bits.*

Classification results. In our earlier study [Behley et al., 2010], we compared a binary logistic regression [Komarek and Moore, 2005] and a combination of spectral hashing and locally learned logistic regressions, called spectrally hashed logistic regression, using a spin image with global reference frame. The earlier results evaluated on a dataset similar to the Wachtberg dataset presented in Section 3.4 indicated a clear advantage of the spectrally hashed logistic regression over the logistic regression in that setting. The extended evaluation using different descriptors and support radii in this chapter reveals a more differentiated view on the specific performance in different conditions.

In this chapter, we used softmax regression, which directly learns a multi-class classifier from a dataset with multiple classes. Another well-established method for learning a multi-class classifier is to learn K separate binary classifiers $P_k(y|\boldsymbol{x})$ with $y = \{1, 0\}$ for each of the K classes, where all other $K-1$ classes are combined into a single auxiliary class $y = 0$; we used this so-called 1-vs-K strategy in [Behley et al., 2010] and a final predictive distribution $P(y|\boldsymbol{x})$ over all K classes is calculated by normalizing the vector of $P_k(y = 1|\boldsymbol{x})$ predictions. Both learning paradigms are capable of learning an effective multi-class classification approach as shown by Rifkin and Klautau [2004] or Daniely et al. [2012]. However, as argued by Bishop [2006], the learning of K separate classifiers may lead to ambiguous decision regions, where all binary classifiers predict $P_k(y = 1|\boldsymbol{x}) < 0.5$, i.e., every classifier is confident that the feature vector $\boldsymbol{x}$ does not belong to its class k. In a real multi-class classification approach, e.g., softmax regression, the decision boundaries between classes are influenced by every class and consequently, such situations are impossible. In consequence, this behavior also influences the performance advantage in favor of the spectrally hashed logistic regression compared to the logistic regression, since a partitioning of the feature space in subspaces naturally reduces the occurrence of ambiguous regions. Therefore, the results in this section do not always show the clear advantage of a partitioning of the feature space as stated in our previous work [Behley et al., 2010].

Figure 4.2 show the results of the SHSR using a global reference frame in comparison to

Table 4.1: *Precision and recall on the Freiburg dataset.*

C	Feature	b	δ	F_1	ground	facade	pole	vegetation
SR	NH	15	2.0	70.3	98/95	94/88	12/24	82/86
	DH	7	1.5	77.7	99/99	90/82	57/56	77/79
	SI	10	2.0	76.2	99/99	84/81	55/62	79/74
	SHOT	10	2.0	82.3	99/99	93/91	60/63	85/84
	SH	7	2.0	80.8	98/98	93/87	51/66	84/83
SHSR	NH	10	2.0	72.8	98/96	93/90	20/27	81/85
	DH	7	2.0	77.8	99/99	89/84	47/61	84/77
	SI	5	2.0	77.2	99/98	88/83	49/65	78/75
	SHOT	15	2.0	79.9	99/98	89/92	46/62	87/80
	SH	5	1.5	80.9	99/98	92/89	48/65	86/82

softmax regression using different support radii. In these results, we can see larger improvements especially in the Wachtberg dataset, but only slight improvements in the other datasets with small support radius. These improvements are mainly caused by the ability of the SHSR to represent different appearances of different classes correctly.

In line with the results of Chapter 3, "Histogram Descriptors for Laser-based Classification," the global reference frame significantly improves the performance of the SHSR. Moreover, we can observe an overall improvement of the classification results compared to the softmax regressioneven even with the local reference frame (cf. Appendix B). This observation can also be explained by the increase in model capacity of the SHSR. Due to the local models of the SHSR, we can expect that the model can disambiguate different classes even if the reference frame is unstable. The local models adapt to this situation and still allow an improved classification performance of the overall classification approach.

Tables 4.1, 4.2, and 4.3 show the classification results for each class in terms of precision and recall. Both classifiers show a similar best performance for each class, where the classes ground, vegetation, and facade are consistently better classified than the other classes. As pointed out earlier, the other classes show a more diverse appearance and vegetation shows a more scatter-like appearance, which is often assigned to sparse points at boundaries of partial occlusions. The presented class-wise results do not reveal a clear advantage of one or the other approach, but a visual inspection shows a more consistent labeling of the SHSR compared to the results of the softmax regression.

The comparison of the visual results in Figure 4.3 shows the improvement of the classification results for different classes in the Wachtberg dataset. Compared to the softmax regression, our proposed approach is able to capture the multiple appearances of the pole and consistently assigns the correct label. As also shown in the close-ups, there is a large

Table 4.2: *Precision and recall on the Pittsburgh dataset.*

C	Feature	b	δ	F_1	wire	pole	ground	vegetation	facade	vehicle
SR	NH	5	1.5	53.3	18/43	1/25	99/98	92/90	89/72	25/41
	DH	7	2.0	69.2	24/42	55/59	99/99	94/87	79/81	65/69
	SI	10	2.0	67.8	27/45	57/64	99/99	94/84	74/80	69/64
	SHOT	15	2.0	68.7	26/33	40/56	99/100	96/92	88/85	74/71
	SH	7	1.5	65.0	22/34	56/73	99/99	91/85	79/74	40/60
SHSR	NH	5	2.0	54.4	26/33	5/29	99/98	96/92	89/80	26/37
	DH	5	1.5	67.8	30/34	60/59	99/99	93/86	77/77	65/61
	SI	10	2.0	67.0	28/28	58/62	99/99	90/84	76/72	72/68
	SHOT	15	2.0	67.4	21/26	38/51	99/99	96/91	86/86	73/71
	SH	5	1.0	65.7	40/37	52/62	99/99	92/85	80/75	43/57

Table 4.3: *Precision and recall on the Wachtberg dataset.*

C	Feature	b	δ	F_1	vehicle	ground	facade	pole	vegetation
SR	NH	15	2.0	52.0	9/37	93/87	94/85	2/23	71/69
	DH	7	1.5	68.7	41/55	96/92	80/77	50/66	74/73
	SI	20	2.0	73.4	60/64	97/94	76/81	55/70	78/74
	SHOT	15	2.0	71.5	54/56	96/92	91/90	40/55	76/78
	SH	7	1.5	68.7	22/57	97/93	79/77	63/80	79/70
SHSR	NH	15	2.0	56.2	24/32	91/88	91/91	6/15	71/69
	DH	7	1.0	69.7	49/54	93/92	80/76	54/59	77/72
	SI	10	2.0	73.4	67/67	96/94	77/75	56/64	76/75
	SHOT	15	1.5	69.6	54/60	93/93	90/89	35/46	80/73
	SH	7	1.5	71.2	52/60	96/94	79/78	52/61	77/74

improvement in the classification of the car. Here the softmax regression, as well as the Functional Max-Margin Networks (see Figure 3.8), assigns the label vegetation to a large upper part of the car, but the SHSR manages to assign the correct label here. However, as noted in Chapter 3, “Histogram Descriptors for Laser-based Classification,” these results also reveal the shortcomings of a local approach compared to collective approaches. We still see single points in vicinity to correctly classified points, which are assigned to the wrong class. The borders of scan shadows on buildings are still incorrectly labeled as vegetation and the other example for a car still shows a mix of vehicle and building classifications.

In summary, the presented results confirm our hypothesis that the increased capacity of the proposed SHSR improves the classification results of a local classification approach. The

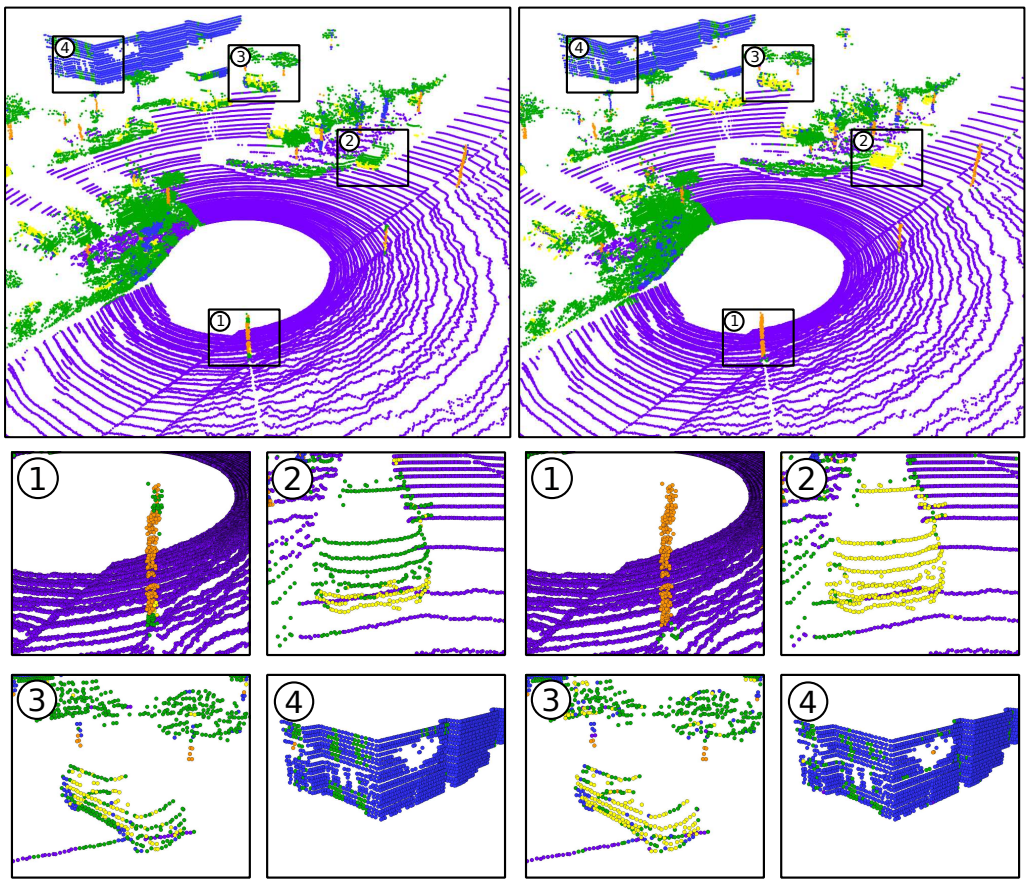


Figure 4.3: *Qualitative results on the Wachtberg dataset. The labels are colored as follows: purple = ground, blue = building facades, green = vegetation, orange = poles, yellow = vehicles. In subfigure (a) the softmax regression results are shown and subfigure (b) depicts the spectrally hashed softmax regression results.*

visual results on the Wachtberg dataset clearly show a more consistent labeling and that different appearances are correctly handled by the spectrally hashed softmax regression.

Runtime Performance. For comparison to established approaches, we calculated spin images using a global reference frame using 10 bins and a support radius of 2.0 m. In the following, we report the average time needed to determine the labeling of all folds of the Wachtberg dataset. These timings do not include the feature calculation time nor the time needed to estimate the normals.

We implemented a standard nearest neighbor classifier using k-d trees [Arya et al., 1998]

classification approach	F_1	avg. inference [s]
Nearest Neighbor ($k = 1$)	69.32	1,805.59
Nearest Neighbor ($k = 3$)	69.54	2,026.20
Nearest Neighbor ($k = 5$)	69.62	2,168.33
Spectrally Hashed Nearest Neighbor (4 bits)	68.90	3,580.19
Spectrally Hashed Nearest Neighbor (8 bits)	68.25	916.98
Spectrally Hashed Nearest Neighbor (16 bits)	65.76	178.64
Spectrally Hashed Softmax Regression (4 bits)	**74.23**	0.45
Spectrally Hashed Softmax Regression (8 bits)	73.47	1.01
Spectrally Hashed Softmax Regression (16 bits)	67.53	3.16
Softmax Regression	72.25	**0.11**
Functional Max-Margin Markov Networks	72.26	83.86

Table 4.4: *Inference time of different classification approaches.*

to show in principle the capability of a nearest neighbor classifier. We additionally implemented a spectrally hashed nearest neighbor classifier, where we exploited the spectral hashing to speed-up the search for nearest neighbors. We used a look-up table for an efficient query of nearest neighbors and store for every binary codeword the feature vectors with that binary codeword from the training data. The spectrally hashed nearest neighbor uses the exact same inference procedure of the spectrally hashed softmax regression using neighboring codewords, as described in Section 4.2.2. But instead of applying a local classification model, we simply compare the stored feature vectors with the feature vector of the query laser range point. We finally compare the inference time of all approaches with the softmax regression and Functional Max-Margin Markov Networks [Munoz et al., 2009a], which were already discussed in Chapter 3, "Histogram Descriptors for Laser-based Classification."

Table 4.4 summarizes the average time needed to estimate a label for every point and the averaged F_1 rate overall folds of the Wachtberg dataset. On average 120,000 laser range points per scan must be classified and the average training set contains approximately 450,000 feature vectors, where ground and vegetation are responsible for the most laser returns.

In line with our argumentation in the beginning of the chapter, the nearest neighbor classifier shows a competitive classification performance compared to the other more complex approaches. However, the average time needed to classify a complete scan is clearly to high — at least 30 minutes for the classification of a complete scan. Increasing the number of considered neighbors k does not improve the classification rate significantly, which indicates that the nearest neighbor classifier is not strongly affected by overfitting. Nevertheless, we did not exploit the similarity of a large part of the feature space and kept very similar

feature vectors to reduce the overall number of training examples.

In comparison to the *k*-d tree based nearest neighbor search, the search exploiting Spectral Hashing with large binary codewords is significantly faster. Especially with increasing number of bits for learning the hashes, the simple linear search in feature vectors sharing the same binary codeword turned out to be up to 10 times faster (178.64 s vs. 1805.59 s) than the *k*-d tree. This is remarkable, as we can only see a small drop in the overall performance of the spectrally hashed nearest neighbor approach compared to the real nearest neighbor classifier (65.76 compared to 69.32). However, both approaches are by far not applicable in real world applications; waiting for 3 minutes for a classification result is not acceptable for an autonomous system operating in dynamic environments.

Using our proposed approach of learning local classification models on subsets of the training set sharing the same binary codeword significantly reduces the inference time (0.45 s vs. 178.64 s) and improves the classification rate (74.23 vs. 65.76) compared to the spectrally hashed nearest neighbor. Learning a local classifier for a hash bin reduces the influence of outliers and therefore improves the overall classification performance. As we simply have to exponentiate and multiply by the learned weight vector, we additionally save the time consuming search for nearest neighbors. Thus, the (offline) learning gets more complicated, but the (online) inference time is significantly reduced compared to the plain nearest neighbor approach. As the likelihood of finding a collision reduces with increasing number of bits, we can see an increase in the inference time with increasing number of bits (0.45 s with 4 bits vs. 3.16 s with 16 bits). We observed in our experiments with spectral hashing that the occupancy of the hash bins reduces significantly with increasing number of bits. Hence, as the number of available codes increases, we have to increase the search radius in terms of hamming distance more often and this consequently increases the inference time in our unoptimized implementation.

The softmax regression is the most efficient classification approach in comparison to the other the approaches, as expected. But we have to note that the additional hashing with 4 bit hash codes increases the time for inference by only 0.34 s and still allows near real-time application with update rates of 3 Hz, if we neglect the time needed for feature computation. As argued in Chapter 3, “Histogram Descriptors for Laser-based Classification,”, the Functional Max-Margin Networks visually show the most consistent classification results, but also an increased inference time. In this computation time the construction of the 5-nearest neighbor graph[1], and the computation of the edge features is included. However, even if we neglect this additional overhead, i.e., the graph construction and the calculation of the edge potentials, compared to the local classification approaches, the inference in the graph still needs 16.30 s.

[1] Note, the graph is constructed using *k*-nearest neighbor search in $\mathbb{R}^3$, which is significantly faster than searching neighbors in high-dimensional feature spaces like our classification problem, i.e., $\mathbb{R}^{100}$.

In conclusion, our proposed approach achieves the best classification performance of all local classification approaches and still shows a very efficient inference with only a small overhead compared to softmax regression.

4.4 Summary

In this chapter, we presented a simple and efficient algorithm for classifying three-dimensional laser range scan points of rigid objects, the spectrally hashed softmax regression. The combination of spectral hashing and local softmax regressions learned on subsets sharing the same binary codeword enabled us to learn an efficient classifier outperforming other local classification approaches. As shown in the extensive experimental evaluation, our new approach allows us to use a smaller support radius for the descriptors than the softmax regression and achieves superior classification accuracy. Despite these gains in classification accuracy, we showed that our approach needs only little more time to classify a laser range scan, which still enables very efficient inference. In summary, the visual inspection of the classification results revealed that the presented classification approach leads to more consistent label assignments, because of an increase in model capacity, which enables us to learn multiple appearances of complex object classes, such as cars, poles, and vegetation.

Future Work. There are several interesting extensions of the proposed approach. Recently, Xu et al. [2012] proposed an approach to learn a distance-preserving hashing function for proportional data, which is exactly the kind of representation used by the histogram-based descriptors. In our current approach, we use binary codewords preserving euclidean distances of the original vectors, but these distances are only an approximation of histogram distances, i.e., vectors close to each other in euclidean space might have large distance on the simplex. Hence, an investigation of other hashing algorithms could lead to more consistent partitions for learning local models and therefore further improve the classification accuracy. In our current implementation, we learn the mapping to binary codewords and the classification separately, but research on combination of classification models indicate that jointly learning the classification and the weighting of the classifiers could lead to substantial performance gains. Therefore, a possible enhancement of the presented approach might be achieved by learning a hash function enabling the local classifiers to discriminate different object classes.

Next Chapter. In the next chapter, we aim at detecting object classes relevant for autonomous driving and we only need to consider objects that reside on the ground. The overall setting is different, since we are interested in extracting objects with distinct boundaries instead of getting a rather general classification of all surfaces visible in the vicinity

of the robot. The presented point-wise classification of this chapter could be used in an exploration-like application, where we are interested in a general description of the environment.

In the next chapter, we are particularly interested in object classes that are very similar locally, such as pedestrians and bicyclists. We use a segment-based classification instead of a point-wise classification, since it allows us to encode the overall appearance of the whole segment instead of relying on only local cues. The next chapter still uses the already introduced local descriptors, but combines these into an object-level description of a segment. In addition, we show how to jointly learn multiple local softmax regressions and the weighting of these local models. This approach is similar to the approach presented in this chapter – we also partition the classification problem into different parts and learn local classification models. But instead of averaging each classifier contribution using the same weight factor, we jointly learn a weighting function.

Chapter 5

Segment-based Classification

In the previous chapters, we investigated and proposed methods for point-wise classification of three-dimensional laser range data. We showed that two main ingredients are essential for an effective classification approach using sparse laser range measurements: (1) large descriptors using a global reference frame and (2) local classifiers enabling the learning of multiple appearances of the object classes. Combining both approaches yields an effective and efficient approach to point-wise classification, which is superior to other local classifiers and on par with more complex collective classification.

In this chapter, we concentrate on the classification of objects that are relevant for autonomous driving — cars, pedestrians, and bicyclists. For this objective, the main challenge lies in the local appearance similarity of pedestrians and bicyclists, as shown in Figure 5.1, that motivates the segment-based approach of this chapter as follows: A bicyclist and a pedestrian can only be distinguished if we additionally take the bicycle into account: co-occurrence of upper body and wheels of a bicycle constitutes the semantic class bicyclist. If we only use local appearance, we must use a very large support radius to get a feature representation encoding the difference between a pedestrian and a bicyclist. Consequently, we always need the complete object to distinguish both classes. Thus, we propose to directly use segments, parts of the point cloud corresponding to single objects, which are extracted by an efficient segmentation approach. For classification, the segments are encoded using a bag-of-words representation with point-wise local features using a global reference frame.

In image-based object recognition, bag-of-word approaches [van der Sande et al., 2011] are a well-established concept, but rarely applied in laser-based perception. This is remarkable, since they offer several desirable properties by design, which are advantageous particularly in laser-based object recognition: (1) bag-of-words are robust to partial occlusions,

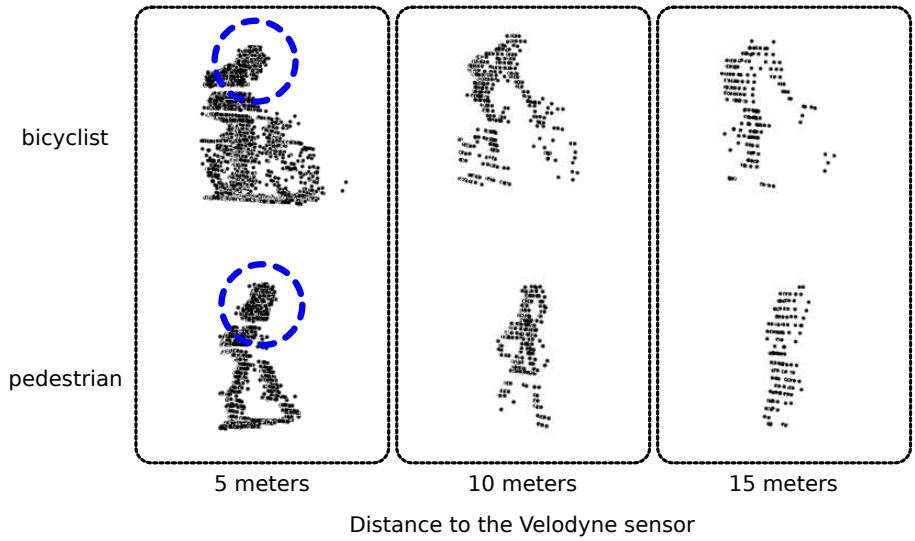


Figure 5.1: *The image shows the distance-dependent sparseness of point clouds generated by the Velodyne HDL-64E laser rangefinder of the exact same bicyclist and pedestrian at different ranges. At a distance of* 5 *meters, we can observe fine details of the bicyclist and the pedestrian, which are lost due the sparse sampling with range measurements at large distances. Highlighted (blue/dashed circle) is the apparent local similarity between bicyclists and pedestrians, which motivates the usage of segments-based instead of point-wise classification.*

(2) the entries for a certain class should still be visible in a part of the bag-of-words, even if we encounter an under-segmentation, (3) point-wise descriptors can be computed independently, which makes a concurrent evaluation possible. Thus, bag-of-words extracted from laser range data are a fundamental building block of the approach presented in this chapter. We can profit from our earlier findings and use the previously introduced histogram descriptors as words. Our experimental results indicate that the insights of Chapter 3, "Histogram Descriptors for Laser-based Classification," naturally transfer to the bag-of-words representation: a global reference frame enables a more robust classification using smaller vocabularies.

Recent work on object detection [Chen et al., 2012, Felzenszwalb et al., 2010] suggests that it is crucial to consider intra-class variations of objects. It has been shown that the performance of an object recognition approach can be improved significantly by learning a mixture of classification models, where specific detectors learn variations of a class. Felzenszwalb et al. [2010] use a bounding box criterion to initialize different mixture components of a class. In our approach, we use distance, volume, and the extents of the three-dimensional bounding box as latent variables and additionally learn every mixture component using different parameterizations of a histogram descriptor. This choice is motivated by the distance-dependency of three-dimensional scans, i.e., we can distinguish fine details at small ranges, but get only a sparse point cloud at far distances (see Figure 5.1). In line with

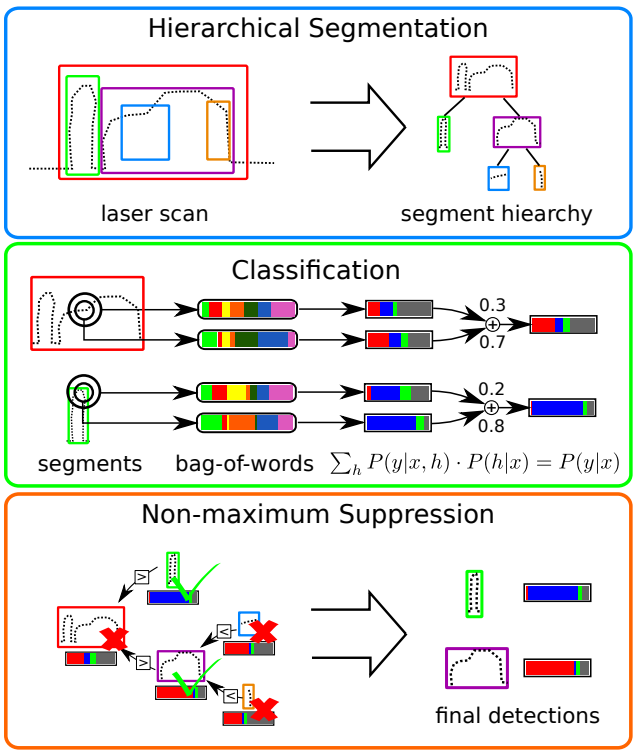


Figure 5.2: *Overview of our approach. First, we use a hierarchical segmentation with a coarse-to-fine segmentation. Each level of the segmentation is classified using a mixture of bag-of-words with different parameterizations of a local descriptor. In the final stage, we filter duplicate segments using the hierarchy.*

the experimental results of Chapter 4, "Efficient hash-based Classification," the combination of multiple classifiers enables an increase in classification performance. In contrast to the last chapter, we learn both the partition or weighting of classifiers, and the classification models depending on the re-weighted training examples.

Overall, our approach is divided in the following three stages: c.f. Figure 5.2: First, we propose a hierarchical segmentation approach resulting in coarse-to-fine hierarchies of segments. Our aim is to reduce the effects of segmentation errors on later stages in the classification pipeline. We explicitly include over- and under-segmentations and let the later stages filter these additional segments. In the second stage, we employ a mixture of multiple bag-of-word classifiers to classify all extracted segments. We use different parameterizations of a local descriptor for each classifier, which enables the overall approach to adapt to different aspects of the data. The results of the specialized classifiers are averaged using mixture weights jointly learned with the classifiers. In the final step, we filter duplicate detections. We apply a greedy breadth-first search strategy to ensure consistent final detection hypotheses with maximal confidence.

The main results of this chapter are published in [Behley et al., 2013] and in this chapter we additionally evaluate two strategies to improve the overall efficiency of the approach. Since we see a vast number of points on near ranges carrying no additional information, our first strategy aims at removing these superfluous points. On the contrary, the second strategy reduces the number of operations needed to construct a bag-of-words representation by a uniform sub-sampling of descriptor evaluations. This means we still use all points to compute the descriptor, but instead of densely evaluating a descriptor at every point of the segment, we use only a reduced set of points to determine the bag-of-words. Our extensive evaluation shows that both strategies reduce the overall runtime of the proposed approach significantly and maintain comparable predictive accuracy on the investigated detection task.

The chapter is organized as follows. In the next Section 5.1, "Related Work," we discuss prior work concerning segmentation and segment-based classification of laser range data. Section 5.2, "Fundamentals," briefly introduces laser-based segmentation and the concept of bag-of-words. Building on the foundations of a simple segmentation strategy, we propose in the next Section 5.3, "Approach," our hierarchical segmentation approach. We furthermore describe our multiple bag-of-words learning approach combining multiple classifiers using different vocabularies. In the following Section 5.4, "Improving the Efficiency," we propose two strategies to improve the overall runtime performance of our approach. In Section 5.5, "Experiments," we evaluate both the general approach and our strategies to improve the efficiency. Finally, in Section 5.6, "Summary," we conclude this chapter and outline interesting avenues for future work.

5.1 Related Work

Segmentation is a basic preprocessing step applied in many approaches dealing with large-scale three-dimensional point clouds. One goal of segmentation is the reduction of the overall number of points by discarding irrelevant segments.

The segmentation of laser range data is usually easier than the same task in images due to the availability of distance measurements, which can be used to determine object boundaries. Over the recent years, different segmentation algorithms – mainly to discriminate laser returns from drivable area from other obstacle points – were proposed. Herein, we will only reference approaches that directly process three-dimensional point clouds without any other information, such as images or map data.

Klasing et al. [2008] determine segments by an efficient distance-based clustering, where each cluster is defined by points with a given maximal distance to each other. Due to this property, the ground must be filtered beforehand. Himmelsbach et al. [2009] use a two-dimensional elevation map to discriminate between ground and non-ground points.

They group adjacent cells with large height differences above a threshold, and finally extract laser range points using oriented bounding boxes estimated from cell coordinates. Petrovskaya and Thrun [2009] exploit the smoothness of road surfaces to filter laser returns from ground in an autonomous driving context. Ground points are identified using an angular grid and the following heuristic: a point is labeled as ground, if the angle between vectors starting at that point is large enough. Himmelsbach et al. [2010] use a similar approach to filter ground points, combine non-ground points using an elevation map and further refine segments exceeding a pre-defined height threshold using three-dimensional voxel grids. Douillard et al. [2011] evaluate different approaches for dense and sparse laser data. For densely sampled data, a voxel-grid based approach is applied and sparse laser range scans are either handled by a mesh-based approach or Gaussian processes.

Other solutions explicitly exploit the geometry of the sensor to attain real-time capable solutions with graph-based approaches. Moosmann et al. [2009] efficiently build a mesh by exploiting the rotation of the Velodyne laser sensor — points in the scan are connected, if they are produced by the same laser diode or if they show a similar yaw angle. Mesh nodes are combined into a single segment using a region growing approach if they satisfy a local convexity criterion. Klasing et al. [2009] segment a scan incrementally using a surface normal criterion, where a two-dimensional laser scanner is vertically swept over the environment. Spinello et al. [2010, 2011] use jump distance clustering over scan lines of laser range points sorted by ascending azimuth angles. A segment is extracted if two consecutive points exceed a given threshold and all points share a similar distance towards the sensor.

All approaches share a non-trivial selection of suitable parameters and the selection is usually specific to the task and object classes of interest [Douillard et al., 2011]. We apply multiple stages of elevation-based segmentation, like Himmelsbach et al. [2009] or Teichman et al. [2011], and are therefore more independent of a specific choice of parameters. Compared to other approaches, our approach is independent of the data acquisition method, but still real-time capable due to an efficient implementation of the elevation maps. It generates more segments than really needed, but we rather filter these irrelevant segments later. The approach of van der Sande et al. [2011] also generates an over-complete hierarchy of segments in images, but they do not exploit the hierarchy to eliminate duplicate detections.

Classification of three-dimensional laser range data in urban environments was mainly investigated for dynamic objects. Himmelsbach et al. [2009] classify segments represented by a histogram of multiple point-based features and remission intensities using a support vector machine. Teichman et al. [2011] use tracking information to smooth the segment-based classification results of an AdaBoost-based approach [Friedman et al., 2000, Torralba et al., 2004]. The segments are represented by multiple spin images [Johnson and Hebert, 1999] with different resolutions and Histogram of Gradient (HoG) [Dalal and Triggs, 2005] features calculated on orthogonal projections of the point cloud. The feature set is addition-

ally enriched by holistic features, which represent track-based properties, and spin images calculated over accumulated and aligned point clouds from multiple track positions. Himmelsbach and Wuensche [2012] use tracking information to correct under- and over-segmentations. In their approach, segments are represented by the extent of the bounding box and track-based velocity features.

In contrast to these approaches, we aim at learning multiple classifiers using an individual bag-of-word per classifier, where each bag uses a different descriptor parameterization and consequently an individual vocabulary. The approach is related to the mixture-of-experts [Jacobs et al., 1991] and we use classifiers jointly learned using a probabilistic weighting of individual classifiers.

5.2 Fundamentals

We introduce basic concepts needed in context of our approach in this section. In the first part, we briefly describe segmentation and a basic approach for laser-based segmentation using an elevation map. This simple approach is then extended in later sections to result in more robust segmentations at different ranges. In the second part of this chapter, we formally introduce the concept of bag-of-words using local descriptors.

5.2.1 Segmentation

Segmentation is often applied as a preprocessing step in large-scale laser based perception. The goal of laser-based segmentation is the subdivision of a point cloud $\mathcal{P} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_N$ into mutually exclusive subsets $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, called *segments*, fulfilling a similarity or grouping criterion. Often these segments are then categorized into relevant and irrelevant segments using a simple heuristic, and only relevant segments are processed further. Main application of segmentation is therefore the reduction of the computational complexity in presence of a vast number of laser range points by discarding irrelevant segments. In our application of detecting cars, pedestrians, and bicyclists, we are only interested in parts of the point cloud that could potentially correspond to these object classes. Hence, we can filter out every point corresponding to flat drivable areas or ground and are specifically interested in all objects residing on the ground. In contrast to earlier chapters, we will not apply a classification approach to distinguish relevant and irrelevant points, but apply a simple segmentation approach using a grid representation. Later, we will use a classifier to assign object classes to the extracted segments.

segments

In the following, we describe an efficient and simple method to extract parts of the point cloud corresponding to objects that have a certain minimal height. We first build a simple grid map and then combine neighboring non-ground grid cells into complete segments.

A *grid map* is a regular two-dimensional grid $\mathcal{G} = \{\mathcal{G}_{i,j}\}, 0 \leq i < N, 0 \leq j < M$ of resolution (or grid cell size) r, where each grid cell $\mathcal{G}_{i,j}$ contains all points in a certain spatial location: grid map

$$\mathcal{G}_{i,j} = \left\{ \boldsymbol{p} \in \mathcal{P} |\ i \leq r^{-1} \left(\boldsymbol{p}^{(0)} + r \cdot \frac{N}{2} \right) < i + 1 \quad \wedge \right.$$
$$\left. j \leq r^{-1} \left(\boldsymbol{p}^{(1)} + r \cdot \frac{M}{2} \right) < j + 1 \right\}. \tag{5.1}$$

Each three-dimensional point is therefore projected onto the $x-y$ ground plane and assigned to a distinct grid cell and we assume that the point cloud is given in a local reference frame of the autonomous system, i.e., the sensor is always in the center of the grid. Reducing the point cloud to a two-dimensional representation enables us to efficiently identify potential obstacles and combine parts of the point cloud into coherent segments corresponding to objects of interest.

In the second step, we use the grid map to extract all grid cells $\mathcal{G}'_{i,j}$ that contain points with height differences larger than a given threshold η:

$$\mathcal{G}' = \left\{ \mathcal{G}_{i,j} \left| \max_{\boldsymbol{p} \in \mathcal{G}_{i,j}} \boldsymbol{p}^{(2)} - \min_{\boldsymbol{p} \in \mathcal{G}_{i,j}} \boldsymbol{p}^{(2)} > \eta \right. \right\} \tag{5.2}$$

This grid $\mathcal{G}'$ is also often denoted as *obstacle grid map* containing areas that are likely to be occupied at the current time. We emphasize that we are interested in a local map at the current time and do not perform registration or mapping as in simultaneous localization and mapping (SLAM) approaches [Thrun et al., 2005]. obstacle grid map

Since we might have projected single objects into different grid cells, we use the resulting obstacle grid map and the implicit neighboring relationship to combine adjacent occupied grid cells. To this end, we use efficient *flood fill* on the grid to find connected components in the grid representation: flood fill

1. Let $\mathcal{R} = \mathcal{G}'$ be the remaining obstacle grid cells to be visited, and $\mathcal{S}$ set of the segments $\mathcal{S}_k$ extracted from the obstacle grid map.
2. Take any $\mathcal{G}_{i,j}$ from $\mathcal{R}$ and initialize $\mathcal{Q} = \{\mathcal{G}_{i,j}\}$. Let $\mathcal{S}_k = \emptyset$ be the set of extracted segment points. While $\mathcal{Q} \neq \emptyset$, repeat the following steps.
   a) Remove the first element $\mathcal{G}_{i,j}$ from $\mathcal{Q}$ and update $\mathcal{R} = \mathcal{R} - \mathcal{G}_{i,j}$.
   b) Add points in $\mathcal{G}_{i,j}$ to $\mathcal{S}_k$, $\mathcal{S}_k = \mathcal{S}_k \cup \{\boldsymbol{p} \in \mathcal{G}_{i,j}\}$.
   c) Update $\mathcal{Q}$ with neighboring grid cells, i.e., $\mathcal{Q} = \mathcal{Q} \cup \{\mathcal{G}_{\mathcal{N}(i,j)} \in \mathcal{R}\}$, where $\mathcal{N}(i,j) = \{(i-1,j),(i+1,j),(i,j-1),(i,j+1)\}$.
3. Update segmentation, $\mathcal{S} = \mathcal{S} \cup \mathcal{S}_k$, and repeat step 2 until $\mathcal{R} = \emptyset$.

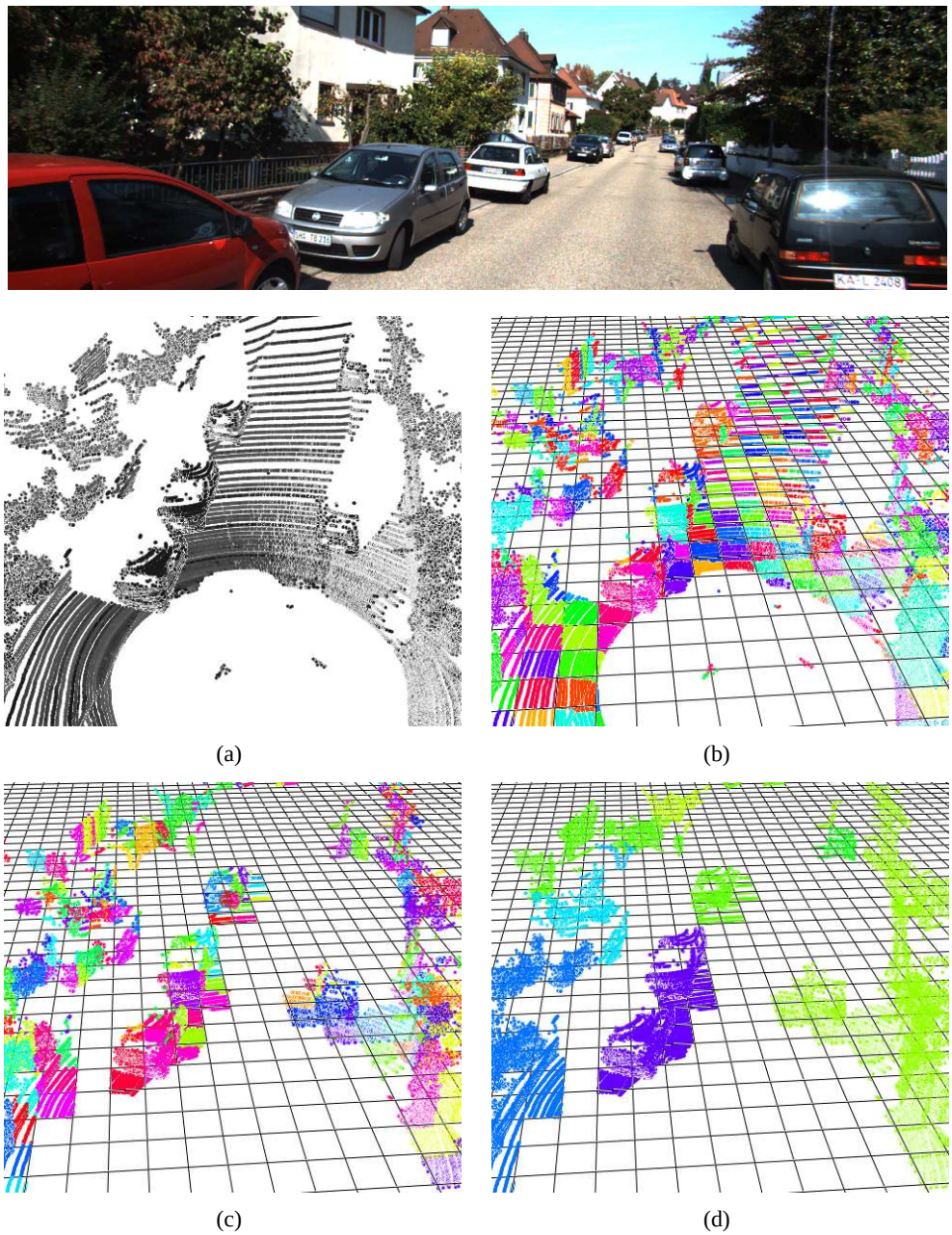


Figure 5.3: *Stages of the laser-based segmentation. On top the corresponding image of the same scene. In (a), the original point cloud of a street scene with some cars is shown. (b) First, the points are stored in a regular grid, where we indicated by colors the corresponding grid cell. (c) Only grid cells with points having a given height difference (*0.3 *m in the shown example) are considered to finally combine neighboring grid cells to segments (d). In the last image, an under-segmentation of two cars (purple points) is visible.*

The set of segments $\mathcal{S}$ is then reported as final segmentation of point cloud $\mathcal{P}$, where irrelevant ground points were already filtered. Figure 5.3 exemplarily shows some steps for a typical point cloud of urban environment (a), where we first show the subdivision of the point cloud in a regular grid (b), the obstacle grid map (c), and the final segments (d) extracted by the flood fill approach.

We refer to a segment as over-segmented, if the original object has been segmented in too many small segments. Under-segmentation is the opposite effect, where two distinct objects have been assigned to the same segment. Figure 5.3d shows an example of under-segmentation, where two cars (purple) get merged into a single segment. Both effects often occur in laser-based perception of outdoor environments and are directly caused by the fixed grid resolution. We show later how the effects of over- and under-segmentation can be alleviated by a hierarchical coarse-to-fine segmentation, where we extend the presented grid-based approach and use multiple grid resolutions.

over- and under-segmentation

5.2.2 Bag-of-words Representation

The concept bag-of-words originates in the natural language processing community and the domain of document classification [Manning et al., 2009]. For the purpose of assigning a document to a given topic, we do not have to really know where exactly each word is located in the document. In most cases, it is enough to simply take a look at the frequencies of certain words to decide on the topic of a document. Words such as 'algorithm', 'computer', or 'memory' are often used in computer science documents, but rarely occur in documents about ancient history. Thus, we can reduce documents of arbitrary length to a (normalized) histogram of word counts, which can be used in a classification approach.

The same idea can be transfered to other application areas, such as scene classification using images [Csurka et al., 2004, Sivic and Zisserman, 2003, 2009]. In this context the term 'visual word' is often used and it has been shown in numerous studies that a bag-of-words representation is sufficient to attain state-of-the-art results for challenging tasks, such as object detection [van der Sande et al., 2011] and recognition [Coates et al., 2011a,b]. But how do we get the mentioned visual words and a whole vocabulary of visual words to describe an image?

In almost all approaches, the words consist of single patch-based image descriptors of a small part of the image. The vocabulary is learned unsupervised from a large collection of images representing the intended application domain. It is common practice to sample descriptors randomly from these images and build the vocabulary by a clustering approach such as *k*-means [Manning et al., 2009]. The cluster centers finally form the vocabulary as they are expected to be representative to describe reoccurring descriptors. The bag-of-words is then build using the vocabulary by first extracting descriptors from the image and

then encoding these descriptors using the vocabulary. A simple method for encoding of an image, called vector quantization, is searching for the nearest vocabulary word and counting the occurrence by increasing the corresponding entry in the histogram.

Thus, the overall classification approach using a bag-of-words representation can be summarized as follows: (1) During training, we learn a vocabulary using a large number of descriptors extracted from the training set. (2) Next, the training set is encoded using the learned vocabulary. (3) Finally, we optimize the model of a classifier to discriminate the labeled classes using the encoded bag-of-words representation. Unseen test data is then classified by first encoding the data with the vocabulary and applying the learned classifier to this new representation.

Over the last years, a lot of research and evaluations concentrated on the improvement of this pipeline. Different techniques for learning a representative dictionary [Coates et al., 2011b] and different encoding methods were developed [Boureau et al., 2010, Lazebnik et al., 2006, Moosmann et al., 2007] and investigated [Chatfield et al., 2011]. The learning of a descriptive dictionary has been investigated and simple approaches, like *k*-means, proved to be competitive to more advanced methods [Coates et al., 2011b]. However, recent research indicates that the encoding, i.e., the assignment of a descriptor to dictionary entries, seems to be more important than the learning step [Chatfield et al., 2011, Coates and Ng, 2011].

In the next section, we will discuss laser-based classification of segments using bag-of-words. We propose an extension of the approach to incorporate different descriptors and independently learned dictionaries for the purpose of a better representation of segments.

5.3 Approach

Our objective is to determine all segments belonging to the classes pedestrian, car, and bicyclist, using only a single three-dimensional laser range scan. To this end, we regard the detection problem as a classification task and learn a classifier to output a probability distribution $P(y|\boldsymbol{x})$ for a segment $\boldsymbol{x}$ belonging to either the target classes or background. In a post-processing step, we finally remove segments belonging to background and also non-maximal detections, i.e., detections that overlap with other detections and are less likely than the other detections.

5.3.1 Hierarchical Segmentation

As already discussed in Section 5.2.1, model-free segmentation is usually less complex using a single laser range scan then using only a single image. This is mainly caused by the availability of depth information, which separates objects from each other and the ground.

Hence, in most cases less complex methods are sufficient to attain very good results. Still, we have to cope with under- and over-segmentation – especially in outdoor environments, where distances to objects range from few meters to more than 20 meters. Hence, the point cloud density varies drastically leading to difficulties in finding suitable parameters for distance-based segmentation methods, which result in coherent segments for different ranges. Finding a single parameter setting for the segmentation approach is hard and established segmentation approaches use specific heuristics to post-process the segmentation.

In this chapter, we are not aiming at generating a single perfect segmentation, but rather generate multiple coarse-to-fine segmentations. Later, we will use a classification approach in conjunction with an intelligent non-maximum suppression to finally decide which segments are irrelevant.

The basic building block of the proposed hierarchical segmentation approach is the height-based segmentation introduced in Section 5.2.1. Let r_0 be the resolution of this initial segmentation and, as before, η the height threshold to distinguish ground points from obstacle points. This initial segmentation already discards many ground points and results in a reduced set of segments. However, it still contains many under-segmented parts containing different objects.

For every segment, we further apply the height-based segmentation, but with a smaller resolution $r_{i+1} < r_i$, until we reach a desired depth. Thus, we get a smaller obstacle grid map and consequently can subdivide a segment into smaller sub-segments, if necessary. We finally get multiple *segment trees* containing at every level a finer segmentation of the original point cloud. Each segment tree contains in the root a single segment generated with the largest resolution and each child of a node is generated from the same segment at a larger resolution. Later, we will use these hierarchies to efficiently winnow out non-maximal or duplicated detections.

segment trees

Figure 5.4 shows an example of a three-level hierarchical segmentation, where segments of the same segment tree are depicted by the same color in each level of the segmentation. For better visualization, we projected the laser points of a segment into the image and calculated a convex hull of the two-dimensional projections; the segmentation itself uses only information from the point cloud. Segments of the first coarse segmentation (upper row) were generated with a resolution of $r_0 = 1.0\,\text{m}$, the second level (middle row) was further subdivided by a segmentation using $r_1 = 0.5\,\text{m}$, and the final level (bottom row) results from a subdivision with $r_2 = 0.2\,\text{m}$. The first coarse level is most effective at far ranges, where laser range measurements get sparse and objects are only covered by very few laser range measurements. In the second row, we see that a finer grid enables the segmentation approach to subdivide the large segments into smaller ones — for instance the subdivision of the large red segment into multiple cars in the left of the image. Although we see many correct segmentations of cars, we still see some under-segmentations, e.g., the person on the right is still not satisfactorily segmented. In the last and finest level, we see the exposure

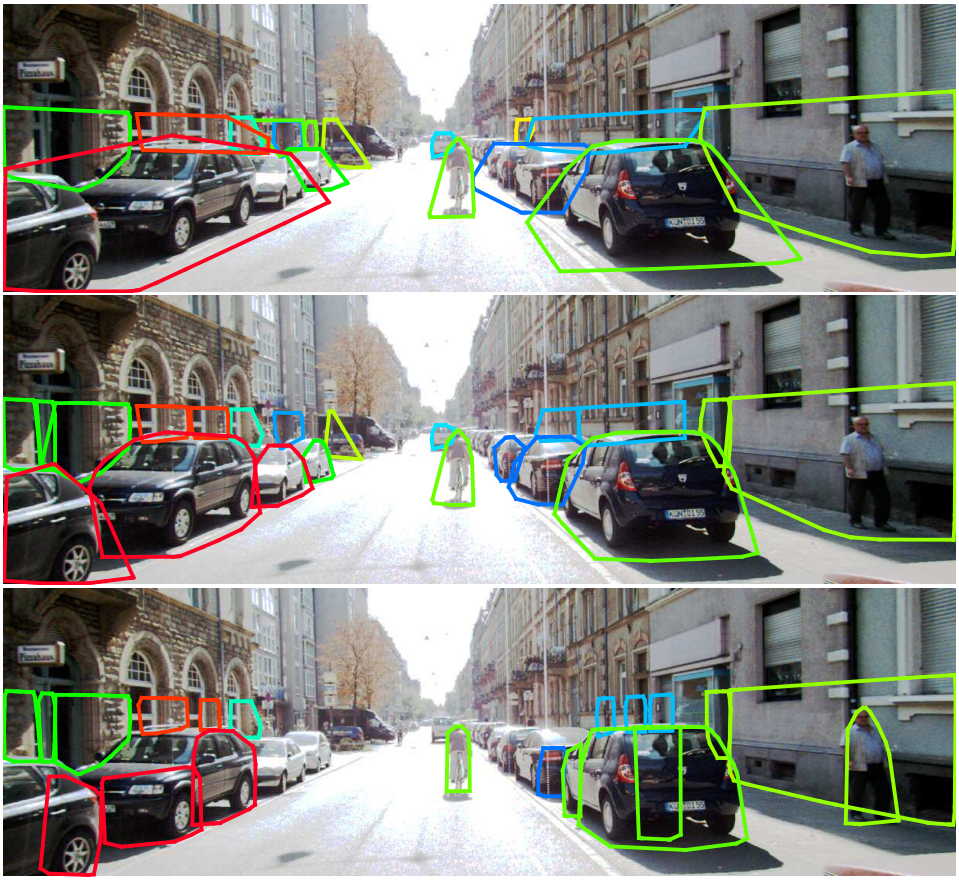


Figure 5.4: *Exemplary results of the hierarchical segmentation. The first row shows the segmentation using a grid resolution of* 1.0 *m, the second row refines the segmentation using a resolution of* 0.5 *m, and the bottom row using a resolution of* 0.2 *m. The coarse resolution of* 1.0 *m is best at large distances and produces more consistent segments of cars at far distances then the finer grids, but generates also under-segmentations visible on the left side of the image. A finer resolution of* 0.5 *m leads to splitting of the large segment in three distinct cars, but still is not able to correctly segment the person on the right side of the image. The last layer with a very fine resolution of* 0.2 *m manages to separate the pedestrian and the wall, but generates also many over-segmentations.*

of the pedestrian near the wall, but also an over-segmentation of cars. None of the single segmentation results alone contains all desired segments — cars and bicyclists are best segmented using a moderate grid resolution, and pedestrians using a very fine grid. Later in the experimental evaluation, we will quantify the improvement in the quality of the generated segmentation.

5.3.2 Learning a mixture of bag-of-words

Given the segment trees, we determine multiple bag-of-word representations using only points from each segment. In particular, we learn multiple vocabularies on subsets of the training data using differently parameterized descriptors. We ultimately aim at learning context-dependent vocabularies to account for differences in the density of the laser range measurements.

We are interested in a discriminative classification approach $P(y|\boldsymbol{x})$, and for our purposes we introduce a hidden or latent variable h:

$$P(y|\boldsymbol{x}) = \sum_h P(y, h|\boldsymbol{x}) \tag{5.3}$$

$$= \sum_h P(h|\boldsymbol{x})P(y|h, \boldsymbol{x}) \tag{5.4}$$

The value of the hidden variable $h \in \{1, \ldots, M\}$ depends on the segment $\boldsymbol{x}$ and for each hidden variable we learn a separate multi-class classifier $P(y|h, \boldsymbol{x})$, where $y \in \{1, \ldots, K\}$. Each segment classifier $P(y|h, \boldsymbol{x})$ uses a separately learned bag-of-word representation using differently parameterized laser features. $P(h|\boldsymbol{x})$ determines the weighting of the single segment classifier results regarding the distance, extent, and volume of the segment bounding box. Similar to the last chapter, where we separated the overall classification problem into multiple local classification problems on subsets of the feature space, we try to learn classifiers specializing on different aspects of the classification problem. But now we explicitly learn the weighting of the local classification models and additionally use multiple feature representations.

Since both models map a feature vector to a discrete target value, either a hidden variable or a label, we learn a softmax regression for both models $P(h|\boldsymbol{x})$ and $P(y|h, \boldsymbol{x})$, cf. Section 2.2.1:

$$P(h = j|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_j^T \cdot \boldsymbol{x})}{\sum_l \exp(\boldsymbol{w}_l^T \cdot \boldsymbol{x})} \tag{5.5}$$

$$P(y = k|h, \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_{k,h}^T \cdot \boldsymbol{x})}{\sum_l \exp(\boldsymbol{w}_{l,h}^T \cdot \boldsymbol{x})} \tag{5.6}$$

Here $\boldsymbol{w}_h$ and $\boldsymbol{w}_{y,h}$ represent the weight vectors for every hidden variable h and class y, respectively. In the following, we summarize these parameter vectors of all models by $\boldsymbol{\theta}_t = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M, \boldsymbol{w}_{1,1}, \ldots, \boldsymbol{w}_{K,1}, \ldots, \boldsymbol{w}_{1,M}, \ldots, \boldsymbol{w}_{K,M})$, where t denotes the iteration in the optimization process.

In Section 2.2, "Classification," we discussed the estimation of model parameters $\boldsymbol{\theta}$ given a

training set $\mathcal{X}$ by maximizing 2.13:

$$\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{5.7}$$

$$= \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathcal{X}) \tag{5.8}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i} P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) P(\boldsymbol{\theta}). \tag{5.9}$$

Assuming a uniform prior $P(\boldsymbol{\theta})$, we can now replace $P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})$ by Equation 5.4:

$$\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}} \prod_{i} \sum_{h} P(h|\boldsymbol{x}_i, \boldsymbol{\theta}) P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}). \tag{5.10}$$

In line with Section 2.2.1, we apply our known 'learning machinery': (1) derive a gradient of the log-likelihood in respect to our parameters $\boldsymbol{\theta}$, and (2) use gradient-based optimization to find the maximum $\boldsymbol{\theta}^{\star}$. First, we use the logarithm to get a more accessible expression of the likelihood:

$$\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}} \sum_{i} \log \left[\sum_{h} P(h|\boldsymbol{x}_i, \boldsymbol{\theta}) P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}) \right] \tag{5.11}$$

Unfortunately, this expression is intractable with our basic recipe, since we have a coupling between parameters of the models $P(h|\boldsymbol{x}_i, \boldsymbol{\theta})$ and $P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta})$. This is caused by the sum over hidden variables h inside the logarithm, which makes it intractable to calculate the gradient in closed form.

Expectation Maximization

Fortunately, there exist an approximate and efficient solution known as *Expectation Maximization* [Dempster et al., 1977]. Instead of maximizing the log-likelihood of Equation 5.11 directly, we maximize a carefully chosen lower bound $\mathcal{B}_{\boldsymbol{\theta},q}$:

$$\mathcal{B}_{\boldsymbol{\theta},q} = \sum_{i} \sum_{h} q_h(\boldsymbol{x}_i, y_i) \log \left[\frac{P(h|\boldsymbol{x}_i) P(y_i|h, \boldsymbol{x}_i)}{q_h(\boldsymbol{x}_i, y_i)} \right] \tag{5.12}$$

$$\leq \sum_{i} \log \left[\sum_{h} P(h|\boldsymbol{x}_i) P(y_i|h, \boldsymbol{x}_i) \right], \tag{5.13}$$

Jensen's inequality

where *Jensen's inequality* $\sum P(y) \log(y) \leq \log \left[\sum P(y) \cdot y \right]$ was exploited by adding the probability distribution $q_h(\boldsymbol{x}_i, y_i)$. Expectation Maximization alternates now the following steps to improve the lower bound $\mathcal{B}_{\boldsymbol{\theta},q}$ iteratively:

1. **(E-Step)** Estimate $q_h(\boldsymbol{x}_i, y_i)$ to maximize $\mathcal{B}_{\boldsymbol{\theta}_{t-1},q}$ with fixed $\boldsymbol{\theta}_{t-1}$.
2. **(M-Step)** Maximize $\mathcal{B}_{\boldsymbol{\theta}_t,q}$ in respect to $\boldsymbol{\theta}_t$ with fixed $q_h(\boldsymbol{x}_i, y_i)$.

One can show, see e.g. Prince [2012], that this alternating process increases the bound $\mathcal{B}_{\theta,q}$ in every step if we use $P(h|\boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1})$ to estimate $q_h(\boldsymbol{x}_i, y_i)$. But we have to note that Expectation Maximization does not necessarily converge to the global optimum, i.e., the solution might be only a local optimal solution and depends on the initial parameters $\boldsymbol{\theta}_0$.

Continuing the derivation of the parameter estimation for our mixture model, we determine the distributions $q_h(\boldsymbol{x}_i, y_i)$ for every training instance $(\boldsymbol{x}_i, y_i)$ by $P(h|\boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1})$ using the parameters $\boldsymbol{\theta}_{t-1}$ from the last iteration $t-1$:

$$
\begin{aligned}
q_h(\boldsymbol{x}_i, y_i) &= P(h|\boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1}) && && (5.14)\\
&= \frac{P(h, \boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1})}{P(\boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1})} && \text{using (A.2)} && (5.15)\\
&= \frac{P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})P(h, \boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})}{P(\boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1})} && \text{using (A.3)} && (5.16)\\
&= \frac{P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})P(h, \boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})P(\boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})}{P(\boldsymbol{x}_i, y_i, \boldsymbol{\theta}_{t-1})P(\boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})} && && (5.17)\\
&= \frac{P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})P(h|\boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})}{P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})} && \text{using (A.2)} && (5.18)\\
&= \frac{P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})P(h|\boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})}{\sum_k P(y_i|\boldsymbol{x}_i, k, \boldsymbol{\theta}_{t-1})P(k|\boldsymbol{x}_i, \boldsymbol{\theta}_{t-1})}, && && (5.19)
\end{aligned}
$$

where we replace the denominator in Equation 5.18 by Equation 5.4. In consequence, the distribution $q_h(\boldsymbol{x}_i, y_i)$ encodes for each hidden variable h how well the current parameter explains the training example $\boldsymbol{x}_i$ with label y_i.

The log-likelihood $l(\boldsymbol{\theta}_t)$ in the M-Step with fixed $q_h(\boldsymbol{x}_i, y_i)$ is given by

$$
\begin{aligned}
l(\boldsymbol{\theta}_t) &= \sum_i \sum_h q_h(\boldsymbol{x}_i, y_i) \log\left[\frac{P(h|\boldsymbol{x}_i, \boldsymbol{\theta}_t)P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_t)}{q_h(\boldsymbol{x}_i, y_i)}\right] && (5.20)\\
&= \sum_i \sum_h q_h(\boldsymbol{x}_i, y_i) \left\{\log\left[P(h|\boldsymbol{x}_i, \boldsymbol{\theta}_t)P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_t)\right] - \log\left[q_h(\boldsymbol{x}_i, y_i)\right]\right\} && (5.21)\\
&= \sum_i \sum_h q_h(\boldsymbol{x}_i, y_i) \log\left[P(h|\boldsymbol{x}_i, \boldsymbol{\theta}_t)P(y_i|h, \boldsymbol{x}_i, \boldsymbol{\theta}_t)\right], && (5.22)
\end{aligned}
$$

where we dropped terms not depending on $\boldsymbol{\theta}$ in the last line. Hence, the partial derivatives in respect to the parameters $\boldsymbol{w}_j$ and $\boldsymbol{w}_{k,h}$ are given by:

$$
\begin{aligned}
\frac{\partial l}{\partial \boldsymbol{w}_j} &= \sum_i \sum_h q_j(\boldsymbol{x}_i, y_i)\left[\mathbf{1}\{h = j\} - P(j|\boldsymbol{x}_i)\right]\boldsymbol{x}_i && (5.23)\\
\frac{\partial l}{\partial \boldsymbol{w}_{k,h}} &= \sum_i q_h(\boldsymbol{x}_i, y_i)\left[\mathbf{1}\{y_i = k\} - P(k|h, \boldsymbol{x}_i)\right]\boldsymbol{x}_i. && (5.24)
\end{aligned}
$$

Comparing these partial derivatives to the partial derivative of the softmax regression (see Equation 2.28), we note that parameters $\boldsymbol{w}_j$ and $\boldsymbol{w}_{k,h}$ are optimized using a weighting of every training example.

In summary, the complete training of the mixture components involves the following steps:

1. Estimate $q_h(\boldsymbol{x}_i, y_i)$ for every training example $(\boldsymbol{x}_i, y_i)$ using Equation 5.19 and the parameters $\boldsymbol{\theta}_{t-1}$ from last iteration $t-1$.
2. Re-learn vocabularies $\mathcal{V}_k$ over subset $\mathcal{X}_k = \{\boldsymbol{x}_i | q_k(\boldsymbol{x}_i, y_i) \geq q_h(\boldsymbol{x}_i, y_i)\}$.
3. Maximize Equation 5.22 with respect to $\boldsymbol{\theta}$ after encoding every segment using the newly learned vocabularies $\mathcal{V}_k$.

5.3.3 Hierarchical Non-maximum Suppression

Using the learned mixture model, we classify every segment in all hierarchies and get $P(y|\boldsymbol{x})$ for every segment. As we might get contradicting classifications in one hierarchy, we have to determine which of the segments are likely to be correct hypotheses and suppress non-maximal detections.

For this purpose, we use a greedy algorithm starting at the root of every hierarchy and descend the tree in breadth-first order. Background segments are not reported. We mark a segment for the final set of reported segments, when the overlap with non-background parent nodes is smaller than than a threshold γ. In this case, we assume that we found a smaller segment, which for itself is a valid detection, such as a person standing by a car. If the overlap between a node and an ancestral node is larger than γ, we suppress the non-maximal detection, i.e., the hypothesis where $P(y|\boldsymbol{x})$ is smaller. Thus, if an ancestral node classifies a segment differently at a coarser level, we only report the detection with larger confidence. The non-maximum suppression is summarized in Algorithm 2.

5.4 Improving the Efficiency

The proposed approach needs to calculate a descriptor for every laser range point of the segments. In Chapter 3, “Histogram Descriptors for Laser-based Classification,” we discussed that the computation of point-wise descriptors and most descriptors involve (1) the search for nearest neighbors in a certain radius r, and (2) the accumulation of statistics over the support. The runtime of both processing steps mainly depends on the number of points N and the radius r of the descriptor. Enlarging the support radius r usually entails the inspection of larger regions in the nearest neighbor data structure, and as more points are inside the neighborhood, more points must be accumulated to derive the descriptor.

Algorithm 2: Hierarchical Non-maximum suppression

Input: Segment tree $\mathcal{T} = \{n_0, n_1, \ldots, n_M\}$, where $n_i = (P(y|\boldsymbol{x_i}), \boldsymbol{x}_i)$
Result: set of detections $\mathcal{D}$

Let $\pi(n) : n \rightarrow n$ return the parent node of n or $\perp$ for the root.
Let $C(n) = \{n_i | n = \pi(n_i)\}$ be the children of a node n.
Let $\mathbf{Q} = [n_0]$ be a queue containing initially the root node n_0 of $\mathcal{T}$.

while $Q \neq \emptyset$ **do**
  remove first element n_j from $\mathbf{Q}$
  append $C(n_j)$ to $\mathbf{Q}$
  $\mathcal{D} = \mathcal{D} \cup \{n_j\}$
  $n_k = \pi(n_j)$
  while $n_k \neq \perp$ **do**
    if *Overlap between* n_k *and* $n_j < \gamma$ **then**
      break
    else if n_k *is not background* ***and*** $\max_y P(y|\boldsymbol{x}_j) > \max_y P(y|\boldsymbol{x}_k)$ **then**
      $\mathcal{D} = \mathcal{D} \setminus \{n_k\}$
    else if n_k *is not background* **then**
      $\mathcal{D} = \mathcal{D} \setminus \{n_j\}$
      break
    $n_k = \pi(n_k)$
  end
end

The first approach for more efficient processing is the reduction of the number of laser range points. Our aim is the removal of redundant laser points, which do not carry additional information, such as duplicated or very close points. However, we can only remove a certain number of points until we lose information.

Another option to accelerate the the calculation is to reduce the amount of needed computations by omitting some of the descriptor evaluations. Instead of estimating a descriptor for every laser range point, we just calculate a reduced set of descriptors and generate from this set a reduced bag-of-words.

We investigate the effects – benefits and drawbacks – of both options to speed-up the computation of bag-of-words vectors. Both variants need an efficient implementation, i.e., both the reductions and the computations must be possible in less time than computing all descriptors. Therefore, we will also discuss implementation details needed to achieve an efficient preprocessing time.

5.4.1 Point Sub-Sampling

voxel grid

The point filtering uses a regular *voxel grid* with voxel size ρ, where a voxel $v_{i,j,k}$ with indices i, j, k is specified by its midpoint $\boldsymbol{m}^{i,j,k}$. For each voxel, we first determine all points $\mathcal{P}^{i,j,k} \subset \mathcal{P}$ inside the voxel:

$$\mathcal{P}^{i,j,k} = \left\{ \boldsymbol{p} \,\middle|\, \left\| \boldsymbol{p} - \boldsymbol{m}^{i,j,k} \right\|_\infty < \frac{\rho}{2} \right\}, \tag{5.25}$$

where $\|\cdot\|_\infty$ denotes the maximum norm $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$, and replace these points with their average

$$\overline{\boldsymbol{p}}^{i,j,k} = |\mathcal{P}^{i,j,k}|^{-1} \sum_{\boldsymbol{p} \in \mathcal{P}^{i,j,k}} \boldsymbol{p} \tag{5.26}$$

in the resulting point cloud $\mathcal{P}'$. Consequently, we only have to store voxel-wise means and generate from these means the new point cloud. Increasing the voxel size ρ reduces the overall number of points, but reduces also the density of the point cloud.

The naive implementation of the voxel grid is too inefficient, so we have to carefully implement the voxel grid. Depending on the resolution ρ of the voxel grid, we have to store a large number of voxels. Allocation of a large number of voxels is usually too inefficient and therefore we reuse the data structure by reseting each voxel before we insert new laser range points. We can additionally reduce the needed resetting operations in every iteration by only resetting voxels containing points. Thus, we store a separate list of occupied voxels, which is updated on inserting a laser range point into a voxel, allowing us to reset only occupied voxels. Both implementation details improve the overall performance of the point sub-sampling significantly.

5.4.2 Descriptor Sub-Sampling

For reducing the number of descriptor evaluations per segment, we also use the already introduced voxel grid. But in contrast to the point sub-sampling, we only generate a voxel grid for a single segment. For each voxel, we then select a point from the original segment point cloud and calculate the descriptor using all segment points. The regular subdivision of the segment guaranties that we uniformly sub-sample the descriptor calculations in respect to the segment. Increasing the voxel size consequently reduces the number of descriptor evaluations, but should allow us to extract a bag-of-words that keeps the relative proportions between the entries.

5.5 Experiments

In this section, we experimentally evaluate segmentation and detection on challenging real-world datasets. First, we compare the proposed hierarchical segmentation with a single layer height-based segmentation. Then, we will use the hierarchical segmentation to extract segments and classify these segments either with a single bag-of-words or the proposed mixture of bag-of-words. Finally, we present results demonstrating the computational speed-up using the proposed sub-sampling of laser range points or descriptors.

Dataset and evaluation metric. For evaluation of the complete pipeline, we use the recently published KITTI Vision Benchmark Dataset [Geiger et al., 2012]. We additionally use the Stanford Track Collection (STC) [Teichman et al., 2011] for experiments using the classification model only. All data was recorded using a car equipped with common sensors used in autonomous driving context, including a Velodyne laser rangefinder, and an inertial navigation system for odometry information. In both datasets, we have to classify cars, cyclists, and pedestrians in everyday traffic situations.

The KITTI dataset contains 7,481 annotated images with additional Velodyne scans and appropriate calibration information. Additionally, 7,518 unlabeled test images with laser range scans are provided, where the task is to detect dynamic objects by annotating the image with bounding boxes. We have to emphasize that we solely use the laser scans in the following experiments and therefore project scan points into the image using the provided calibration matrices to estimate an image-based bounding box.

The detections are evaluated and scored following common image-based detection metrics [Everingham et al., 2010] and must be sent to a server-side evaluation script. Thus, we present here results for different parameter values using the training set only and will report results on the testset for a specific setting later.

More specific, we have to provide for the (test) images *bounding boxes* containing the object classes of interest. Thus, we have to localize the object classes in the images and return also a score for each bounding box, which corresponds to our belief that this bounding box contains the object classes, i.e., we simply return the probability $P(y|\mathbf{x})$ in our approach. In the following derivations, we denote a detection by a pair (D, s), where D corresponds to the bounding box and s its score. An (axis-aligned) bounding box D is defined by its top-left (l_D, t_D) and bottom-right corner (r_D, b_D).

To determine the number of found objects over the whole collection of images, all detections are compared to manually annotated ground truth bounding boxes. All of these ground truth bounding boxes are additionally annotated with a class label, an occlusion ratio, i.e., a qualitative categorization, and a truncation value, i.e., the amount of the object, which is not inside the image. Depending on these values, the following bounding box difficulties were defined by Geiger et al. [2012]:

- *easy*: bounding boxes of at least 40 pixels height, fully visible, and up to 15% truncated,
- *moderate*: bounding boxes of at least 25 pixels height, at least partial visible, and up to 30% truncated,
- *hard*: bounding boxes of at least 25 pixels height, at most difficulty to see, and up to 30% truncated.

We concentrate on *easy* and *moderate* difficult bounding boxes in the following discussion, since bounding boxes classified as *hard* are sometimes incorrectly annotated by humans.

For the evaluation, all detections need to be matched to a corresponding ground truth bounding box. To allow for small inaccuracies in the manual annotation with bounding boxes, the detections are matched using the bounding box overlap $O(A, B)$:

$$O(A, B) = \frac{\text{Area}(A \cap B)}{\text{Area}(A \cup B)} \tag{5.27}$$

$$= \frac{\text{Area}(A \cap B)}{\text{Area}(A) + \text{Area}(B) - \text{Area}(A \cap B)}, \tag{5.28}$$

where A, B correspond to bounding boxes. The area $\text{Area}(A)$ of a bounding box A is given by $(r_A - l_A) \cdot (b_A - t_A)$ corresponding to width and height of the rectangle. The intersection of two bounding boxes $A \cap B$ is given by top-left $(\max\{l_A, l_B\}, \max\{t_A, t_B\})$ and bottom-right corner $(\min\{r_A, r_B\}, \min\{b_A, b_B\})$. Geiger et al. [2012] require a minimal overlap $\omega = 0.5$ between a detection and an annotated bounding box for pedestrians and cyclists, and a minimal overlap $\omega = 0.7$ for cars. In the following, we say a detection D is *matched* to a ground truth annotation B, if the overlap is larger than ω, i.e., $O(D, B) \geq \omega$.

matching

To quantify the class-wise performance of the detection approach, all detections D_i are first ranked according to their score s_i and then matched to ground truth bounding boxes, where we only consider bounding boxes from a single class. Let $\mathcal{B} = \{B_1, \ldots, B_N\}, |\mathcal{B}| = N$ be all ground truth annotations of a certain label, i.e., we take only ground truth annotations of single class into account, and $\mathcal{D} = \{(D_1, s_1), \ldots, (D_M, s_M)\}$ the detection bounding boxes D_i with corresponding score s_i of the same label. Furthermore, we need the following sets,

$$\mathcal{D}(s) = \left\{(D_i, s_i) \mid s_i > s\right\} \tag{5.29}$$

of all detections with score larger than s and

$$\mathcal{B}(s) = \mathcal{B} - \left\{B \;\middle|\; \max_{(D_i, s_i) \in \mathcal{D}(s)} O(B, D_i) > \omega\right\} \tag{5.30}$$

of all ground truth bounding boxes, which have not been matched, i.e., there exist no other matching detection with score larger than s. These definitions are required to ensure that

each ground truth annotation is only matched to a single detection. Thus, each additional matching detection for an already matched ground truth bounding box is counted as false positive.

After this preparations, we can now define *ranked precision* p_k and *ranked recall* r_k up to rank k using the k top ranked detections $\mathcal{D}^k = \{(D_i, s_i) \in \mathcal{D} \,|\, |\mathcal{D}(s)| < k\}$ and ground truth annotations $\mathcal{B}$: ranked precision and recall

$$p_k = \frac{\left|\left\{(D,s) \in \mathcal{D}^k \,\middle|\, \max_{B \in \mathcal{B}(s)} O(B,D) > \omega\right\}\right|}{\left|\mathcal{D}^k\right|} \tag{5.31}$$

$$r_k = \frac{\left|\left\{(D,s) \in \mathcal{D}^k \,\middle|\, \max_{B \in \mathcal{B}(s)} O(B,D) > \omega\right\}\right|}{|\mathcal{B}|} \tag{5.32}$$

Similar to the class-wise precision and recall in Equation 3.5 and 3.6, the nominator corresponds to the total number of correctly matched detections up to rank k The denominator of the precision is simply the total number of elements up to rank k, i.e., $\left|\mathcal{D}^k\right| = k$ and therefore expresses the rank precision p_k, i.e., the precision of the first k detections. Since we already filtered the bounding boxes to include only bounding boxes of a single class, the denominator is the total amount of annotated bounding boxes $\mathcal{B}$. From the precision-recall pairs (p_k, r_k), we can now get the *precision-recall curve* $\pi(r_k) = p_k$, where intermediate values are linearly interpolated. The precision-recall curve graphs the performance of the detection approach, but here we will use only the *interpolated precision-recall curve* defined by

$$\pi_{\text{interp}}(r) = \max_{r' \geq r} \pi(r'). \tag{5.33}$$

In line with Everingham et al. [2010], the *average precision* (AP) [Manning et al., 2009] will be used to asses the performance of the detection approach. The average precision is the average over equally distant points of the interpolated precision-recall curve $\pi_{\text{interp}}(r)$ at 11 recall levels $\mathcal{R} = \{0.0, 0.1, \ldots, 1.0\}$: average precision

$$\text{AP} = \frac{1}{11} \sum_{r \in \mathcal{R}} \pi_{\text{interp}}(r) \tag{5.34}$$

We used the provided evaluation script of Geiger et al. [2012] to compute the average precisions for our detections.

The STC dataset contains roughly 14,000 tracks with segments extracted by a height-based segmentation and 83.3% of all segments are background. Note that we get pre-segmented laser scans and therefore evaluate only the classification model, either using a single vocabulary or the proposed mixture of multiple vocabularies. We use the same experimental setup as Teichman et al. [2011] and consequently evaluate the performance using classifica-

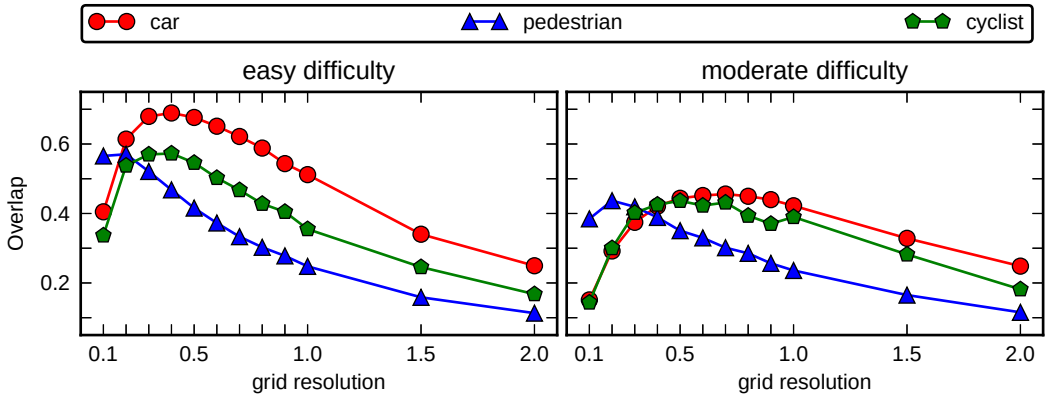


Figure 5.5: *Overlap with ground truth annotations. Shown is the overlap of the single layer segmentation for 'easy and 'medium' segments. We get different optimal grid resolutions depending on class and distance.*

resolutions	all	car	pedestrian	cyclist
(0.7)	0.53/0.45	0.62/0.46	0.33/0.30	0.47/0.43
(1.0, 0.5)	0.60/0.49	0.68/0.50	0.43/0.37	0.53/0.45
(1.0, 0.5, 0.2)	0.69/0.51	0.73/0.52	0.58/0.49	0.61/0.47

Table 5.1: *Overlap results for hierarchical segmentation (easy/moderate bounding boxes).*

tion accuracy, i.e, number of correctly classified segments divided by the overall number of segments.

Implementation details. We calculated spin images [Johnson and Hebert, 1999], since these proved to be effective and also efficient for the point-wise classification, see Chapter 3, "Histogram Descriptors for Laser-based Classification." All descriptors are calculated using a global reference frame, i.e., we use the z-axis to determine the bin in the histogram, which also significantly improved the descriptiveness of the bag-of-words (cf. Section 3.5). We used for all spin images 5 bins per dimension and performed a bilinear interpolation to calculate the contributions of every neighboring point. Every descriptor vector is finally normalized using the maximum norm L_∞.

We learn the vocabularies using off-the-shelf k-means clustering [Arthur and Vassilvitskii, 2007] and encode the descriptors using a hard vector quantization, i.e., we search in a kD-tree [Arya et al., 1998] for the nearest cluster center. Finally, we normalize the resulting bag-of-words vector using the L_1 norm.

All reported timings were measured on a system equipped with an Intel Xeon X5550 with 2.67 GHz and 12 GB memory using a single thread implementation.

5.5.1 Bounding box overlap

In the first experiment, we investigate the performance of the proposed hierarchical segmentation. For this purpose, we generate segments for all provided training data using either a single-layer, two-layer, or three-layer hierarchy. The laser points extracted by these approaches are then projected into the image and an image-based bounding box is determined. For all approaches, we used a minimum height $\eta = 0.3$ and discarded segments with fewer than 50 laser points.

Next, we determine the maximal overlap $o_{k,j} = \max_{B \in \mathcal{B}^k} O(A_j, B)$ between annotated bounding boxes $\mathcal{B}^k$ of scan k and bounding boxes generated from the laser range scan $A_j \in \mathcal{A}^k$.The overall overlap score is then averaged over all N scans to get the average overlap O for each class:

$$O = N^{-1} \sum_k |\mathcal{A}^k|^{-1} \sum_j o_{k,j}. \tag{5.35}$$

Figure 5.5 depicts the class-wise performance of the single layer segmentation with different grid resolutions. As motivated in the beginning, we can see that a generic choice of the resolution parameter is difficult. While for pedestrians, a smaller grid is preferred to reduce over- and under-segmentation, the resolution should be larger for cars and cyclists. But also for different distances, we can observe a dependence: nearby objects are better segmented using a smaller resolution, while objects at larger distances are better segmented using a larger resolution. This dependence is hardly surprising, since laser points show a larger sparsity and distance to each other at large distances.

Table 5.1 shows the best results of the single-, two- and three-layer segmentations, where we selected the best configuration for each segmentation approach using the moderate overall overlap. As can be seen from these results, the proposed multi-layer segmentation approaches clearly outperform the single-layer approach. Especially the results for pedestrian (increase of up to 0.25 overlap) and cyclist (increase of up to 0.14 overlap) are noteworthy.

Despite the significant increase in performance, we still have a gap of more than 0.3 between image-based and laser-based bounding boxes. A reason for this is that black objects can not be detected by the laser range sensor. Therefore, a lot of black cars, which can be easily marked in an image, are simply invisible in the laser range data or only represented by non-black parts in the point cloud. Furthermore, glass is sometimes not sensed by the laser sensor either and hence we get very few points on car windows. Furthermore, segments of cars at larger distance usually do not include the roof part and consequently, only partly overlap the annotation in the image, which includes also the windows.

	pedestrian	car	cyclist	background
training	2090/1140	5400/8844	584/401	152158/23827
validation	220/119	571/895	70/43	*n*/*a*

Table 5.2: *Segments per class (easy/moderate bounding boxes)*

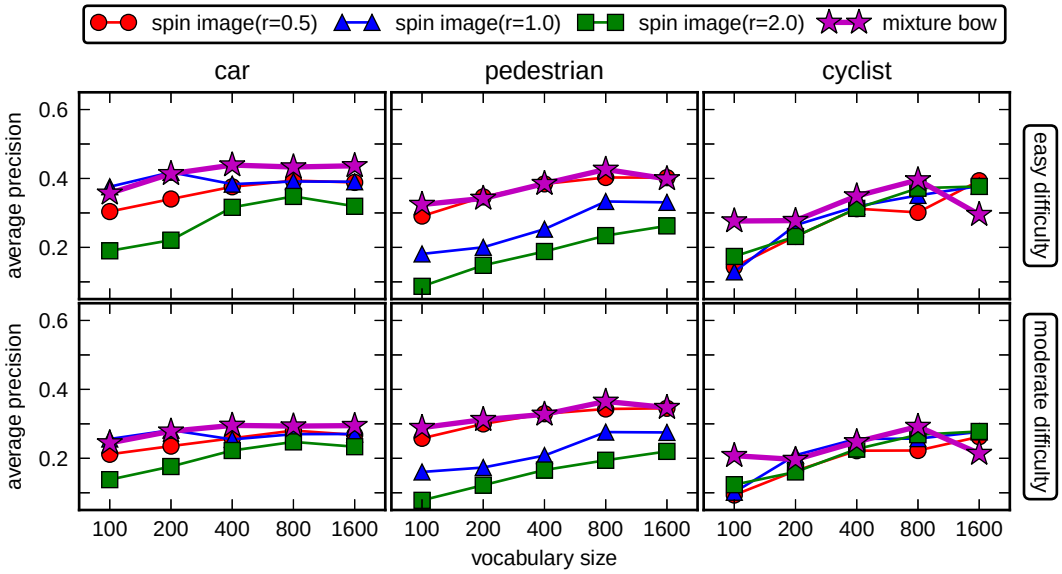


Figure 5.6: *Influence of the vocabulary size on single-layer bow with different parameterizations of spin images and the proposed mixture of bag-of-words. The upper row depicts the results for 'easy' bounding boxes, and the lower row shows the results for 'medium' bounding boxes.*

5.5.2 Detection performance

As introduced earlier, the results presented in this section are generated using a randomly selected validation set. For this purpose, we selected 10% of the training laser scans uniformly at random (see Table 5.2). For training and validation set, we applied the three-layered hierarchical segmentation with $r_0 = 1.0, r_1 = 0.5$, and $r_2 = 0.2$ and ignored segments with less than 50 points and width or length larger than 6 m. In the training data, background segments were discarded if the image-based overlap to ground truth annotation was larger than 0.2. We used $\gamma = 0.5$ for the hierarchical non-maximum suppression.

The performance of bag-of-words (bow) approaches is primarily influenced by the size of the vocabulary and the choice of the descriptor. Figure 5.6 shows the influence of the size of the vocabulary and the results for different support radii of the spin images (0.5, 1.0 and 2.0 m radius) with 5 bins in each dimension. The smallest spin image with a bin resolution

approach	car	pedestrian	cyclist
LSVM-MDPM-sv [Geiger et al., 2011]	0.68/0.56	0.47/0.39	0.38/0.29
LSVM-MDPM-us [Felzenszwalb et al., 2010]	0.66/0.55	0.45/0.38	0.35/0.27
Mixture of bag-of-words	0.36/0.23	0.44/0.31	0.28/0.21

Table 5.3: *Results on the testset (easy/moderate).*

of 0.1 m clearly outperforms the larger spin images with larger support radii for the detection of pedestrians. Fine details are more important for the distinction between background and pedestrian. The performance of the other classes is less effected by a specific choice of the radius.

In line with earlier studies on bag-of-words in image-based classification [Chatfield et al., 2011, Coates et al., 2011b], we can also conclude that a larger vocabulary size is beneficial in our application. Especially, in case of cyclists and pedestrians we see a significant increase in performance with more words.

The mixture of bag-of-words combines all three descriptor radii and in the first iteration of the EM algorithm we split the training data depending on the distance of the bounding box into three subsets. However, the hidden variable model $P(h|\boldsymbol{x})$ is learned using distance, volume, and the extent of the three-dimensional bounding box. The mixture of bag-of-words improves the results especially with smaller vocabularies.

Table 5.3 finally shows the resulting detection rates of our approach compared to image-based approaches on the testset. We choose a dictionary size of 800 as this showed the best performance in the experiments on the validation set. The other image-based approaches use a latent variable model of Felzenszwalb et al. [2010] and an extension of this approach by Geiger et al. [2011]. We have to emphasize again that we solely use laser range information and compare all approaches with image-based overlap metrics. Thus, the extracted segments and consequently the bounding boxes are affected by the insufficiencies of the laser rangefinder.

We have to acknowledge certain limitations of our laser-based approach: (1) detection of black objects is not possible due to missing laser range measurements, (2) sparseness of the point cloud that limits the effective range of operations up to 30 m, and (3) ambiguities in appearance often makes it impossible to reliably distinguish certain object classes — some poses of a pedestrian are impossible to distinguish from trunks of trees using only the shape. Visual inspection of the resulting bounding boxes shows that we often see false positive detections of cars in areas with vegetation. Another reason for false positive detections are mismatches between the annotated image-based bounding box and the bounding boxes generated from the laser data. Particularly, as discussed for the segmentation, the car detections are strongly affected by too low overlap values, as we need at least $\omega = 0.7$ minimal overlap

approach	car	pedestrian	cyclist	overall
AdaBoost Teichman et al. [2011]	95.8%	98.3%	98.4%	93.1%
Mixture bow	95.0%	98.3%	98.4%	92.3%
single bow (1.0 m)	91.6%	97.8%	97.7%	87.8%
single bow (2.0 m)	91.7%	97.5%	96.8%	86.7%
single bow (0.5 m)	89.4%	97.8%	96.3%	83.8%

Table 5.4: *Classification accuracy for the STC dataset.*

between ground truth annotation and detections instead of $\omega = 0.5$ overlap for the other classes.

5.5.3 Classification performance

The classification performance of the approach is strongly affected by the overlap of image bounding boxes and bounding boxes extracted from the laser range scan. Thus, we show additional results on a laser only dataset with provided segmentation of the laser scan, the so-called Stanford Track Collection (STC). In consequence, we evaluate only the classification model without hierarchical segmentation and non-maximum suppression.

Table 5.4 show the results on the STC dataset in comparison to an AdaBoost-based approach presented by Teichman et al. [2011]. In contrast to the other experiments, we used 1,600 words for each bag-of-words vocabulary, but the other parameters remained unchanged. The results clearly show the advantage of the mixture of multiple vocabularies over single vocabularies and comparable performance to the state-of-the-art in laser-based classification of segments.

5.5.4 Runtime performance

To evaluate the impact of the sub-sampling strategies, we ran the complete mixture of bag-of-words learning for different voxel resolutions and measured the time needed to determine the label of the segments. We used in both experiments a dictionary size of 800 words and fixed the rest of the parameters to the values of the other experiments. The complete classification of a single frontal laser range scan currently needs 2.75 s on average, where the majority of time (2.72 s, or 98%) is needed to calculate the descriptors. The hierarchical segmentation using three layers needs 10.9 ms on average.

Figure 5.7 shows the average inference times for a single scan, where we separated the time needed to compute the different descriptors. The other computations include the segmentation, sub-sampling, bag-of-word generation, and classification using the learned mixture

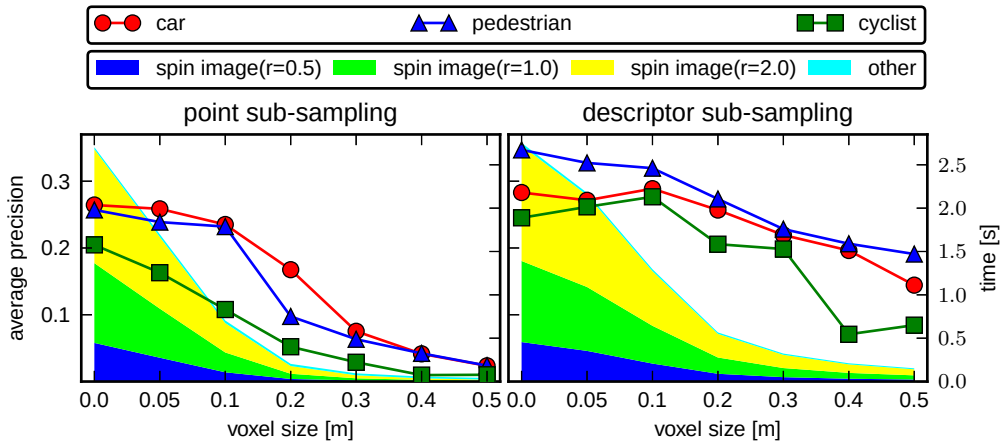


Figure 5.7: *Impact of sub-sampling strategies. The images show the average time needed to infer segment labels for a single scan and the average precision of moderately difficulty bounding boxes. Left plot depicts the performance gains of the point-based sub-sampling and the right plot shows the results of the descriptor sub-sampling. Both strategies can reduce the inference time per scan significantly, but these reductions usually entail declines in the classification accuracy.*

models. In these plots, it is clearly visible that the overall runtime is mainly influenced by the time needed for computing the descriptors (shading in the background). We also plotted the average precision of the moderate difficult bounding boxes (lines in the foreground). The results for bounding boxes of easy difficulty show similar trends.

The left plot of Figure 5.7 shows the reduction in computation time with the point sub-sampling strategy. With a resolution of $\rho = 0.05\,\text{m}$, we already see a significant reduction of the overall computation time from 2.75 s to 1.70 s (37.17% reduction), but also see a decrease in the average precision of the different classes. The average precision for the class “cyclist” decreases with this small resolution by 20%. The same trend is also observable with larger resolutions: the overall runtime is further reduced, but the classification performance is also dropping. These results suggests that dense point clouds are vital for the discrimination of pedestrians and cyclists, but also needed to classify cars correctly. Albeit considerable gains in runtime performance can be achieved with a point sub-sampling, we have to balance these gains with the decline in classification accuracy. The presented results indicate that we should use at most a resolution of 0.05 m to achieve competitive classification results compared to an approach using all laser range data.

The right plot of Figure 5.7 shows the results using the sub-sampling of descriptors to determine a bag-of-words for the descriptors. For these experiments, we used the point clouds with all laser range points and varied the size of voxels. The descriptor sub-sampling also reduces the classification time significantly: a resolution of $\rho = 0.05\,\text{m}$ leads to a reduction of 20.93% (2.18 s compared to 2.75 s) in processing time. This gain is not as large as with the

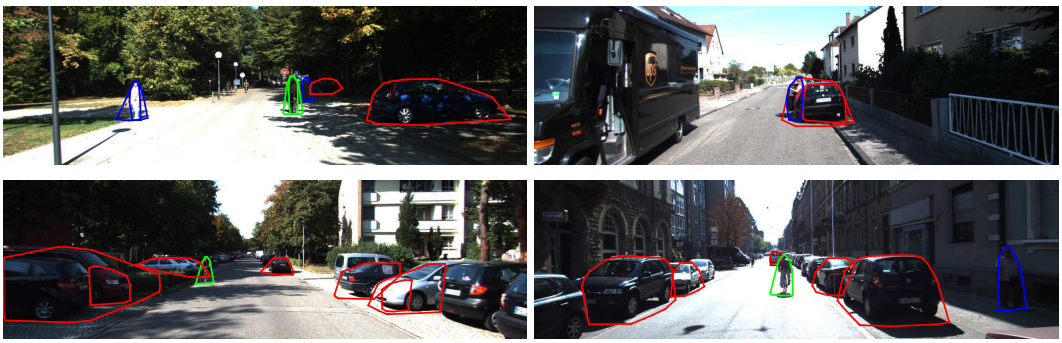


Figure 5.8: *Detection results of the mixture of bag-of-words for cars (red), pedestrian (blue), and cyclists (green). We show the convex hulls of the projected laser range points.*

point sub-sampling of the same resolution, but the reduction incurs not the same reduction in classification performance. Even with a larger voxel size of $\rho = 0.1\,\text{m}$, we can see a similar classification performance of our approach compared to the non-reduced classification. Here, we only observe a slight decrease of the average performance for pedestrians and an slight increase of the average precision of cyclists, which can be explained by the similarity of both classes. We can reduce the overall computation time on average by half (1.30 s, 52.94%) using a resolution of $\rho = 0.1\,\text{m}$. We see a decline in classification performance first with a resolution of $\rho = 0.2\,\text{m}$.

In summary, the descriptor sub-sampling is capable to reduce the overall runtime, while attaining non considerable declines in classification performance. In contrast to this, the point sub-sampling is not able to retain comparable overall classification performance. However, the current implementation does not reuse nearest neighbor queries, which would further reduce the overall computation time considerably. These promising results show that we can possibly reduce the overall runtime to less than one second. Further investigation of parallel processing of descriptors using multiple threaded implementations could lead to practical relevant processing times of under 0.1 s or update rates of 10 Hz.

5.6 Summary

In this chapter, we introduced an approach for segment-based classification using a mixture of different bag-of-words vocabularies. The learning of multiple vocabularies, each using different descriptors as words, was motivated by the characteristics of the laser range scans showing dense point clouds at near range and sparse point clouds at larger distances.

For segmentation, we proposed a novel hierarchical combination of coarse-to-fine segmentations, which allowed us to extract suitable segments more reliably. We learned the hidden

variable model with Expectation Maximization and derived the required expressions for learning of softmax regression models. Finally, we proposed to use the segment hierarchies to filter irrelevant, i.e., duplicated or non-maximal detections, by a greedy non-maximum suppression. The proposed segment-based mixture of bag-of-words needs to calculate a descriptor for each point of a segment, which motivated the investigation of practical methods to improve the efficiency. We proposed to reduce the overall number of laser range points and the reduction of descriptor evaluations needed to construct a bag-of-words. Both approaches use a voxel grid as an efficient implementation and therefore could be applied with only negligible computational overhead. Our extensive experimental evaluation showed that a mixture of bag-of-word classifiers outperforms a single vocabulary bag-of-words approach on a challenging real-world data set.

Future Work. Based on these promising results, we can extend the proposed framework in different ways. First, we would like to investigate the fusion of laser-based detection with other types of information, such as image and map data, to filter false positive detections and improve the detection of 'invisible' objects. The segmentation would be still performed in laser range data, but additionally refined using the image using a approaches like grab cut [Rother et al., 2004], where we can use the projected points to initialize the image-based segmentation. Regions without laser information could be used to extract corresponding regions in the image, and to derive image descriptors for classification.

The stability and efficiency of the mixture learning could also be improved. First, we would like to investigate other methods for learning and encoding [Chatfield et al., 2011, Coates et al., 2011b] to further improve the classification accuracy. Currently, we relearn the complete vocabulary in every iteration from scratch and this could be replaced by methods that change only the relevant parts of the vocabulary. The joint optimization using Expectation Maximization is also a critical point in the proposed classification framework, which certainly needs further investigation regarding efficiency and stability. Local optima in the optimizations could be avoided by random restarts and simulated annealing [Prince, 2012]. But also the overall efficiency could be improved by hard negative mining [Felzenszwalb et al., 2010], i.e., explicitly searching for false positive background segments to reduce the overall number of needed background segments.

Chapter 6

Conclusions

The dissertation covered the classification of three-dimensional laser range data in outdoor environments. As motivated earlier, a classification approach usually comprises of two parts in traditional machine learning: the features and the classification model. In consequence, both parts should be considered if we strive after a robust and efficient classification approach. Additionally, the approaches presented in this thesis must be able to process a massive amount of data, where the processed point clouds show a distance dependent sparseness. Our main contributions and insights on both ends of the pipeline, which resulted in more consistent classification at different ranges, can be recapitulated as follows.

First, we thoroughly investigated the softmax regression and Functional Max-Margin Markov Networks for point-wise classification using three real-world outdoor datasets. Our extensive experimental evaluation showed that a global reference frame and large descriptors improve the classification performance significantly. A simple, but highly efficient softmax regression attained competitive classification performance that is comparable to the more complex graph-based collective classification by Functional Max-Margin Markov Networks. However, regarding label consistency showed the collective approach superior performance compared to the local approach.

Second, we proposed the combination of similarity-preserving hashing and softmax regression to further improve the classification performance and label consistency of the softmax regression for object classes showing varying local appearances. To this end, we presented a two-stage learning algorithm and an inference scheme exploiting the hash-based similarities. Our results indicate that the combination improves the label consistency considerably and achieves this improvement with little computational overhead compared to ordinary softmax regression.

Third, we introduced a novel segment-based classification approach, which uses multiple softmax regressions with different feature representations taking the distance dependent sparseness of the point clouds into account. Main ingredient of the proposed classification approach are bag-of-words and the weighting of multiple models based on segment properties, which is learned jointly with the individual classifiers. In conjunction, we presented an efficient and novel hierarchical segmentation method, which allows to extract complete and consistent segments over a wide range of distances. Finally, the combination of the hierarchical segmentation and the proposed classification model allowed use to filter irrelevant segments generated by the over-complete segmentation including over- and under-segmentations. We evaluated both approaches, the segmentation and the novel classification model, on challenging real-world datasets relevant for autonomous driving. We demonstrated the superior performance of the hierarchical segmentation compared to a common segmentation approach and experimentally validated that a combination of multiple vocabularies outperforms approaches using a single vocabulary.

On the basis of the presented results, there are multiple promising avenues for future research to further enhance the accuracy and runtime of the proposed approaches.

GPU-based descriptor computation. In the beginning of this thesis, we also investigated the time needed to compute descriptors and showed that these computations cannot match the high update rates, say 10 Hz, of modern laser range sensors. A straightforward solution might be the reduction of the amount of data, but we inevitably lose valuable information at some point. In Chapter 5, “Segment-based Classification,” we showed this decline in classification performance in case of the bag-of-words generation if we reduce the amount of laser returns or the number of descriptor evaluations. Thus, we have to improve the single descriptor computation itself to attain considerable speed-ups in the classification process. A possible loophole might be the usage of concurrency offered by general-purpose computing on graphics processing units (GPGPU). In our current implementation, all descriptors are evaluated sequentially, but all presented approaches are inherently parallelizable: each descriptor can be computed independently of another descriptor. However, the special hardware architecture needs some specifically suited algorithms and also carefully designed memory transfers from the host to the graphics device to harness the full power offered by GPUs [Owens et al., 2008, Park et al., 2011]. Both design considerations can affect the efficiency of the concurrent execution on modern GPUs significantly and makes the development of concurrent programs non-trivial, i.e., simple separation of independent calculations in threads will not reach the potential speed-ups. Nonetheless, the potential gains in efficiency makes this direction of future research attractive.

End-to-End learning. Driven by the availability of large-scale datasets, so-called end-to-end classification attracted increasing interest in machine learning recently [Coates et al.,

2013, Krizhevsky et al., 2012, Le et al., 2012]. Here, the goal is to learn the complete pipeline of classification: starting with the features to the final model predicting the labels. Multiple layers of increasingly complex features allowed such approaches to attain remarkable improvements in the state-of-the-art in image-based object recognition [Krizhevsky et al., 2012]. Recently, several large-scale datasets of annotated laser range scans became available [Geiger et al., 2012, Teichman et al., 2011] and this opens the door to move in the same direction as with image data retrieved from the web. Instead of tedious manual feature engineering, one may use the vast amounts of data to learn even millions of parameters for hierarchies of features. In Chapter 5, "Segment-based Classification," we took a first step in this direction and learned a mid-level feature representation on basis of local descriptors. Extending this to learn also the used local descriptors seems to be an obvious next step, which relieves us from the design of suitable feature representations.

Online learning. All classification approaches in this thesis are trained before their potential application in the real world. In consequence, the trained models depend heavily on the collected training data. However, generating datasets that really capture the variety of the real world in all aspects is surprisingly complicated and prone to introduce dataset bias [Torralba and Efros, 2011]. Indeed, the variability of the appearance of all object classes is infinite. A solution to this inherent problem might be to adapt the classifier and integrate new data online, i.e., we use unseen data to modify the classification model at inference time. We ultimately aim at learning new appearances of objects, which were not available while training the classification approach.

Combination with Tracking. In the discussion of the previous chapters, we omitted the circumstance that we usually get a stream of observations from the mobile platform. We classified the potentially sequential data sequentially and ignored that we can observe parts of the laser range scans, which correspond to the exact same object at different times. The only difference might be the view onto the object as either the robot or the object changed its position. Another possible change in appearance might result from different articulations of non-rigid objects, like pedestrians. A promising avenue for future research is to exploit the recurrent occurrences of objects. The tracking of objects, i.e., association of past measurements to current measurements constituting tracks of the same object, reveals these recurrences. Since every consistent track must have a distinct label, we could resolve inconsistencies in label assignment of the segment classifier.

Appendix A

Probability Theory

In a large part of the derivations of this thesis, we used basic concepts from probability theory. In this part, we will briefly summarize the most important definitions, lemmas, and theorems underlying these derivations. This part is mainly based on the very extensive introduction of Koller and Friedman [2009].

Let Ω be the *outcome space* of an event, such as $\{\text{head}, \text{tail}\}$ for a coin flip, and $\mathcal{S}$ the set of measurable *events*, which are subsets of Ω including $\emptyset$.

outcome space
events

Definition 1. [Koller and Friedman, 2009] A probability distribution P over $(\Omega, \mathcal{S})$ is a mapping from events in $\mathcal{S}$ to real values that satisfies the following conditions:

Probability distribution

1. $\forall \alpha \in \mathcal{S} : P(\alpha) \geq 0$.
2. $P(\Omega) = 1$.
3. $\alpha, \beta \in \mathcal{S}, \alpha \cap \beta = \emptyset : P(\alpha \cup \beta) = P(\alpha) + P(\beta)$.

Usually, we are not directly interest in outcomes, but other properties determined by these outcomes. Random variables, denoted by upper case roman letters, are formally functions that map outcomes $\alpha \in \Omega$ to values. We write instead of a function $X(\alpha) = x$ simply $X = x$ and denote the set of values a random variable X can take as $Val(X)$. Note, multiple events $\alpha, \beta \in \Omega$ can lead to the same value of a random variable. In robotics, we often are interested in the state X of an autonomous system and its observations Z as random variables, which are all determined by the current world state.

random variable

Since random variables are simply functions mapping events to values, we can transfer the concept of probability distributions to random variables. The probability distribution over X is then defined as $P_X(X = x) = P(\{\alpha \in \Omega | X(\alpha) = x\})$. From this consideration follows directly the following properties of probability distributions over random variables:

Corolary 1. Let P be a probability distribution, X a random variable, and the probability distribution over X be $P_X(X = x) = P(\{\alpha \in \Omega | X(\alpha) = x\})$.

1. $\forall x \in Val(X) : P_X(X = x) \geq 0$.
2. $P_X(X \in Val(X)) = 1$.
3. $\{x, y\} \subset Val(X), x \neq y : P_X(X = x \cup X = y) = P_X(X = x) + P_X(X = y)$.

In the following, we continue the derivations dropping the decorator X from the probability distribution $P(X)$ over random variable X. Let X and Y be (discrete) random variables over the same outcome space Ω. The *joint distribution* of X and Y will be denoted by $P(X, Y)$ and models the probability of the occurrence of $X = x$ *and* $Y = y$. Since $P(X, Y)$ is a valid probability distribution, we can get $P(X)$ from $P(X, Y)$ by *marginalization*:

joint distribution

marginalization

$$P(X) = \sum_{y \in Val(Y)} P(X, Y) \qquad \text{(Marginalization).} \tag{A.1}$$

conditional probability

The *conditional probability* distribution $P(X|Y)$ defined by

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \qquad \text{(Conditional Probability)} \tag{A.2}$$

models our belief over X, if we know that Y has the specific value y, and is for itself a valid probability distribution.

chain rule

Using the definition of the conditional distribution, Equation A.2, and the marginalization, Equation A.1, we can derive easily the *chain rule* for joint distributions:

$$\begin{aligned} P(X_1, \ldots, X_n) = {} & P(X_1|X_2, \ldots, X_n)P(X_2|X_3, \ldots, X_n) \\ & \cdots P(X_{n-1}|X_n)P(X_n) \qquad \text{(Chain Rule).} \end{aligned} \tag{A.3}$$

Bayes' rule

The *Bayes' rule* follows directly from the definition of conditional probability

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \qquad \text{(Bayes' Rule).} \tag{A.4}$$

Bayes' rule might look like a trivial formula, but it plays an important role in inference. Using Bayes' rule, we can invert the conditional probability and express $P(X|Y)$ by the conditional probability $P(Y|X)$. This simple conversion is the key to inference of otherwise complicated and impossible to model probability distributions. In literature, $P(X)$ is commonly called *prior* distribution, since it represents our belief *before* we have any new knowledge. Likewise, $P(X|Y)$ is often referred to as *posterior* distribution, since it represents our belief *after* seeing the value of Y.

prior

posterior

To motivate the importance of the Bayes' Rule, we shortly present an example from robotics and show how the aforementioned rules are applied to manipulate probability distributions in practice. Localization is a classical application in robotics and we want to infer $P(X_1|Z_1)$, i.e., the first robot state X_1 given the first observation Z_1, without any additional motion. Imagine, we have a robot with a sensor, which measures the distance into a single direction at a time and we can therefore improve our estimate of the location over time by measuring other parts of the environment. Directly modeling the distribution $P(X_1|Z_1)$ using a parametric model is very complex, since we have to take multiple effects into account. But if we apply Bayes' rule, we get can express the same probability distribution using quantities we can effectively model, $P(Z_1|X_1)$ and $P(X_1)$.

$$P(X_1|Z_1) = \frac{P(Z_1|X_1)P(X_1)}{P(Z_1)} \tag{A.5}$$

$$= \eta P(Z_1|X_1)P(X_1) \tag{A.6}$$

Note, the denominator in Equation A.5 does not depend on X_1 and is therefore the same for all locations. Thus, it is possible to calculate the unnormalized posterior distribution $\tilde{P}(X_1|Z_1)$ and normalizing all values afterwards. $P(Z_1|X_1)$ is called the observation model and is usually simpler to model. $P(X_1)$ is our prior belief over all locations without sensing anything. Furthermore, it is usually more natural to model how likely it is to observe something rather than the other way around.

Another important concept for effective inference is conditional independence.

(Conditional) Independence

Definition 2. [Koller and Friedman, 2009] Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be sets of random variables. $\mathcal{X}$ is *conditionally independent* of $\mathcal{Y}$ given $\mathcal{Z}$, if

$$P(\mathcal{X}, \mathcal{Y}|\mathcal{Z}) = P(\mathcal{X}|\mathcal{Z})P(\mathcal{Y}|\mathcal{Z}) \tag{A.7}$$

for all possible values of $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$. We denote conditional independence shortly as $(\mathcal{X} \perp \mathcal{Y}|\mathcal{Z})$. We simply say $\mathcal{X}$ and $\mathcal{Y}$ are *independent*, if $\mathcal{Z} = \emptyset$.

In robotics, we often exploit conditional independence to simplify the derivation of a probabilistic model. The independence of variables drastically reduces the amount of parameters that need to be represented and, consequently, leads to tractable inference.

We now extend our example for incorporating measurements over time, $Z_1, \ldots, Z_t = Z_{1:t}$. Hence, we are now interested in modeling $P(X_t|Z_{1:t})$ and the state depends on all previous measurements. We can apply Bayes' rule and get a quite complicated model:

$$P(X_t|Z_{1:t}) = \eta P(Z_t|X_t, Z_{1:t-1})P(X_t|Z_{1:t-1}) \tag{A.8}$$

In probabilistic robotics, it is usually assumed that the state X_t only depends on the previous state X_{t-1}, the so-called Markov assumption. Furthermore, the current observation Z_t is conditionally independent of past observations $Z_{1:t-1}$ given the current state X_t. Hence, the state X_t implicitly incorporates already all past observations. Thus, we can simplify the more complex model using these implied conditional independences

$$P(X_t|Z_{1:t}) = \eta P(Z_t|X_t)P(X_t|Z_{1:t-1}), \tag{A.9}$$

and see that the first term corresponds again to our observation model. To update our belief of the location at time t, our goal is to derive a recursive formula depending on the last belief $P(X_{t-1}|Z_{t-1:1})$ and we can derive this using our previously introduced machinery

$$\begin{aligned}
&P(X_t|Z_{1:t}) \\
&= \eta P(Z_t|X_t)P(X_t|Z_{1:t-1}) && && \text{(A.10)}\\
&= \eta P(Z_t|X_t) \int P(X_t, X_{t-1}|Z_{1:t-1})\, dx_{t-1} && \text{using (A.1)} && \text{(A.11)}\\
&= \eta P(Z_t|X_t) \int P(X_t|X_{t-1}, Z_{1:t-1})P(X_{t-1}|Z_{1:t-1})\, dx_{t-1} && \text{using (A.3)} && \text{(A.12)}\\
&= \eta P(Z_t|X_t) \int P(X_t|X_{t-1})P(X_{t-1}|Z_{1:t-1})\, dx_{t-1}. && && \text{(A.13)}
\end{aligned}$$

Bayes filter

Equation A.13 corresponds to a *Bayes filter* and allows us to update our posterior incrementally over time. Note that we included everything not depending on X_t in the normalization constant η, which can we determine later to normalize the posterior again. $P(X_t|X_{t-1})$ is usually called motion model and encodes how likely it is to get from one state to another.

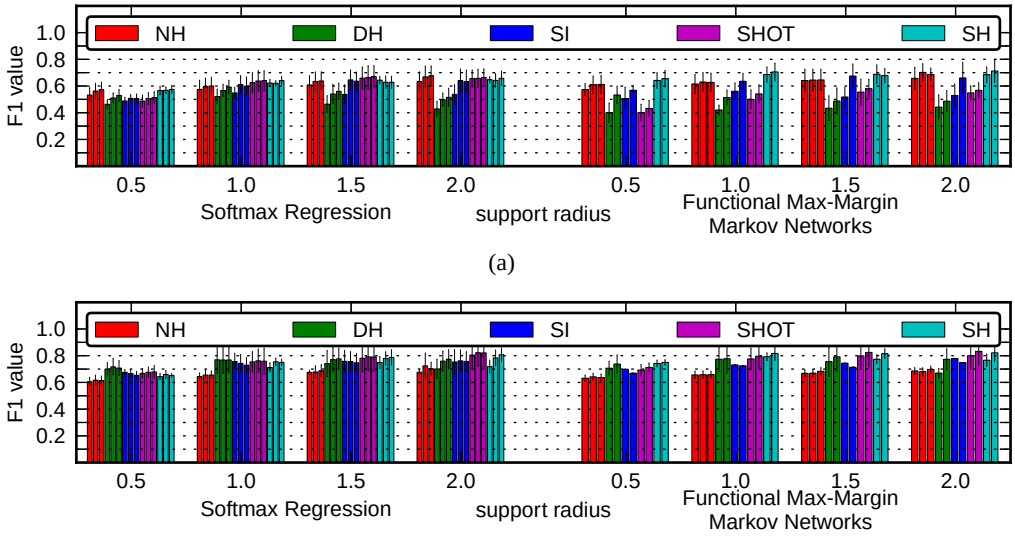
Appendix B

Additional Results

In this chapter, we present additional results of Chapter 3 and Chapter 4. The following diagrams contain all evaluated dimensions in a more condensed representation.

The first three figures show extended results over all evaluated dimensions for the feature evaluation of Chapter 3, “Histogram Descriptors for Laser-based Classification,”. Figure B.1 shows all results generated on the Freiburg dataset using softmax regression and the functional max-margin markov networks. Subfigure (a) shows the descriptor performances using a local reference frame, and subfigure (b) show the same descriptors using a global reference frame. Figure B.2 depicts all results generated on the Pittsburgh dataset. As before, subfigure (a) shows the performance using a local reference frame, and subfigure (b) using a global reference frame. And finally, Figure B.3 shows all results of the Wachtberg dataset.

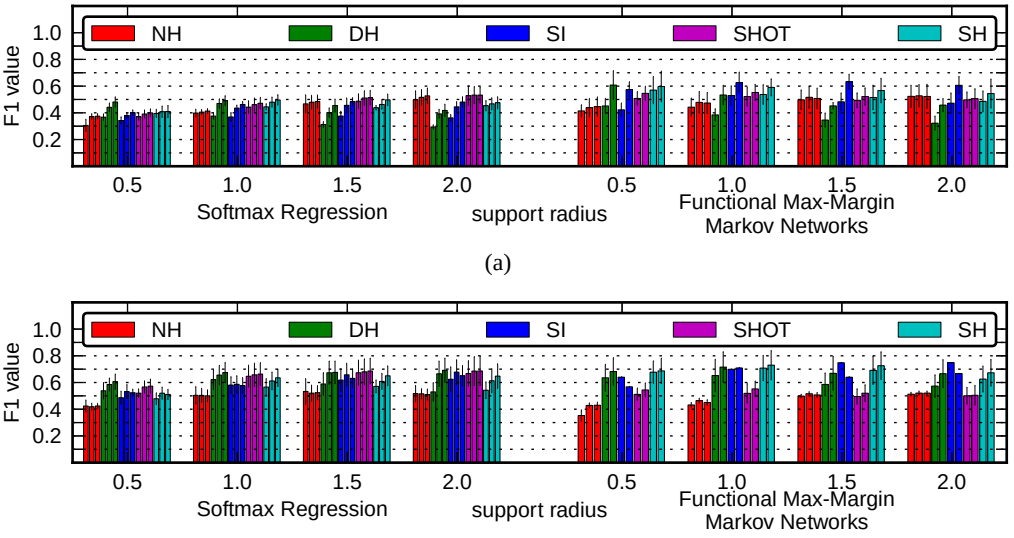
The second part of figures show extended results for the spectrally hashed softmax regression compared to the softmax regression. Figure B.4 show the results for the Freiburg dataset, Figure B.5 show the results for the pittburgh dataset, and Figure B.6 show the results for the Wachtberg dataset.



(a)

(b)

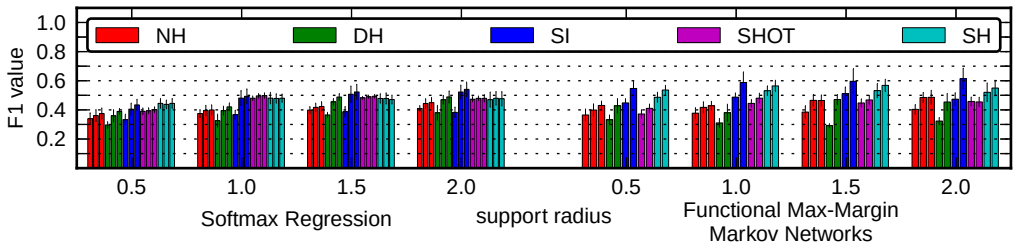
Figure B.1: *Results on the Freiburg dataset. In (a) we used a local reference frame or axis, in (b) the global reference frame was used. Multiple bars for a feature result from different numbers of bins.*



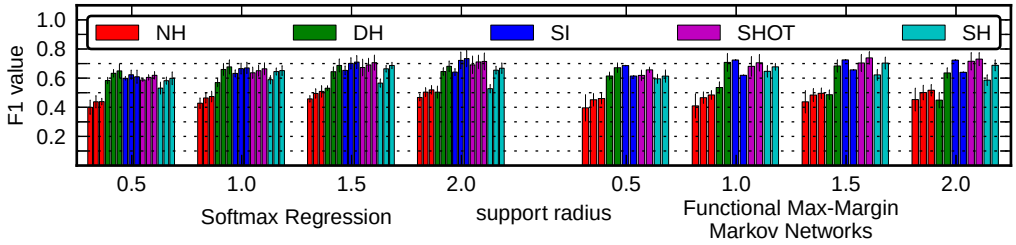
(a)

(b)

Figure B.2: *Results on the Pittsburgh dataset. In (a) we used a local reference frame or axis, in (b) the global reference frame was used. Multiple bars for a feature result from different numbers of bins.*

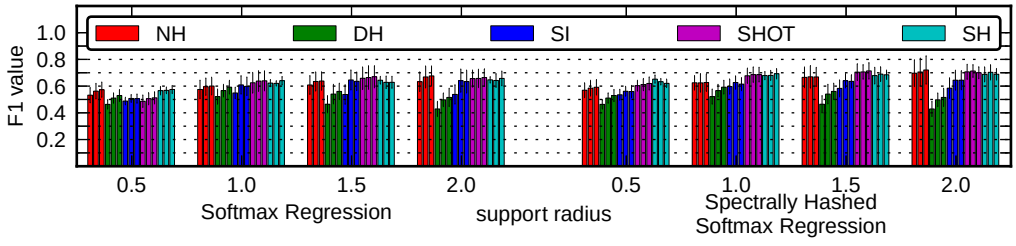


(a)

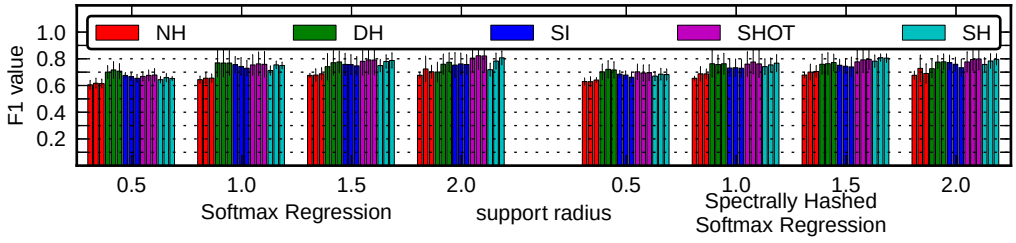


(b)

Figure B.3: *Results on the Wachtberg dataset. In (a) we used a local reference frame or axis, in (b) the global reference frame was used. Multiple bars for a feature result from different numbers of bins.*

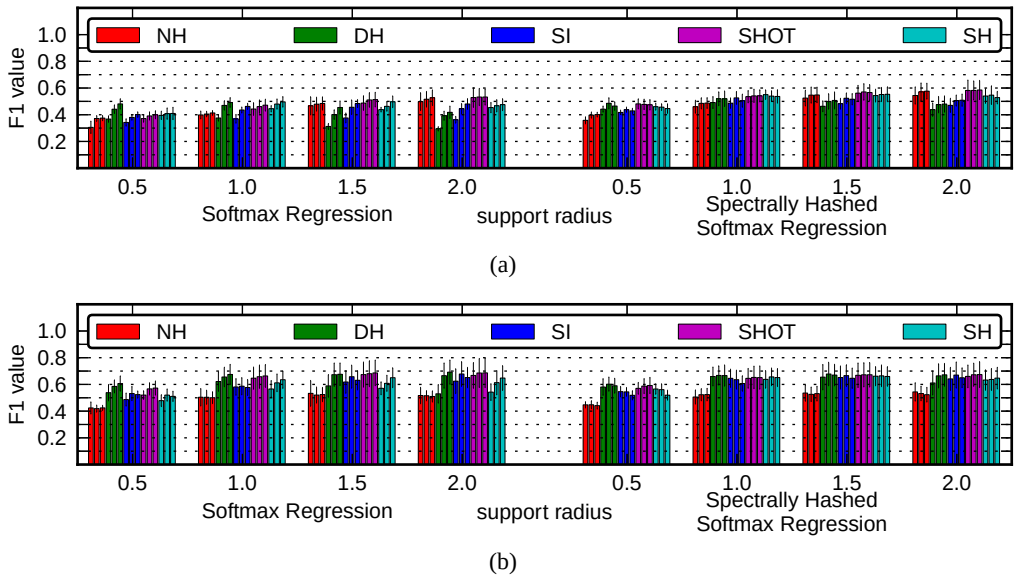


(a)



(b)

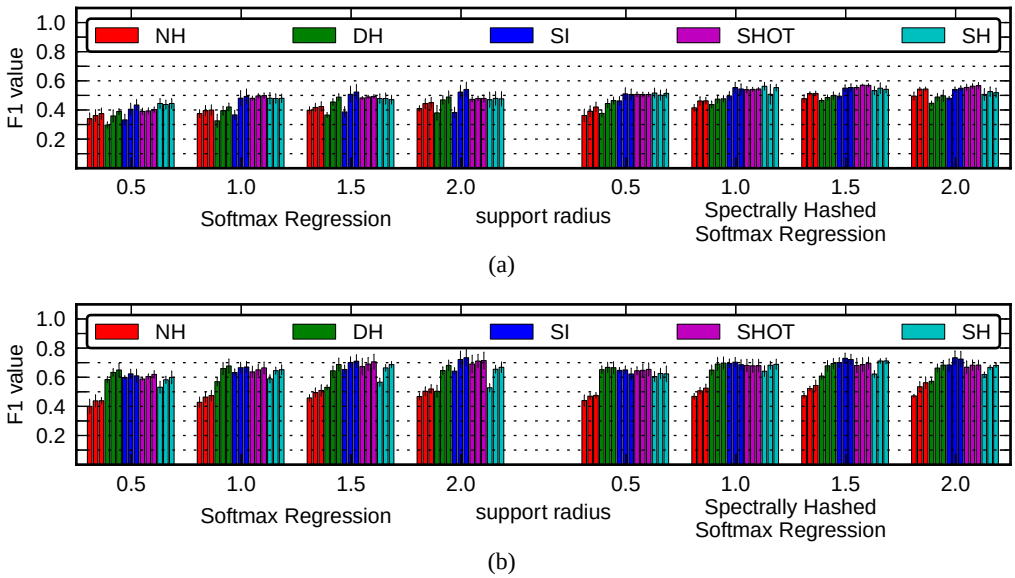
Figure B.4: *Results on the Freiburg dataset. In (a) we used a local reference frame or axis, in (b) the global reference frame was used. Multiple bars for a feature result from different numbers of bins.*



(a)

(b)

Figure B.5: *Results on the Pittsburgh dataset. In (a) we used a local reference frame or axis, in (b) the global reference frame was used. Multiple bars for a feature result from different numbers of bins.*



(a)

(b)

Figure B.6: *Results on the Wachtberg dataset. In (a) we used a local reference frame or axis, in (b) the global reference frame was used. Multiple bars for a feature result from different numbers of bins.*

List of Figures

2.1 Two-dimensional laser rangefinder and service robot 6
2.2 Common laser scanner setups 8
2.3 Subdivisions of octree and k-d tree for the Stanford bunny 10
2.4 Normals of a part of a torus and eigenvectors of a 2D point set 11
2.5 Classification example showing a softmax regression and *k*-nn. 14

3.1 Normal Histogram for curved and flat surfaces. 27
3.2 Spectral shape features and subdivisions of SHOT and spectral histogram. 29
3.3 Local and global reference frame. 30
3.4 Exemplary scans and laser point distribution. 31
3.5 Influence of the reference frame 36
3.6 Influence of the support radius 37
3.7 Influence of the number of bins 38
3.8 Qualitative results of SR and FM3N on the Wachtberg dataset 41
3.9 Time needed for descriptor calculation. 42

4.1 SHSR using different number of bits. 53
4.2 Comparison of SR and SHSR with different support radii 54
4.3 Qualitative results of SR and SHSR on the Wachtberg dataset 57

5.1 Distance-dependency of object point clouds 64
5.2 Segment-based Mixture of Bag-of-Words 65
5.3 Obstacle grid-based segmentation 70
5.4 Hierarchical segmentation example 74
5.5 Segmentation results. 84
5.6 Influence of the vocabulary size 86
5.7 Average precision and runtime of sub-sampling approaches 89
5.8 Detection results of the mixture of bag-of-words 90

B.1 Freiburg results using SR and FM^{3}N. 102
B.2 Pittsburgh results using SR and FM^{3}N. 102
B.3 Wachtberg results using SR and FM^{3}N. 103

B.4 Freiburg results using SR and SHSR. 103
B.5 Pittsburgh results using SR and SHSR. 104
B.6 Wachtberg results using SR and SHSR. 104

List of Tables

3.1 Descriptor parameters. 35
3.2 Precision/Recall Freiburg dataset. 39
3.3 Precision/Recall Pittsburgh dataset 39
3.4 Precision and recall on the Wachtberg dataset. 40

4.1 Precision/Recall Freiburg dataset. 55
4.2 Precision/Recall Pittsburgh dataset 56
4.3 Precision and recall on the Wachtberg dataset. 56
4.4 Inference time of different classifiers. 58

5.1 Overlap results of hierarchical segmentation 84
5.2 Segments per class (easy/moderate bounding boxes) 86
5.3 Results on the testset (easy/moderate). 87
5.4 Classification accuracy for the STC dataset. 88

Bibliography

Agrawal, A., Nakazawa, A., and Takemura, H. (2009). MMM-classification of 3D Range Data. In *Proc. of the International Conference on Robotics and Automation(ICRA)*.

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. (2005). Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 169–176.

Arbeiter, G., Fuchs, S., Bormann, R., Fischer, J., and Verl, A. (2012). Evaluation of 3D Feature Descriptors for Classification of Surface Geometries in Point Clouds. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1644–1650.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The Advantages of Careful Seeding. In *Proc. of the ACM-SIAM Symposium of Discrete Algorithms (SODA)*, pages 1027–1035.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923.

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally Weighted Learning. *AI Review*, 11:11–73.

Barber, D. (2012). *Baysian Reasoning and Machine Learning*. Cambridge University Press.

Behley, J., Kersting, K., Schulz, D., Steinhage, V., and Cremers, A. B. (2010). Learning to Hash Logistic Regression for Fast 3D Scan Point Classification. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5960–5965.

Behley, J., Steinhage, V., and Cremers, A. B. (2012). Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. In *Proc. of the IEEE International Conference on Robotics and Autonmation (ICRA)*, pages 4391–4398.

Behley, J., Steinhage, V., and Cremers, A. B. (2013). Laser-based Segment Classification Using a Mixture of Bag-of-Words. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. to appear.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Boureau, Y.-L., Bach, F., LeCun, Y., and Ponce, J. (2010). Learning Mid-Level Features For Recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2559–2566.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1–2):3–55.

Byrd, R., Lu, P., Nocedal, J., and Zhu, C. (1995). A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208.

Calonder, M., Lepetit, V., Özuysal, M., Trzcinski, T., Strecha, C., and Fua, P. (2012). BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) : (2012)*, 34(7):1281–1298.

Chatfield, K., Lempitsky, V., Vedaldi, A., and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *Proc. of the British Machine Vision Conference (BMVC)*, pages 76.1–76.12.

Chen, Q., Song, Z., Hua, Y., Huang, Z., and Yan, S. (2012). Hierarchical Matching with Side Information for Image Classification. In *Proc. of the IEEE Conference on Computer Vision (CVPR)*, pages 3426–3433.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D. J., and Ng, A. Y. (2011a). Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning. In *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 440–445.

Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y., and Catanzaro, B. (2013). Deep learning with COTS HPC systems. In *Proc. of the International Conference on Machine Learning (ICML)*.

Coates, A., Lee, H., and Ng, A. Y. (2011b). An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 215–223.

Coates, A. and Ng, A. (2011). The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 921–928.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *In ECCV Workshop on Statistical Learning in Computer Vision*, pages 1–22.

Dalal, N. and Triggs, B. (2005). Histogram of Oriented Gradients for Human Detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 886–893.

Daniely, A., Sabato, S., and Shalev-Shwartz, S. (2012). Multiclass Learning Approaches: A Theoretical Comparison with Implications. In *Advances in Neural Information Processing Systems (NIPS)*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the Segmentation of 3D LIDAR Point Clouds. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2798–2805.

Elseberg, J., Magnenat, S., Siegwart, R., and Nüchter, A. (2012). Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics (JOSER)*, 3(1):2–12.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645.

Forsyth, D. A. and Ponce, J. (2012). *Computer Vision: A Modern Approach*. Pearson.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics*, 28(2):337–374.

Friedman, J. H. and Bentley, J. L. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361.

Geiger, A., Wojek, C., and Urtasun, R. (2011). Joint 3D Estimation of Objects and Scene Layout. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1467–1475.

Gong, Y., Kumar, S., Verma, V., and Lazebnik, S. (2012). Angular Quantization-based Binary Codes for Fast Similarity Search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1205–1213.

Gong, Y. and Lazebnik, S. (2011). Iterative Quatization: A Procrustean Approach to Learning Binary Codes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–824.

Gross, H. M., Boehme, H., Schroeter, C., Mueller, S., Koenig, A., Einhorn, E., Martin, C., Merten, M., and Bley, A. (2009). TOOMAS: Interactive Shopping Guide Robots in Everyday Use - Final Implementation and Experiences from Long-term Field Trials. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2005–2012.

Halevy, A., Norvig, P., and Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition.

He, K., Wen, F., and Sun, J. (2013). K-means Hashing: an Affinity-Preserving Quantization Method for Learning Binary Compact Codes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2938–2945.

Himmelsbach, M., Luettel, T., and Wuensche, H.-J. (2009). Real-time Object Classification in 3D Point Clouds Using Point Feature Histograms. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 994–1000.

Himmelsbach, M., v. Hundelshausen, F., and Wuensche, H.-J. (2010). Fast Segmentation of 3D Point Clouds for Ground Vehicles. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, pages 560–565.

Himmelsbach, M. and Wuensche, H.-J. (2012). Tracking and Classification of Arbitraty Objects with Bottom-Up/Top-Down Detection. In *Proc. of the IEEE Intelligent Vehicles Symposium(IV)*, pages 577–582.

Hoeller, F., Röhling, T., and Schulz, D. (2010). Offroad Navigation using Adaptable Motion Patterns. In *Proc. of the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 186–191.

Horn, B. (1984). Extended gaussian images. *Proc. of the IEEE*, 72(12):1656–1678.

Jacobs, R. A., Jordan, M. I., Nowlan, S., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:1–12.

Johnson, A. and Hebert, M. (1999). Using spin images for effcient object recognition in cluttered 3D scenes. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):433–449.

Kaneva, B., Torralba, A., and Freeman, W. T. (2011). Evaluation of Image Features Using a Photorealistic Virtual World. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 2282–2289.

Klasing, K., Wollherr, D., and Buss, M. (2008). A Clustering Method for Efficient Segmentation of 3d Laser Data. In *Proc. of the International Conference on Robotics and Automation(ICRA)*, pages 4043–4048.

Klasing, K., Wolllherr, D., and Buss, M. (2009). Realtime Segmentation of Range Data Using Continuous Nearest Neighbors. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2431–2436.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models*. MIT Press.

Komarek, P. and Moore, A. (2005). Making Logistic Regression A Core Data Mining Tool: A Practical Investigation of Accuracy, Speed, and Simplicity. Technical report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114.

Kulis, B. and Grauman, K. (2012). Kernelized Locality-Sensitive Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(6):1092–1104.

Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., and Burgard, W. (2013). A Navigation System for Robots Operating in Crowded Urban Environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. to appear.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilstic Models for Segmenting and Labeling Sequence Data. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 282–289.

Lai, K. and Fox, D. (2010). Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *International Journal of Robotics Research*, 29(8):1019–1037.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond Bag of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., and Ng, A. Y. (2012). Building High-level Features Using Large Scale Unsupervised Learning. In *Proc. of the International Conference on Machine Learning (ICML)*.

Levinson, J. and Thrun, S. (2010). Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In *Proc. of the IEEE International Conference on Robotics and Automation(ICRA)*, pages 4372–4378.

Li, B., Godil, A., Aono, M., Bai, X., Furuya, T., Li, L., Lopez-Sastre, R. J., Johan, H., Ohbuchi, R., Redondo-Cabrera, C., Tatsuma, A., Yanagimachi, T., and Zhang, S. (2012). SHREC'12 Track: Generic 3D Shape Retrieval. In *Proc. of the Eurographics Workshop on 3D Object Retrieval (3DOR)*, pages 119–126.

Li, P., Shrivastrava, A., Moore, J., and König, A. C. (2011). Hashing Algorithms for Large-scale Learning. In *Advances in Neural Information Processing Systems*.

Lim, E. H. and Suter, D. (2007). Conditional Random Field for 3D point clouds with Adaptive Data Reduction. In *International Conference on Cyberworlds*, pages 404–408.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110.

Lu, Y. and Rasmussen, C. (2012). Simplified Markov Random Fields for Efficient Semantic Labeling of 3D Point Clouds. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 2690–2697.

Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge University Press.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The Office Marathon: Robust Navigation in an Indoor Office Environment. In *Proc. of the IEEE International Conference on Robotics and Autonmation (ICRA)*.

Marton, Z.-C., Pangericic, D., Blodow, N., Kleinehellefort, J., and Beetz, M. (2010). General 3D Modelling of Novel Objects from a Single View. In *Proc. of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 3700–3705.

Meagher, D. (1982). Geomertic Modeling Using Octree Encoding. *Computer Graphics and Image Processing*, 19:129–147.

Medioni, G., Lee, M.-S., and Tang, C.-K. (2000). *A Computational Framework for Segmentation and Grouping*. Elsevier.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(10):1615–1630.

Moosmann, F., Pink, O., and Stiller, C. (2009). Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion. In *Proc. of the IEEE Intelligent Vehicles Symposium(IV)*, pages 215–220.

Moosmann, F. and Stiller, C. (2010). Velodyne SLAM. In *Proc. of the IEEE Intelligent Vehicles Symposium(IV)*, pages 393–398.

Moosmann, F., Triggs, B., and Jurie, F. (2007). Fast discriminative visual codebooks using randomized clustering forests. In *Advances in Neural Information Processing Systems (NIPS)*.

Munoz, D., Bagnell, J. A. D., Vandapel, N., and Hebert, M. (2009a). Contextual Classification with Functional Max-Margin Markov Networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 975–982.

Munoz, D., Vandapel, N., and Hebert, M. (2008). Directional Associative Markov Network for 3-D Point Cloud Classification. In *Proc. of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 63–70.

Munoz, D., Vandapel, N., and Hebert, M. (2009b). Onboard Contextual Classification of 3-D Point Clouds with Learned High-order Markov Random Fields. In *Proc. of the IEEE International Conference on Robotics and Automation(ICRA)*.

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., and Phillips, J. C. (2008). GPU Computing. *Proceedings of the IEEE*, 96(5):879–899.

Park, K., Singhal, N., Lee, M. H., Cho, S., and Kim, C. W. (2011). Design and Performance Evaluation of Image Processing Algorithms on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 22(1):91–104.

Pastuszka, R. (2013). Untersuchungen zur effizienten Verarbeitung von dreidimensionalen Laserentfernungsdaten. Bachelor thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.

Patterson, A., Mordohai, P., and Daniilidis, K. (2008). Object Detection from Large-Scale 3D Datasets using Bottom-up and Top-down Descriptors. In *Proc. of the European Conference on Computer Vision(ECCV)*, pages 553–566.

Petrovskaya, A. and Thrun, S. (2009). Model Based Vehicle Detection and Tracking for Autonomous Urban Driving. *Autonomous Robots*, 26(2-3):123–139.

Pharr, M. and Humphreys, G. (2010). *Physically based Rendering*. Morgan Kaufmann.

Prince, S. (2012). *Computer Vision: Models, Inference and Learning*. Cambridge University Press.

Ratliff, N., Bagnell, J. A., and Srinivasa, S. (2007). Imitation Learning for Locomotion and Manipulation. In *Proc. of the IEEE Humanoids*.

Rifkin, R. and Klautau, A. (2004). in Defence of One-Vs-All Classification. *Journal of Machine Learning Reasearch(JMLR)*, 5:101–141.

Rother, C., Kolmogorov, V., and Blake, A. (2004). "GrapCut" — Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics (TOG) – Proc. of ACM SIGGRAPH*, 23(3):309–314.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1–3):157–173.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D Registration. In *Proc. of the International Conference on Robotics and Automation(ICRA)*, pages 3212–3217.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*.

Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. (2008). Learning Informative Point Classes for the Acquisition of Object Model Maps. In *Proc. of the International Conference on Control, Automation, Robotics, and Vision (ICARCV)*.

Salakhutdinov, R. and Hinton, G. (2009). Semantic Hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

Schöler, F., Behley, J., Steinhage, V., Schulz, D., and Cremers, A. B. (2011). Person Tracking in Three-Dimensional Laser Range Data with Explicit Occlusion Adaption. In *Proc. of the International Conference on Robotics and Automation(ICRA)*, pages 1297–1303.

Sivic, J. and Zisserman, A. (2003). Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 1470–1477.

Sivic, J. and Zisserman, A. (2009). Efficient Visual Search Cast as Text Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(4):591–606.

Spinello, L., Arras, K. O., Triebel, R., and Siegwart, R. (2010). A Layered Approach to People Detection in 3D Range Data. In *Proc. of the Conference on Artificial Intelligence(AAAI)*.

Spinello, L., Luber, M., and Arras, K. O. (2011). Tracking People in 3D Using a Bottom-Up Top-Down Detector. In *Proc. of the International Conference on Robotics and Automation(ICRA)*, pages 1304–1310.

Steder, B., Ruhnke, M., Grzonka, S., and Burgard, W. (2011a). Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature based Relative Pose Estimation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 1249–1255.

Steder, B., Rusu, R. B., Konolige, K., and Burgard, W. (2011b). Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries. In *Proc. of the International Conference on Robotics and Automation(ICRA)*, pages 2601–2608.

Swift, J. J., Johnson, J. A., Morton, T. D., Crepp, J. R., Montet, B. T., Fabrycky, D. C., and Muirhead, P. S. (2013). Characterizing the Cool KOIs. IV. Kepler-32 as a Prototype for the Formation of Compact Planetary Systems throughout the Galaxy. *The Astrophysical Journal*, 764(1):105–119.

Tangelder, J. W. and Veltkamp, R. C. (2008). A survey on content based 3D shape retrieval methods. *Journal of Multimedia Tools and Applications*, 39(3):441–471.

Taskar, B., Chatalbashev, V., and Koller, D. (2004). Learning Associative Markov Networks. In *Proc. of the International Conference on Machine Learning (ICML)*.

Teichman, A., Levinson, J., and Thrun, S. (2011). Towards 3D Object Recognition via Classification of Arbitrary Object Tracks. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4034–4041.

Teichman, A. and Thrun, S. (2012). Tracking-based semi-supervised learning. *International Journal of Robotics Research(IJRR)*, 31(7):804–818.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). MINERVA: A second generation mobile tour-guide robot. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1999–2005.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilisitic Robotics*. MIT Press.

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

Tombari, F., Salti, S., and Stefano, L. D. (2010). Unique Signatures of Histograms for Local Surface Description. In *Proc. of the European Conference on Computer Vision(ECCV)*, pages 356–369.

Torralba, A. and Efros, A. A. (2011). Unbiased Look at Dataset Bias. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528.

Torralba, A., Fergus, R., and Freeman, W. T. (2008a). 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(11):1958–1970.

Torralba, A., Fergus, R., and Weiss, Y. (2008b). Small Codes and Large Image Databases for Recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 1–8.

Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–762–II–769.

Triebel, R., Kersting, K., and Burgard, W. (2006). Robust 3D Scan Point Classification using Associative Markov Networks. In *Proc. of the International Conference on Robotics and Automation(ICRA)*, pages 2603–2608.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D. (2008). Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Jorunal of Field Robotics (JFR)*, 25(8):425–426.

van der Sande, K. E. A., Uijlings, J. R. R., Gevers, T., and Smeulders, A. W. M. (2011). Segmentation as Selective Search for Object Recognition. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 1879–1886.

Velodyne Lidar Inc. (2010). High Definition Lidar HDL-64E S2 Datasheet. http://www.velodyne.com/lidar/products/specifications.aspx.

Weiss, Y., Torralba, A., and Fergus, R. (2008). Spectral Hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1753–1760.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Wolpert, D. H. (1996). The lack of a prior distinctions between learning algorithms and the existence of a priori distinctions between learning algorithms. *Neural Computation*, 8:1391–1421.

Xiong, X., Munoz, D., Bagnell, J. A., and Hebert, M. (2011). 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *Proc. of the IEEE International Conference on Robotics and Automation(ICRA)*, pages 2609–2616.

Xu, Z., Kersting, K., and Bauckhage, C. (2012). Efficient Learning for Hashing Proportional Data. In *Proc. of the IEEE International Conference on Data Mining (ICDM)*, pages 735–744.

Index

Symbols

k-nearest neighbors 9
classifier 19

A

average precision 83

B

Bayes filter 100
Bayes' rule 98
Bayesian approach 16
bias/variance trade-off 20
bounding box 81
intersection 82

C

chain rule 98
complete scan 9
conditional probability 98
covariance matrix 12
cross-validation 20
curse of dimensionality 20

D

decision boundary 15
decision region 14
descriptor 26
distribution histogram 28

E

events 97
Expectation Maximization 76

F

F_1 measure 33
feature space 14
feature vector 14
flood fill 69

G

grid map 69

H

histogram descriptors 26

I

indicator function 18

J

Jensen's inequality 76
joint distribution 98

K

k-d tree 10

L

laser rangefinder

rotating 8
sweeping 7
three-dimensional 6
tilting 8
two-dimensional 6
LiDAR device 5
likelihood 13
linear separable 16
local features 27

M

marginalization 98
matching 82
maximum a posteriori 16
maximum likelihood 16
model capacity 13
model parameters 15

N

normal histogram 27
normalization constant 27

O

obstacle grid map 69
octree 10
outcome space 97
outliers 15
over-fitting 21

P

point cloud 6
posterior distribution 13, 98
precision
class-wise 33
ranked 83
precision-recall curve 83
interpolated 83
prior distribution 13, 98

R

radius neighbors 9
recall
class-wise 33
ranked 83
reference frame
global 30
local 30

S

scan 6
segment 68
tree 73
SHOT 28
similarity-preserving hashing 48
softmax 17
softmax regression 17–19
spectral histogram 29
spectral shape features 29
spin image 28
supervised learning 13
support 23

T

test set 14
time-of-flight 6
training set 13

V

validation error 20
validation set 20
Velodyne HDL-64E S2 9
voxel grid 80